Trading Safety Versus Performance: Rapid Deployment of Robotic Swarms with Robust Performance Constraints

Yin-Lam Chow
Institute for Computational & Mathematical Engineering
Stanford University
Stanford, CA 94305
Email: ychow@stanford.edu

Marco Pavone
Aeronautics and Astronautics
Stanford University
Stanford, CA 94305
Email: pavone@stanford.edu

Brian M. Sadler
Army Research Lab
Adelphi, MD 20783
Email: brian.m.sadler6.civ@mail.mil

Stefano Carpin
School of Engineering
University of California
Merced, CA 95343
Email: scarpin@ucmerced.edu

In this paper we consider a stochastic deployment problem, where a robotic swarm is tasked with the objective of positioning at least one robot at each of a set of pre-assigned targets while meeting a temporal deadline. Travel times and failure rates are stochastic but related, inasmuch as failure rates increase with speed. To maximize chances of success while meeting the deadline, a control strategy has therefore to balance safety and performance. Our approach is to cast the problem within the theory of constrained Markov Decision Processes, whereby we seek to compute policies that maximize the probability of successful deployment while ensuring that the expected duration of the task is bounded by a given deadline. To account for uncertainties in the problem parameters, we consider a robust formulation and we propose efficient solution algorithms, which are of independent interest. Numerical experiments confirming our theoretical results are presented and discussed.

1 Introduction

Recent technological advances have made possible the deployment of robotic swarms comprising hundreds of small, minimalistic, and inexpensive ground, air, and underwater mobile platforms [1]. The potential advantages of robotic swarms are numerous. For example, it is possible to reduce the total implementation and operation cost, and add flexibility, robustness, and modularity with respect to monolithic approaches. Accordingly, planning and control for multi-robot systems (including robotic swarms) has received tremendous attention in the past decade from the control, robotics, and artificial intelligence communities, see [2] and references therein.

When deploying robotic swarms, safety and speed often come as *contradicting* objectives: the faster it is required for robots to accomplish a task, the higher the chance that the task is not accomplished. For example, robot batteries discharge more rapidly when robots operate at high velocities, thus shortening operational time and increasing the probability that the assigned task will not be completed. Simi-

larly, sensor accuracy often decreases with speed, thereby interfering with the ability of a robot to accomplish its mission (see, e.g., [3]). Accordingly, the goal of this paper is to devise analysis tools and control algorithms to address such safety/performance trade-off within the context of the *stochastic deployment problem*, whereby it is desired that a robotic swarm deploys within a given map so that each of a set of target locations is reached by at least one robot. The set up is stochastic in the sense that traversal times and robot failures represent stochastic events, and we propose an approach that is robust to additive errors in the modeling of traversal times.

Deployment problems have been recently studied along a number of dimensions and with a variety of techniques, including coverage control through locational optimization [4, 5], robot dispersion through minimalistic control strategies (e.g., random walks) [6–8], workload sharing in dynamic environments through distributed optimization [9], deployment with temporal logic specifications, e.g., through formal methods tools [10–12], deployment under communication constraints through Partially Observable Markov Decision Processes [13–15], and strategic deployment with complex mission specifications through the theory of constrained Markov Decision Processes [16]. This list is necessarily incomplete (see [2] for a thorough literature review on this problem), but it however outlines the main facets and tools employed for this problem in the last decade, and shows the research shift from results mostly focusing on the quality of the steady-state solution (e.g., coverage control) to studies enforcing constraints on the transient (e.g., strategic deployment).

Yet, despite the wealth of results available for the deployment problem and, more in general, for multi-robot coordination, to the best of the authors' knowledge no results exist today that explore the trade-off between safe and rapid deployment in stochastic environments (perhaps with the exception of [17], where stochasticity is shown to increase fault tolerance, but no temporal constraints are considered). The purpose of this paper is to bridge this gap. Our strategy is to frame the problem within the paradigm of constrained Markov Decision Processes (CMDPs). Specifically, our contributions are as follows:

1. We show how the rapid *single-robot* deployment problem, where it is desired to trade-off safety versus performance, can be modeled as a *robust* CMDP (RCMDP). The robustness aspect of the formulation stems from the fact that the parameters of the constraint cost functions belong to a (known) uncertainty set.
2. We show that an optimal policy for a RCMDP can be determined using a linear optimization problem based on the concept of occupation measures developed for CMDPs.
3. We demonstrate how the linear optimization problem can be efficiently solved using a transformation that reduces the number of constraints from (possibly) exponential to linear in the size of the problem.
4. We illustrate how RCMDPs can be used to determine both centralized and (minimalistic) decentralized policies to solve the rapid *multi-robot* deployment problem.

The rest of the paper is organized as follows. Section 2 formally introduces the CMDP model and provides well-known results, together with pointers to selected references. The RCMDP model is introduced in Section 2 as well. Section 3 defines the rapid deployment problem for both the single- and multi-robot cases, and shows how RCMDPs together with a task assignment strategy can be used for its solution. An efficient algorithm for the RCDMP problem is presented in Section 4, while in Section 5 we describe the task assignment algorithm used in our approach. Section 6 presents the overall solving algorithm. Detailed numerical experiments are presented in Section 7, and in Section 8 we draw our conclusions and provide directions for future research.

2 Background Material

In this section we provide a brief summary about Markov Decision Processes and constrained Markov Decision Processes. We limit our discussion to finite MDPs and we embrace the notation used in [18]. For a complete treatment on this subject, the reader is referred to [19,20] and [21] for MDPs and CMDPs, respectively.

2.1 Total Cost Markov Decision Processes

A *stationary*, finite MDP is a quadruple $\mathbf{X}, A, c, \mathcal{P}$ where:

1. $\mathbf{X}$ is a finite set of $n = |\mathbf{X}|$ states. The temporal evolution of the state of an MDP is stochastic and the state at time t is given by the random variable X_t.
2. A is a collection of n finite sets. Each set is indicated as $A(x)$ and represents the set of actions that can be applied when the system is in state x. It is convenient to define the set $\mathcal{K} = \{(x,a) : x \in \mathbf{X}, a \in A(x)\}$. $\mathcal{K}$ is the set of allowable state/action pairs. In general, the action taken at time t is a random variable indicated by the letter A_t.
3. $c : \mathcal{K} \to \mathbb{R}_{\geq 0}$ is the function defining the objective cost incurred when applying action $a \in A(x)$ while in state $x \in \mathbf{X}$. We assume that these costs are non-negative.
4. $\mathcal{P}_{xy}^a$ is the one step transition probability from state x to state y when action a is applied, i.e., $\mathcal{P}_{xy}^a = \Pr[X_{t+1} = y | X_t = x, A_t = a]$.

The above model is *stationary* because costs and transition probabilities do not depend on time. Based on the above definitions, the sequence of states and actions over time constitutes a stochastic process that we denote as (X_t, A_t). Without loss of generality (since the model is stationary), we assume that the evolution starts at $t = 0$.

The optimal control of an MDP entails the determination of a closed-loop policy π defining which action should be applied in order to minimize an aggregate (sum) objective function of the objective costs. A policy π induces a mass distribution[1] over the realizations of the stochastic pro-

[1] Such mass distribution not only exists, but can be explicitly computed.

cess (X_t, A_t). Let Π_{D} be the set of closed-loop, Markovian, stationary, and *deterministic* policies $\pi : \mathbf{X} \to A$. It is well known that for MDPs there is no loss of optimality in restricting the attention to policies in Π_{D} (instead, e.g., of also considering history-dependent or randomized policies). On the other hand, as we will discuss later, randomization is needed for optimal policies for CMDPs.

Depending on the specific form of the objective function, MDP problems can be categorized into four main classes: finite horizon MDPs, infinite horizon MDPs with discounted cost, average-cost infinite horizon MDPs, and total cost MDPs (or optimal stopping MDPs), where the cost is on an infinite horizon but the state will eventually enter an absorbing set where no additional costs are incurred. Given their relevance to deployment problems, in this paper we focus on total cost MDPs. In the literature one finds three types of total cost MDPs: (i) the transient MDPs, for which the total expected time spent in each state is finite under any policy, (ii) the absorbing MDPs, for which the total expected "life time" of the system is finite under any policy, and (iii) contracting MDPs [18]. One can show that all three types of total cost MDPs are equivalent under the assumption of a finite state space [18]. Accordingly, in this paper we focus on *transient total cost MDPs*, with the understanding that our results apply also to the other two types of problems.

Transient total cost MDPs are defined as follows. Consider a partition of $\mathbf{X}$ into sets $\mathbf{X}'$ and $\mathbf{M}$, with $\mathbf{X} = \mathbf{X}' \cup \mathbf{M}$ and $\mathbf{X}' \cap \mathbf{M} = \emptyset$. A policy π is said to be transient in $\mathbf{X}'$ if[2]

1. $\sum_{t=0}^{\infty} \Pr_{x_0}^{\pi}[X_t = x] < \infty$ for every $x \in \mathbf{X}'$, and
2. $\mathcal{P}_{yx}^{a} = 0$ for each $y \in \mathbf{M}$, $x \in \mathbf{X}'$, and $a \in A(y)$.

In order words, the transient policy π ensures that, *eventually*, the state will enter the *absorbing* set $\mathbf{M}$. The second constraint on the transition probabilities implies that once the state enters $\mathbf{M}$ it will remain there. An MDP for which all policies are transient in $\mathbf{X}'$ is called a $\mathbf{X}'$-transient MDP.

We study in this paper the total cost criterion for $\mathbf{X}'$-transient MDPs. We assume throughout the paper that $c(x,a) = 0$ for any $x \in \mathbf{M}$, and that the initial state $x_0 \notin \mathbf{M}$. We define $\sum_{t=0}^{\infty} E_{\pi}[c(X_t, A_t)]$ as the total expected cost until the set $\mathbf{M}$ is reached [18], where the subscript π means that the expectation is with respect to the mass probability induced by the policy π. Note that this definition is well posed because the $\mathbf{X}'$-transient MDP assumption ensures that the expected total cost function exists and is bounded. A transient total cost MDP problem is then defined as follows:

> **Transient total cost MDP** — Given a $\mathbf{X}'$-transient MDP, determine a policy π^* minimizing the total expected cost, i.e., find
>
> $$\pi^* \in \arg\min_{\pi \in \Pi_{\mathrm{D}}} \sum_{t=0}^{\infty} E_{\pi}[c(X_t, A_t)].$$

Optimal policies can be computed in different ways, but, in practice, dynamic programming methods (value iteration or policy iteration) are the techniques most commonly used.

[2] $\Pr_{x_0}^{\pi}[X_t = x]$ is the probability that $X_t = x$ given the initial state $x_0 \in \mathbf{X}'$ and the policy π.

2.2 Total Cost Constrained Markov Decision Processes

A Constrained Markov Decision Process (CMDP) extends the MDP model by introducing additional costs and associated constraints. A CMDP is defined by $\mathbf{X}, A, c, \mathcal{P}, \{d_i\}_{i=1}^{L}, \{D_i\}_{i=1}^{L}$ where $\mathbf{X}, A, c, \mathcal{P}$ are the same as above and furthermore:

1. $d_i : \mathcal{K} \to \mathbb{R}_{\geq 0}$, with $1 \leq i \leq L$, is a family of L constraint costs incurred when applying action $a \in A(x)$ from state x. We assume that these costs are non-negative.
2. $D_i \in \mathbb{R}_{\geq 0}$ is an upper bound for the expected cumulative (through time) d_i costs.

Informally, solving a CMDP means determining a policy π minimizing the expected objective cost defined by c while ensuring that each of the constraint costs defined by the functions d_i are (in expectation) bounded by D_i. This notion can be formalized as follows. First, it is necessary to highlight that for CMDPs an optimal policy in general depends on the probabilistic distribution of the initial state. In the following this distribution is indicated with the letter β, with the understanding that $\beta(x) = \Pr[X_0 = x]$ for $x \in \mathbf{X}$.

The same conditions outlined in the previous subsection to define a transient total cost MDP can be used to define a transient total cost CMDP. A CMDP for which all policies are transient in $\mathbf{X}'$ is called a $\mathbf{X}'$-transient CMDP. As for the total cost MDP problem, we assume that $c(x,a) = 0$, but for the CMDP problem we further assume that $d_i(x,a) = 0$, $i \in \{1, \ldots, L\}$, for any $x \in \mathbf{M}$, and that $\beta(x) = 0$ for all $x \in \mathbf{M}$.

The total expected cost for an $\mathbf{X}'$-transient CMDP is defined as $\sum_{t=0}^{\infty} E_{\pi,\beta}[c(X_t, A_t)]$ where the expectation is taken with respect to the mass distribution induced by π and the initial distribution β. Similarly, we can define the total expected constraint costs as $E_{\pi,\beta}[d_i(X_t, A_t)]$. Note that because of the assumptions made about the costs in $\mathbf{M}$ and because, by assumption, the CMDP is $\mathbf{X}'$-transient, these expectations exist and are finite.

Henceforth, we will use the following notation: $c(\pi,\beta) := \sum_{t=0}^{\infty} E_{\pi,\beta}[c(X_t, A_t)]$, and $d_i(\pi,\beta) := \sum_{t=0}^{\infty} E_{\pi,\beta}[d_i(X_t, A_t)]$, $1 \leq i \leq L$. A transient total cost CMDP problem is then defined as follows:

> **Transient total cost CMDP** — Given a $\mathbf{X}'$-transient MDP, determine a policy π minimizing the total expected cost and satisfying the L constraints on the total expected constraint costs, i.e., find
>
> $$\pi^* \in \arg\min_{\pi \in \Pi_{\mathrm{M}}} c(\pi,\beta) \qquad (1)$$
> $$\text{s.t. } d_i(\pi,\beta) \leq D_i, \quad 1 \leq i \leq L,$$
>
> where Π_{M} is the set of closed-loop, Markovian, stationary, and *randomized* policies (i.e., mapping a state x into a probability mass function over $A(x)$).

It is well known (see, e.g., Theorem 2.1 in [18] and Theorem 6.2 in [21]) that there is no loss of optimality in restricting

the attention to policies in Π_M (instead, e.g., of also considering history-dependent policies). However, one should not restrict π to be in Π_D, as for the MDP case, because an optimal policy might require randomization. CMDPs do not share many of the properties enjoyed by MDPs (see [21] for a comprehensive discussion of the subject.) For example, CMDPs cannot be solved using dynamic programming but can be solved, for example, using a linear programming formulation presented below[3].

A fundamental theorem concerning CMDPs [18] relates the solution of the optimization problem defined in equation (1) to the following linear optimization problem. Let $\mathcal{K}' := \{(x,a), x \in \mathbf{X}', a \in A(x)\}$ and consider $|\mathcal{K}'|$ optimization variables $\rho(x,a)$, each one associated with an element in $\mathcal{K}'$. Let $\delta_x(y) = 1$ when $x = y$ and 0 otherwise, and define the linear optimization problem:

$$\begin{aligned} \min_{\rho} \quad & \sum_{(x,a)\in\mathcal{K}'} \rho(x,a)c(x,a) \qquad (2)\\ \text{s.t.} \quad & \sum_{(x,a)\in\mathcal{K}'} \rho(x,a)d_i(x,a) \le D_i \;\; 1 \le i \le L \\ & \sum_{y\in\mathbf{X}'} \sum_{a\in A(y)} \rho(y,a)(\delta_x(y) - \mathcal{P}^a_{yx}) = \beta(x) \;\forall x \in \mathbf{X}' \\ & \rho(x,a) \ge 0 \quad \forall (x,a) \in \mathcal{K}'. \end{aligned}$$

The optimization problem defined in equation (1) has a solution if and only if the problem defined in equation (2) is feasible [18], and the optimal solution to the linear program induces an optimal, stationary, randomized policy for the CMDP defined as follows:

$$\pi^*(x,a) = \frac{\rho(x,a)}{\sum_{a\in A(x)} \rho(x,a)} \;\; x \in \mathbf{X}', a \in A(x), \qquad (3)$$

where $\pi^*(x,a)$ is the probability of taking action a when in state x. If the denominator in equation (3) is zero, then the policy can be arbitrarily defined for (x,a). Note that equation (3) does not specify a policy for states in $\mathbf{M}$. Since no further costs are incurred in $\mathbf{M}$ and the state cannot leave $\mathbf{M}$ once it enters it, a policy can be arbitrarily defined for those states. The optimization variables $\rho(x,a)$ are referred to as *occupation measures* [18], since for each pair (x,a), they can be written as:

$$\rho(x,a) = \sum_{t=0}^{\infty} \Pr[X_t = x, A_t = a], \qquad (4)$$

where the probability is implicitly conditioned on a policy π and an initial distribution β. Note that an occupation measure is a sum of probabilities, but in general is *not* a probability itself.

[3]More in general, there exist more than one linear programming formulation that can be used, and methods based on Lagrange multipliers have been introduced as well. However, they will not be considered in this paper.

2.3 Robust Constrained Markov Decision Processes

In this section we introduce a natural generalization of the CMDP problem to the case where there is uncertainty in the model's parameters (this uncertainty should not be confused with the probabilistic uncertainty affecting the outcome of a control action). The standing assumption is that the underlying MDP is transient. Specifically, we consider the scenario where the stage-wise constraint costs are affected by *additive* uncertainty in the form:

$$d_\varepsilon(X_t,A_t) := d(X_t,A_t) + \varepsilon(X_t,A_t),\ \forall (X_t,A_t) \in \mathcal{K}',$$

where $d(X_t,A_t)$ represents the *nominal* stage-wise constraint cost, and $\varepsilon(X_t,A_t)$ is the uncertain term. Thus, for a *given* sequence of state-action pairs $\{X_t,A_t\}_t$, the *uncertain* constraint cost is defined as: $d_\varepsilon(\pi,\beta) := \sum_{t=0}^{\infty} E_{\pi,\beta}[d(X_t,A_t) + \varepsilon(X_t,A_t)]$.

Note that, according to our definition, the uncertainty is stage-invariant (i.e., it does not change from stage to stage, but of course it is a function of X_t and A_t). In practice it is often difficult to give a stochastic characterization of such uncertainty. On the other hand, in many cases it is still possible to characterize the support of the uncertainties. Specifically, we let $\mathcal{U} \subset \mathbb{R}^{|\mathcal{K}'|}$ denote the set of admissible values for the uncertainty. In this paper we restrict our attention to *budgeted interval uncertainty*:

Assumption 2.1 (Budgeted Interval Uncertainty). *Let $\{\bar{\varepsilon}(x,a)\}_{(x,a)\in\mathcal{K}'}$ be a given non-negative vector in $\mathbb{R}^{\mathcal{K}'}_{\ge 0}$. Then the uncertainty set is given by*

$$\mathcal{U} = \left\{ \varepsilon \in \mathbb{R}^{|\mathcal{K}'|}_{\ge 0} : \begin{array}{l} 0 \le \varepsilon(x,a) \le \bar{\varepsilon}(x,a), \forall (x,a) \in \mathcal{K}' \\ \sum_{(x,a)\in\mathcal{K}'} \varepsilon(x,a) \le \Gamma \end{array} \right\},$$

where Γ, $0 \le \Gamma \le \sum_{(x,a)\in\mathcal{K}'} \bar{\varepsilon}(x,a)$, is a given "uncertainty budget."

Note that the constraint costs remain non-negative under any perturbation in $\mathcal{U}$, hence the problem is well-posed under any realization of the underlying uncertainties, see Section 2.2. Within the Budgeted Interval Uncertainty model, $\varepsilon(x,a)$ represents the deviation from the nominal cost coefficient for a given $(x,a) \in \mathcal{K}'$. In turn, Γ has the interpretation of a *budget of uncertainty* that a system designer selects in order to easily trade robustness and performance. As an alternative interpretation, it is unlikely that all of the $d(x,a)$, $(x,a) \in \mathcal{K}'$, will deviate up to their worst case values, and the goal is to protect the system up to an uncertainty magnitude equal to Γ. Note a value of $\Gamma = 0$ implies that no uncertainty is expected, while a value $\Gamma = \sum_{(x,a)\in\mathcal{K}'} \bar{\varepsilon}(x,a)$ implies that the system designer expects that all parameters will deviate up to their maximum values (in which case the problem could be rewritten as a problem without uncertainty and with modified constraint stage costs $d(x,a) + \bar{\varepsilon}(x,a)$). This model of uncertainty is fairly common in robust optimization, we refer the reader to [22, 23]. Extensions to alternative uncertainty models are possible (e.g., multiplicative uncertainty), and are left for future research.

Consider, then, the *robust* constraint

$$\sup_{\varepsilon \in \mathcal{U}} d_\varepsilon(\pi, \beta) \leq D. \tag{5}$$

The next lemma (whose proof is provided in the Appendix) shows that we can replace the sup operator with the max operator.

Lemma 2.2. *For any policy π and initial distribution β, the supremum in the optimization problem $\sup_{\varepsilon \in \mathcal{U}} d_\varepsilon(\pi, \beta)$ is achieved. Hence one has $\sup_{\varepsilon \in \mathcal{U}} d_\varepsilon(\pi, \beta) = \max_{\varepsilon \in \mathcal{U}} d_\varepsilon(\pi, \beta)$.*

In light of equation (5) and Lemma 2.2, we can therefore formulate the following Robust Constrained Markov Decision Problem (RCMDP):

Optimization problem RCMDP — Given a $\mathbf{X}'$-transient CMDP, an initial distribution β, and a constraint threshold D, find

$$\pi^* \in \arg\min_{\pi \in \Pi_M} c(\pi, \beta) \tag{6}$$

$$\text{s.t.} \max_{\varepsilon \in \mathcal{U}} d_\varepsilon(\pi, \beta) \leq D, \tag{7}$$

over the class of policies $\pi \in \Pi_M$.

We next show how the RCMDP problem can be solved as a linear optimization problem on the space of occupation measures. Consider the following optimization problem:

Optimization problem $\mathcal{OPT}$ — For a given initial distribution β and risk threshold D, solve

$$\begin{aligned}
\min_{\rho} \quad & \sum_{(x,a) \in \mathcal{K}'} \rho(x,a) c(x,a) \\
\text{s.t.} \quad & \sum_{y \in \mathbf{X}'} \sum_{a \in A(y)} \rho(y,a) \left[\delta_x(y) - \mathcal{P}_{yx}^a \right] = \beta(x), \forall x \in \mathbf{X}' \\
& \max_{\varepsilon \in \mathcal{U}} \sum_{(x,a) \in \mathcal{K}'} \rho(x,a)(d(x,a) + \varepsilon(x,a)) \leq D \\
& \rho(x,a) \geq 0, \quad \forall (x,a) \in \mathcal{K}'.
\end{aligned}$$

Note that problem $\mathcal{OPT}$ is a *robust* linear programming problem. As for non-robust CMDPs, $\rho(x,u)$ has the meaning of occupation measure. The following theorem (whose proof is given in the Appendix) establishes the relation between RCMDP and $\mathcal{OPT}$.

Theorem 2.3. *The RCMDP problem has a solution if and only if problem $\mathcal{OPT}$ is feasible. The optimal solution to the robust linear program $\mathcal{OPT}$ induces an optimal stationary, randomized policy for RCMDP defined as follows:*

$$\pi^*(x,a) = \frac{\rho^*(x,a)}{\sum_{a \in A(x)} \rho^*(x,a)}, \ x \in \mathbf{X}', a \in A(x), \tag{8}$$

where $\{\rho^*(x,a)\}_{(x,a) \in \mathcal{K}'}$ is the optimal solution to problem $\mathcal{OPT}$. For states $x \in \mathbf{X}$ such that $\sum_{a \in A(x)} \rho^*(x,a) = 0$, $\pi^*(x,a)$ is arbitrarily chosen, for every $a \in A(x)$.

3 The Deployment Problem as a RCMDP

In this section we formalize the rapid deployment problem we wish to study and we show how it can be modeled as an instance of a RCMDP.

3.1 General Description

At a high level, the problem is as follows. A set of robots are placed within a bounded environment where a set of points represent target locations. The objective is to deploy the robots so that each location is reached by at least one robot and the deployment task takes no longer than a given temporal deadline. However, the environment is stochastic, in the sense that robots might "probabilistically" fail. In such a stochastic setup, the objective is then to maximize the *probability* that each location is reached by at least one robot while ensuring that the *expected* duration of the deployment task is upper bounded by a given temporal deadline. In this paper we interpret the *duration* of the deployment task as the elapsed time between the common instant when robots start moving and the instant when the *last* robot stops moving, either because of a failure *or* because a target has been reached.

Specifically, following the approach presented in [10], the environment is abstracted into an undirected graph $G = (X, E)$ where X is the set of vertices and E is the set of edges. (The vertex set X will be later mapped into the state space $\mathbf{X}$ in the ensuing RCMDP model.) An edge $e \in E$ between two vertices v_1 and v_2 means that a robot can move from v_1 to v_2 and vice versa. We consider that there are K robots, initially located at a single vertex $v_0 \in X$, that are required to reach a set of target vertices, denoted as $T \subset X$. Self-loops in G are only allowed for vertices in the target set, i.e., $(v,v) \in E$ if and only if $v \in T$. The deployment task is considered successful if (i) each vertex in T is reached by at least one robot, and (ii) this is accomplished within a given temporal deadline D.

To capture the time/safety trade-off, we associate to each edge $e_i \in E$, $i \in |E|$, a *safety function* $S_e : \mathbb{R}_{\geq 0} \to [0,1]$. Function $S_e(t)$, for each t, represents the probability of successfully traversing an edge e given that the traversal time is equal to t. The safety function satisfies the boundary constraints $S_e(0) = 0$ and $\lim_{t \to +\infty} S_e(t) = 1$, and must be non-decreasing. Other than those constraints, the safety function is arbitrary (and potentially different for each edge). An example is provided in Figure 1, where the probability of success is modeled using a sigmoidal function. There are many other success probability models available in the survival analysis community. For example, in [24], the hazard probability functions are modeled either with Exponential, Gamma and Weibull distributions or with other non-parametric estimates. However the advantages of modeling the success probability with a sigmoidal/logit function are, (i) this function serves as a good model for binary classification, and (ii) the model parameters can be easily estimated by logistic regression, after experimental data is given.

Accordingly, the deployment problem entails finding control policies for each robot that prescribe at each vertex (i) which vertex to visit next, and (ii) at what speed the

edge should be traversed. Robots should move slowly to increase their chances of successfully traversing the edges of the graph, however they can not arbitrarily slow down because they will otherwise miss the temporal deadline. In this paper we consider robotic swarms, hence we seek *minimalistic* control strategies that do not involve any communication after the deployment process is initiated.

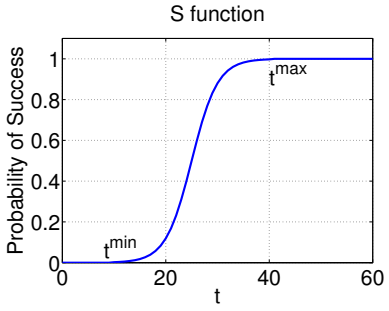

Fig. 1: A sigmoidal shape for the safety function S_e associated with the edges in the graph.

3.2 Formulation as RCMDP — Single-Robot

We now show how to cast the aforementioned deployment problem within the RCMDP model.

Consider, first, the single-robot case and, hence, a target set that comprises a single vertex (i.e., $|T| = 1$). The state space comprises all the vertices of the graph G plus a virtual *sink state* $\mathcal{S}$ modeling failures, i.e., the robot enters the sink state when it experiences an irremediable failure while traversing an edge and then stops functioning. The state space is then $\mathbf{X} := X \cup \{\mathcal{S}\}$. The initial distribution β is defined as $\beta(v_0) = 1$ and $\beta(x) = 0$ for $x \in X, x \neq v_0$, since, in our model, v_0 is (deterministically) the initial location of the robot. At each vertex $y \in X$, the robot needs to decide (i) what vertex to visit next, and (ii) the traversal time (or, equivalently, the traversal speed), within given bounds (dictated by velocity bounds). The traversal time is a continuous variable, which is discretized with a time step $\Delta > 0$ to make the model amenable to dynamic optimization. (For simplicity, we assume that the discretization step Δ is equal for all stages.) Summarizing, the action set at a vertex $y \in X, y \notin T$, is given by:

$$A(y) = \Big\{(x,t) \in X \times \mathbb{R}_{\geq 0} : (y,x) \in E,\ t = t_{yx}^{\min} + k\Delta,\ k \in \mathbb{N}, 0 \leq k \leq \Big\lfloor \frac{t_{yx}^{\max} - t_{yx}^{\min}}{\Delta} \Big\rfloor \Big\},$$

where $t_{yx}^{\min} \geq 0$ and $t_{yx}^{\max} > 0$ are the minimum and maximum, respectively, traversal times for the edge $(y,x) \in E$ (see Figure 1). An action $(x,t) \in A(y)$ means that the robot decides to navigate from y to x spending time t. For the target vertex, i.e., $y \in T$, we consider a single action:

$$A(y) = \Big\{(x,t) \in X \times \mathbb{R}_{\geq 0} : x = y,\ t = t_{yy}\Big\},$$

where $t_{yy} > 0$ models the self-loop time (the choice of this value is immaterial, since T will be the absorbing set in the RCMDP model). For the special sink state $\mathcal{S}$ we define just one action, denoted as $a_{\mathcal{S}}$, which lets the state of the robot transition to the unique vertex in the target set T (this choice is made to ensure that T is the absorbing set in the ensuing RCMDP model).

The above action sets and the fact that the traversal of each edge in E can only be accomplished probabilistically (according to the safety function S_e) induce the following transition probabilities on $\mathbf{X}$:

$$\mathcal{P}_{yx}^{a} = \begin{cases} S_{(y,x)}(t) & \text{if } y \neq \mathcal{S}, a = (x,t) \in A(y), x \neq \mathcal{S}, \\ 1 - S_{(y,x)}(t) & \text{if } y \neq \mathcal{S}, a = (x,t) \in A(y), x = \mathcal{S}, \\ 1 & \text{if } y = \mathcal{S}, a = a_{\mathcal{S}}, x \in T. \end{cases}$$

All other transition probabilities are equal to zero. The first case models the probability of successfully completing a traversal from y to x, where the success probability is dictated by the safety function $S_{(y,x)}$. The second case models the failure probability, i.e., when a robot does not successfully perform the traversal from y to x and, hence, it enters the sink state $\mathcal{S}$. The third case enforces the deterministic transition from a sink state to the target vertex (which, as discussed before, ensures the T is the absorbing set). It is immediate to conclude that, with these transition probabilities, any Markovian, stationary, and randomized policy on $\mathbf{X}$ is transient on the set $\{\mathbf{X} \setminus T\} := \mathbf{X}'$; the corresponding absorbing set is $\mathbf{M} := T$. Figure 2 illustrates the relationship between $\mathbf{X}$, $\mathcal{S}$, and T.

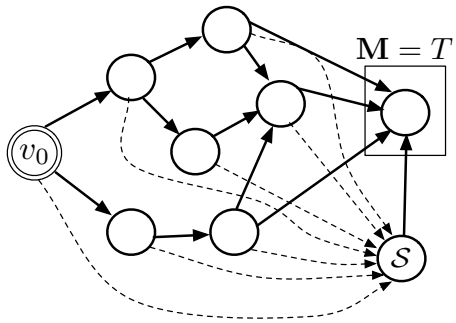

Fig. 2: Given a graph $G = (X,E)$ and a policy π, multiple stochastic paths from the deployment vertex v_0 to the target vertex set T exist. Whenever a failure occurs, the state enters $\mathcal{S}$ (dashed arrows). States outside the box labeled $\mathbf{M}$ are in $\mathbf{X}'$.

To complete the definition of the RCMDP model, we need to define the objective costs $c(x,a)$ and the constraint

costs $d(x,a)$. Given that our stated objective is to maximize the probability of success (or equivalently, minimize the probability of failure), the objective costs are defined as follows:

$$c(y,a) = \begin{cases} 0 & \text{if } y \neq \mathcal{S}, a = (x,t) \in A(y), \\ 1 & \text{if } y = \mathcal{S},\ a = a_{\mathcal{S}}. \end{cases}$$

Note that, with this choice, $\sum_{t=0}^{\infty} E_{\pi,\beta}\left[c(X_t,A_t)\right]$ is exactly the probability that the robot fails to reach the target. To see this, denote the sample space by Ω, define the events $B_t = \{X_t = \mathcal{S}\}$, for $t = 0,1,\ldots$, and let $B := \Omega \setminus \cup_0^{\infty} B_t$. Clearly, the events $\{B_t\}_{t=0}^{\infty}$ and B are collectively exhaustive, i.e., their union yields Ω. Also, there is only one action available when the system is in $\mathcal{S}$, and such action deterministically moves the state to the absorbing set $\mathbf{M}$ (where the state remains trapped). Hence, the events $\{B_t\}_{t=0}^{\infty}$ and, of course, B, are mutually exclusive. We can then apply the law of total probability and write (where we omit the subscripts π and β for simplicity)

$$\begin{aligned} \Pr[\text{Failure}] &= \sum_{t=0}^{\infty} \underbrace{\Pr[F|B_t]}_{=1} \Pr[B_t] + \underbrace{\Pr[F|B]}_{=0} \Pr[B] \\ &= \sum_{t=0}^{\infty} \Pr[X_t = \mathcal{S}] = \sum_{t=0}^{\infty} \Pr[X_t = \mathcal{S}, A_t = a_{\mathcal{S}}] \\ &= \sum_{t=0}^{\infty} E\left[c(X_t,A_t)\right], \end{aligned} \tag{9}$$

and the claim follows. Also, the above derivation highlights that $\sum_{t=0}^{\infty} E_{\pi,\beta}\left[c(X_t,A_t)\right] = \rho(\mathcal{S},a_{\mathcal{S}})$, hence the cost function is indeed equal to the occupation measure for the state-action pair $(\mathcal{S},a_{\mathcal{S}})$.

Similarly, the constraint costs are defined as

$$d(y,a) = \begin{cases} t & \text{if } y \in X \setminus T, a = (x,t) \in A(y), \\ 0 & \text{if } y \in T,\ a = (x,t) \in A(y), \\ 0 & \text{if } y = \mathcal{S},\ a = a_{\mathcal{S}}. \end{cases}$$

Note that, for simplicity, travel times are assumed deterministic, but our framework can be easily extended to the case where travel times are stochastic. With these definitions both the objective and the constraint costs evaluate to zero on the absorbing set $\mathbf{M} = T$ (in particular, $d(y,a)$ for $y \in T$ is set to zero rather than to t_{yy}).

In this paper we assume that the constraint costs (i.e., the travel times) are not exactly known (as it is oftentimes the case in practical scenarios), and they are instead characterized through a budgeted interval uncertainty model (see Assumption 2.1). Conceptually, the traversal time t selected by a robot should be interpreted as an *intended* travel time: the (uncertain) constraint costs then translate intended travel times into *actual* travel times, whereas safety functions translate intended travel times into risk of failure (the idea being that, for example, shorter intended travel times correspond to more dangerous maneuvers). Note that within our model the transition probabilities are *certain* (the uncertainty in the mapping from intended travel times to risk is encapsulated in the safety function S_e). One could also consider formulations where the transition probabilities and the objective costs are uncertain as well: this, however, would significant complicate the model and is left for future research.

The single-robot deployment problem is then reformulated as a RCMDP problem: find a Markovian policy that minimizes the probability of failure (i.e., the summation of the expected objectives costs c) while *robustly* keeping the traversal time (i.e., the summation of the expected constraint costs d) below a given temporal deadline D. The robustness of the formulation stems from the fact that we consider traversal times (i.e., the functions $d(y,a)$) that are uncertain.

This formulation is consistent with our definition of task duration as discussed in Section 3.1. Other formulations are possible, for example one might be interested in minimizing the expected deployment duration while constraining the probability of failure below a given threshold (possibly with uncertain safety functions), or might desire to constrain the duration of the deployment task only for those executions that do not involve failures. Such formulations are left for future research. We mention, however, that in many cases the latter formulation and our formulation are essentially equivalent. This can be seen by observing that $E[\text{task duration}|\text{success}]\Pr(\text{success}) \leq E[\text{task duration}] \leq E[\text{task duration}|\text{success}]$, where the first inequality is a consequence of the law of total expectation and the second inequality follows from the fact that in case of failure the state enters T through a *shortcut* via $\mathcal{S}$. Assuming that $\Pr(\text{success})$ (which can be computed exactly in our formulation) is high, say, 90%, the discrepancy between the two formulations will then be small. We investigate this aspect further in Section 7.

3.3 Formulation as RCMDP – Multi-Robot

The multi-robot formulation is indeed a simple extension of the single-robot formulation. The main tenet in our approach is that we seek minimalistic control strategies that do not involve any communication once the robotic swarm is deployed (which is oftentimes a requirement for the deployment of robotic swarms comprising simple and inexpensive platforms). We note that the proposed model can also serve as a yardstick for comparison with more sophisticated coordination mechanisms, by providing easily-computable bounds for the achievable safety-speed Pareto curves. Our key technical assumption is that congestion effects (i.e., robots colliding into each other) are negligible. In other words, we assume that failures are *statistically independent*. This is normally the case in the fast growing domain of robotic swarms comprising minimalistic, palm-sized micro-aerial vehicles (MAVs) [25–28].

Consider $K \geq |T|$ robots (if $K < |T|$, clearly, the deployment objective as formulated before can not be accomplished). Our formulation of the multi-robot deployment

problem essentially decouples the target assignment problem from the path planning problem, specifically:

1. each robot is assigned a deployment location in T;
2. each robot executes a single-robot deployment policy (by solving the associate RCMDP) to reach the assigned location.

Such formulation fulfills the requirement that robots do not need to communicate during the deployment process (e.g., to learn how many robots are in the team, or to communicate that a certain target vertex has been already reached, or to update the safety functions S_e). The problem, then, is how to assign targets to robots. Let $\alpha := (\alpha_1, \ldots\ldots, \alpha_K) \in T^K$ denote a target assignment, with the understanding that $\alpha_i = v \in T$ implies that the ith robot is assigned to target v, $i = \{1, \ldots, K\}$. Since each robot executes a single-robot deployment policy (in a statistically independent fashion, by assumption), the probability that the overall deployment is successful, given an assignment α, can be easily computed from the probabilities that each robot can successfully reach its assigned target. Specifically, let $\varphi(\alpha)$ denote the probability that the deployment task is successful for a given assignment α, and let $\mathrm{PF}(v)$, for $v \in T$, denote the probability that a robot assigned to target v fails to reach such target (these probabilities are simply the optimal costs of the corresponding RCMDPs). The probability $\varphi(\alpha)$ is then given by

$$\varphi(\alpha) = \prod_{j=1}^{|T|} \Big(1 - \mathrm{PF}(v_j)^{|\{\alpha_i \in \alpha:\, \alpha_i = v_j\}|}\Big),$$

where v_j is the jth target in T. Accordingly, the target assignment problem becomes:

Target Assignment (TA) — For $K \geq |T|$, solve

$$\begin{aligned} \max_{k_j} \quad & \prod_{j=1}^{|T|} \Big(1 - \mathrm{PF}(v_j)^{k_j+1}\Big) \\ \text{s.t.} \quad & \sum_{j=1}^{|T|} k_j = K - |T| \\ & k_j \geq 0,\ k_j \in \mathbb{N},\ j \in \{1, \ldots, |T|\}. \end{aligned}$$

The formulation assigns target $v_1 \in T$ to the robots 1 to $k_1 + 1$, target v_2 to robots $k_1 + 2$ to $k_1 + k_2 + 2$ and so on, in order to minimize the global success probability. The rapid multi-robot deployment problem is then formulated as follows (assuming $K \geq |T|$):

1. For each target in $v \in T$, solve the RCMDP version of the single-robot deployment problem with target set equal to v (note that the optimal cost of the RCMDP is exactly the probability $\mathrm{PF}(v)$).
2. Solve the target assignment problem TA and assign targets to robots accordingly.
3. Let each robot execute the single-robot optimal deployment policy to reach its assigned target.

This formulation requires computationally-efficient solutions for RCMDP and TA, which are discussed next.

4 Efficient Solution for RCMDP

As shown in the proof of Theorem 2.3 (provided in the Appendix), the RCMDP problem can be solved as a robust linear optimization problem, and in particular as a linear optimization problem with a number of constraints equal to the number of vertices of the uncertainty set $\mathcal{U}$. Since this number is, possibly, exponential[4] in the number of states and actions, it becomes critical to find an efficient algorithm for the solution of problem $\mathcal{OPT}$. We next show how problem $\mathcal{OPT}$ can be transformed into a linear programming problem with a number of constraints and variables that is *linear* in $|\mathcal{K}'|$ for *all* choices of the uncertainty budget Γ (the strategy is to introduce an auxiliary set of variables whose cardinality is $|\mathcal{K}'| + 1$). Consider the following optimization problem:

Optimization problem $\mathcal{OPT}_2$ — Given an initial distribution β and a risk threshold D, solve

$$\begin{aligned} \min_{\rho, \lambda, \mu} \quad & \sum_{(x,a) \in \mathcal{K}'} \rho(x,a) c(x,a) \\ \text{s.t.} \quad & \sum_{y \in \mathbf{X}'} \sum_{a \in A(y)} \rho(y,a) \left[\delta_x(y) - \mathcal{P}^a_{yx}\right] = \beta(x), \forall x \in \mathbf{X}' \\ & \sum_{(x,a) \in \mathcal{K}'} \rho(x,a) d(x,a) + \bar{\varepsilon}(x,a) \lambda(x,a) + \mu\Gamma \leq D \\ & \lambda(x,a) + \mu \geq \rho(x,a), \quad \forall (x,a) \in \mathcal{K}' \\ & \rho(x,a), \lambda(x,a) \geq 0,\ \forall (x,a) \in \mathcal{K}' \\ & \mu \geq 0. \end{aligned}$$

Note that optimization problem $\mathcal{OPT}_2$ involves $2|\mathcal{K}'| + 1$ decision variables (μ is just a scalar).

Theorem 4.1. *Let $\{\rho^*(x,a)\}_{(x,a) \in \mathcal{K}'}$ be part of the optimal solution of problem $\mathcal{OPT}_2$. Then, $\{\rho^*(x,a)\}_{(x,a) \in \mathcal{K}'}$ is an optimal solution to problem $\mathcal{OPT}$.*

Proof. For any fixed $\rho := \{\rho(x,a)\}_{(x,a) \in \mathcal{K}'}$, and $D \in \mathbb{R}_{\geq 0}$, consider the constraint:

$$\max_{\varepsilon \in \mathcal{U}} \sum_{(x,a) \in \mathcal{K}'} \rho(x,a)(d(x,a) + \varepsilon(x,a)) \leq D, \qquad (10)$$

which can be written equivalently as

$$\max_{\varepsilon \in \mathcal{U}} \sum_{(x,a) \in \mathcal{K}'} \rho(x,a)\varepsilon(x,a) \leq D - \sum_{(x,a) \in \mathcal{K}'} \rho(x,a) d(x,a).$$

The optimization problem on the left-hand side of the above

[4] The number of vertices indeed depends on the value of Γ. In the extreme case where $\Gamma = 0$ the uncertainty set $\mathcal{U}$ has only *one* vertex.

equation can be written explicitly as:

$$\begin{aligned} p^*(\rho) = \max_{\varepsilon} \quad & \sum_{(x,a)\in\mathcal{K}'} \rho(x,a)\varepsilon(x,a) \\ \text{s.t.} \quad & 0 \le \varepsilon(x,a) \le \overline{\varepsilon}(x,a), \quad \forall (x,a) \in \mathcal{K}' \\ & \sum_{(x,a)\in\mathcal{K}'} \varepsilon(x,a) \le \Gamma. \end{aligned}$$

Consider the dual formulation of the above linear programming problem:

$$\begin{aligned} d^*(\rho) = \min_{\lambda,\mu} \quad & \sum_{(x,a)\in\mathcal{K}'} \overline{\varepsilon}(x,a)\lambda(x,a) + \mu\Gamma \\ \text{s.t.} \quad & \lambda(x,a) + \mu \ge \rho(x,a), \quad \forall (x,a) \in \mathcal{K}' \\ & \lambda(x,a) \ge 0, \quad \forall (x,a) \in \mathcal{K}' \\ & \mu \ge 0. \end{aligned}$$

By strong duality in linear programming, the primal optimal cost is equal to the dual optimal cost. Thus, one has $p^*(\rho) = d^*(\rho)$. The constraint in expression (10) can then be equivalently written as

$$\begin{aligned} & \exists \lambda, \mu : \\ & \sum_{(x,a)\in\mathcal{K}'} \rho(x,a)d(x,a) + \overline{\varepsilon}(x,a)\lambda(x,a) + \mu\Gamma \le D \\ & \lambda(x,a) + \mu \ge \rho(x,a), \quad \forall (x,a) \in \mathcal{K}' \\ & \lambda(x,a) \ge 0, \quad \forall (x,a) \in \mathcal{K}' \\ & \mu \ge 0. \end{aligned} \tag{11}$$

This implies that the robust constraint in problem $\mathcal{OPT}$ can be replaced by the set of linear constraints in expression (11). The claim then follows.

Since the number of constraints in the dual formulation is always $O(|\mathcal{K}'|)$, the computational complexity of problem $\mathcal{OPT}_2$ (assuming one uses an interior point algorithm for its solution) is linear in the number of state-action pairs [29], for *all* choices of the uncertainty budget Γ.

5 Efficient Solution of the Target Assignment Problem

In this section we show how to efficiently solve the TA problem introduced in Section 3.3. In the following, we assume that $\mathrm{PF}(v_j) \in (0,1)$ for all j. This is without loss of generality. In fact, if $\mathrm{PF}(v_j) = 1$ for some j, the deployment problem is infeasible (since it is impossible to reach target j no matter how many robots are destined there), while if $\mathrm{PF}(v_j) = 0$ for some j, then one should just send one robot to target j and consider the reduced problem without target j.

Consider, first, the case where $K \ge |T|$ and $K < 2|T|$. This is the case where the number of robots is on the order of the number of target locations. In practice, the number of elements in T is usually no larger than a small constant, say, 50, and in this case problem TA can be efficiently solved by using branch and bound techniques.

Let us consider the more challenging case where $K \ge 2|T|$ (the usual case for robotic swarms). In the remainder of this section we present a polynomial-time, asymptotically-optimal algorithm for the solution of problem TA. To this purpose, consider the following relaxed version of problem TA:

Relaxed Target Assignment (RTA) — For $K \ge 2|T|$, solve

$$\begin{aligned} \max_{k_j} \quad & \sum_{j=1}^{|T|} \log\left(1 - \mathrm{PF}(v_j)^{k_j+1}\right) \\ \text{s.t.} \quad & \sum_{j=1}^{|T|} k_j = K - 2|T|, \end{aligned}$$

where we have taken the log of the objective function, we have relaxed the integrality and non-negativity constraints, and we have reduced the right hand side of the equality constraint to $K - 2|T|$. Note that problem RTA is a concave maximization problem with a linear equality constraint. The first order sufficient optimality condition reads as [30]:

$$\lambda^* = \frac{\mathrm{PF}(v_j)^{k_j^*+1}\log(\mathrm{PF}(v_j))}{1 - \mathrm{PF}(v_j)^{k_j^*+1}}, \text{ for all } j \in \{1,\dots,|T|\}, \tag{12}$$

which implies:

$$k_j^* = \log\left(\frac{\lambda^*}{\lambda^* + \log(\mathrm{PF}(v_j))}\right)\frac{1}{\log(\mathrm{PF}(v_j))} - 1,$$

for all $j \in \{1,\dots,|T|\}$. Note that since $\mathrm{PF}(v_j) \in (0,1)$, equation (12) implies $\lambda^* \le 0$. The Lagrangian multiplier is found by solving the primal feasibility equation, that is:

$$\sum_{j=1}^{|T|} \log\left(\frac{\lambda^*}{\lambda^* + \log(\mathrm{PF}(v_j))}\right)\frac{1}{\log(\mathrm{PF}(v_j))} = K - 2|T|,$$

which can be readily solved, for example, by using the bisection method.

Consider the approximation algorithm for problem TA in Algorithm 1. We next prove that the above approximation algorithm is asymptotically optimal, that is it provides a feasible solution for problem TA whose cost converges to the optimal cost as $K \to +\infty$.

Theorem 5.1. *The approximation algorithm for problem TA:*

1. *delivers a* feasible *solution for all values of* $K \ge 2|T|$,
2. *is asymptotically optimal.*

Algorithm 1 Approximation algorithm for problem TA

1 Solve problem RTA and obtain λ^* and k_j^* for all $j \in \{1,\dots,|T|\}$
2 Pick (arbitrarily) a set of *non-negative* integers $r_1, r_2, \dots, r_{|T|}$ such that $\sum_{j=1}^{|T|} r_j = K - |T| - \sum_{j=1}^{|T|} \lceil k_j^* \rceil$ (note that $\sum_{j=1}^{|T|} \lceil k_j^* \rceil \leq K - |T|$, so this is always possible)
3 $\bar{k}_j \leftarrow \lceil k_j^* \rceil + r_j$
4 Return $\{\bar{k}_j\}_j$ as approximate solution for problem TA

Proof. First, note that the objective function in problem RTA acts as a barrier function, hence its optimal solution, $\{k_j^*\}_j$, satisfies the condition $k_j^* > -1$ for all $j \in \{1,\dots,|T|\}$. This implies that the solution provided by the approximation algorithm, $\{\bar{k}_j\}$, is a feasible solution for problem TA, in fact:

1. $\bar{k}_j \geq 0$ (since $k_j^* > -1$),
2. $\sum_{j=1}^{|T|} \bar{k}_j = K - |T|$ (by construction).

This proves the first part of the claim. To prove the second part of the claim, consider the approximation error, e, in terms of the log of the objective function of problem TA. The approximation error can be bounded as:

$$
\begin{aligned}
|e| &= \left| \sum_{j=1}^{|T|} \log\left(1 - \mathrm{PF}(v_j)^{k_j^{**}+1}\right) - \sum_{j=1}^{|T|} \log\left(1 - \mathrm{PF}(v_j)^{\bar{k}_j+1}\right) \right| \\
&\leq \sum_{j=1}^{|T|} \left| \log \frac{\left(1 - \mathrm{PF}(v_j)^{k_j^{**}+1}\right)}{\left(1 - \mathrm{PF}(v_j)^{\bar{k}_j+1}\right)} \right|,
\end{aligned}
\tag{13}
$$

where $\{k_j^{**}\}_j$ is the optimal solution to problem TA (which should not be confused with the optimal solution to problem RTA, that is, $\{k_j^*\}_j$). Hence, one can bound e according to

$$
e \leq T \max_j \left| \log \frac{\left(1 - \mathrm{PF}(v_j)^{k_j^{**}+1}\right)}{\left(1 - \mathrm{PF}(v_j)^{\lceil k_j^* \rceil + r_j + 1}\right)} \right|.
$$

Note that *all* $\{k_j^{**}\}_j$'s and $\{k_j^*\}_j$'s tend to $+\infty$ as $K \to \infty$ (this can be easily shown by noticing that if some of the k_j variables stay uniformly bounded as $K \to +\infty$, for K large enough one obtains a suboptimal solution — this exploits the assumption that $\mathrm{PF}(v_j) \in (0,1)$). Hence, as $K \to +\infty$, one obtains $|e| \to 0$.

In summary, when $K \leq 2|T|$, we directly solve the target assignment problem using branch and bound. When $K \geq 2|T|$ the solution can instead be (approximately) found by solving a convex problem.

6 Solution Algorithm for Rapid Swarm Deployment

Collecting the results so far, we can now present a computationally-efficient algorithm for the solution of the rapid multi-robot deployment problem. The pseudocode is presented in Algorithm 2.

Algorithm 2 Deployment Algorithm

1 $\alpha \leftarrow$ solution (possibly approximate) to target assignment problem TA
2 Assign target α_k to each robot $k \in \{1,\dots,K\}$
3 **for each** k **do**
4 Build RCMDP instance with $\mathbf{M} = \{v_{\alpha_k}\}$
5 $(\rho^*, \lambda^*, \mu^*) \leftarrow$ Solve Problem $\mathcal{OPT}_2$
6 **for each** $x \in \mathbf{X}', a \in A(x)$ **do**
7 **if** $\sum_{a \in A(x)} \rho^*(x,a) > 0$ **then**
8 $\pi^*(x,a) \leftarrow \rho^*(x,a) / \left(\sum_{a \in A(x)} \rho^*(x,a)\right)$
9 **else**
10 Choose an arbitrary value for $\pi^*(x,a)$
11 **end if**
12 **end for**
13 Navigate to v_{α_k} according to policy π^*
14 **end for**

In case no a-priori coordination is possible, one should consider as target assignment strategy a random uniform choice of the targets (made in a distributed fashion by the robots).

7 Numerical Experiments and Discussion

We present three sets of numerical experiments. In the first set of experiments we focus on the single-robot deployment problem, for a fixed uncertainty budget. The objective is to experimentally verify the correctness of the results obtained for RCMDP problem. In the second set of experiments we investigate the sensitivity of the solution to the single-robot deployment problem with respect to different values of the uncertainty budget (in other words, we investigate how a system designer can trade robustness and performance). In the third set of experiments, we consider the swarm deployment problem, and we discuss how the success probability scales with the number of robots for both the random target assignment algorithm (which requires no coordination among the robots), and the optimized target assignment algorithm (which requires a priori coordination among the robots).

For all numerical experiments we consider a map that is obtained from the publicly available *Radish* dataset (see Figure 3), where the traversal graph used for the simulations is overlaid onto the blueprint of the environment. For lack of space we present results concerning this map only, but our results are representative for a wide range of environments. Target vertices are marked with a number and the initial vertex is indicated with a pink triangle (vertex number 1). Each edge is characterized by a different safety function S_e. In our experiments we considered safety functions with a sigmoidal shape (as in Figure 1). The parameters $t_{\min}$ and $t_{\max}$ (as defined in Figure 1) were randomly selected (in a way, of

course, that $t_{\min} < t_{\max}$) and are in general different for each edge. This particular choice of the safety function S_e is meant to be just an example, and in practice one can consider any safety function as long as it obeys the constraints outlined in Section 3. In all experiments we assume that the constraint cost uncertainty is upper bounded by $\bar{\varepsilon}(x,a) = 0.5\,d(x,a)$ for all $(x,a) \in \mathcal{K}'$.

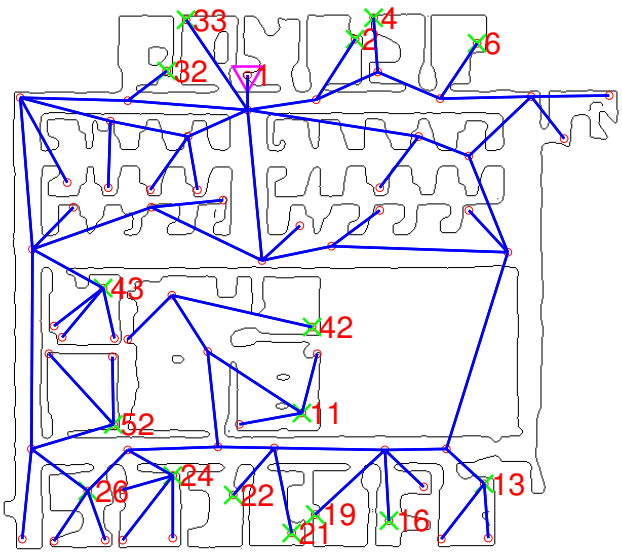

Fig. 3: The map used to experimentally evaluate the deployment policies is the same as the one used in [10]. The deployment vertex is marked with a pink triangle, whereas goal vertices are indicated by green crosses. Edges between vertices indicate that a path exists.

7.1 Single-Robot Deployment Problem

In this section we numerically investigate the rapid single-robot deployment problem along two dimensions: convergence of the empirical success probability to its theoretical value (equal to $\rho(\mathcal{S}, a_{\mathcal{S}})$), and fulfillment of temporal constraint *in expectation*. Collectively, these experiments are aimed at showing the correctness of our approach (i.e., of Algorithm 2) for the single-robot case. For all experiments in this section, we consider an uncertainty budget

$$\Gamma = 1 \times \sum_{(x,a)\in\mathcal{K}'} \bar{\varepsilon}(x,a),$$

and we assume that vertex 13 is the target vertex (a parametric study for different values of Γ is presented in Section 7.2).

To study the convergence of the empirical success probability with respect to the number of Monte Carlo trials, we consider a temporal deadline $D = 237$. The convergence error is defined as:

$$\text{error} := \frac{|\text{empirical PF} - \text{theoretical PF}|}{\text{theoretical PF}}.$$

Alternatively, one can consider the binary random variable $\mathcal{M} \in \{\text{success}, \text{fail}\}$, and then study the Kullback-Leibler (KL) divergence for the probability mass function associated with $\mathcal{M}$, that is:

$$D_{\text{KL}} = \ln\left(\frac{P_{\mathcal{M}}}{1-\rho(\mathcal{S},a_{\mathcal{S}})}\right) P_{\mathcal{M}} + \ln\left(\frac{1-P_{\mathcal{M}}}{\rho(\mathcal{S},a_{\mathcal{S}})}\right)(1-P_{\mathcal{M}}),$$

where $P_{\mathcal{M}}$ is the empirical success probability and $1-\rho(\mathcal{S},a_{\mathcal{S}})$ is the theoretical success probability. The results are presented in Table 1. One can note that (i) the empirical success probability converges to the theoretical success probability, as expected, and (ii) with 100-1,000 Monte Carlo trials convergence is already satisfactory. Accordingly, for the numerical experiments in the remainder of this section and for those in Section 7.2 we will use a number of Monte Carlo trials equal to 1,000, while for the numerical experiments in Section 7.3 we will use a number of Monte Carlo trials equal to 100.

	MC = 100	MC = 1,000	MC = 5,000	MC 10,000
Error	4.82%	1.37%	0.57%	0.46%
D_{KL}	0.0146	0.0011	0.0002	0.0001

Table 1: Convergence analysis for empirical success probability (MC stands for number of Monte Carlo trials).

We next study how the temporal constraint is fulfilled *in expectation*. We consider different values for the temporal deadlines, namely $D = \{175, 237, 299\}$. Results are presented in Table 2. The first and second rows of the table report, respectively, the theoretical and the empirical success probabilities, which, as expected, increase as the temporal threshold is increased. The third row reports the values for the constraint costs, which are always lower (in expectation) than their corresponding thresholds. The fourth row reports standard deviations. One can note that the standard deviations are rather large: this is due to the fact that, in our formulation, the duration of a deployment task is given by the elapsed time between the instant when the robot starts moving and the instant when it stops moving, respectively. Such duration has a large spread (since failures might induce *early* termination), and the standard deviation is consequently quite large. In the fifth row, we report the expected time to successfully complete the deployment task, conditioned on incurring no failures. For $D = 175$, such time is slightly higher than the threshold. This mismatch is explained by the relatively high failure probability shown in the first row of the table. On the contrary, for higher values of D the probability of failure is lower and the temporal deadline is met. These findings confirm our discussion in Section

	$D = 175$	$D = 237$	$D = 299$
Theoretical success prob.	0.5356	0.7321	0.9172
Empirical success prob.	0.5350	0.7370	0.9140
Expectation	115.2550	158.5350	199.7270
Standard dev.	94.8038	72.5002	34.0615
Expectation (success)	200	199.8225	207.9409
Standard dev. (success)	0	0.5441	1.3213

Table 2: Probability of success and fulfillment of temporal constraint for single-robot deployment.

3.2 about the relation between our formulation and a formulation where one constrains the duration of the deployment task *assuming no failures*. Finally, in the last row of the table one can notice very low values for the standard deviations (again, conditioned on no failures): this is due to two facts (i) assuming no failures, the only source of randomness is the randomization of the control policies, and (ii) according to [18, Theorem 3.8], with one constraint (since there is no uncertainty) an optimal policy only uses *one* randomization.

7.2 Sensitivity with Respect to Budget Uncertainty

In this section we study the sensitivity of the solution to the single-robot deployment problem with respect to different values of the uncertainty budget. Specifically, we parameterize the uncertainty budget as:

$$\Gamma = \gamma \times \sum_{(x,a)\in\mathcal{K}'} \bar{\varepsilon}(x,a),$$

where γ, referred to as "factor of uncertainty", takes the values shown in the first column of Table 3. We assume, as before, that vertex 13 is the target vertex and we consider a constraint threshold $D = 237$. The results are reported in Table 3. One can observe that, for this example, the success probability decreases by about 25% when going from the nominal model $\gamma = 0$ to the worst-case perturbation model ($\gamma = 1$). Interestingly, the success probability decreases steeply as soon as a small amount of uncertainty is allowed (say, $\gamma = 0.01$, or 1% uncertainty budget) and then it saturates. In general, this analysis would allow a system designer to assess the "robustness" of the deployment protocol.

7.3 Deployment of Robotic Swarms

Finally, in this section we study how the success probability scales with the number of robots (we recall that the multi-robot deployment is considered successful if each target is reached by at least one robot). We consider an uncertainty budget:

$$\Gamma = 0.25 \times \sum_{(x,a)\in\mathcal{K}'} \bar{\varepsilon}(x,a).$$

Uncertainty budget	Theoretical succ. prob.	Empirical succ. prob.
$\gamma = 0$	0.9853	0.9840
$\gamma = 0.005$	0.8850	0.8910
$\gamma = 0.0075$	0.8390	0.8350
$\gamma = 0.01$	0.8227	0.8220
$\gamma = 0.0125$	0.8166	0.8120
$\gamma = 0.025$	0.7464	0.7450
$\gamma = 0.25$	0.7321	0.7340
$\gamma = 1$	0.7321	0.7310

Table 3: Sensitivity of success probability with respect to factor of uncertainty γ.

First, we test the strategy whereby targets are randomly assigned to robots according to a uniform probability distribution. We consider as temporal deadlines $D = \{20, 51, 82, 113, 144, 175, 206, 237, 268, 299\}$. The results are presented in Figure 4. One can notice that, as expected, the success probability increases with the number of robots and time threshold. We believe that this type of plots can be very valuable in practical applications, as one can then decide upfront how to pick the size of the team based on a given temporal deadline and a desired probability of success. Figure 4 indeed describes the interplay between the size of the team, the temporal deadline, and the probability of success. For example, by using the RCMDP algorithm with uniform random target assignment, for a time threshold equal to 175, one would require at least 118 robots in order to achieve a success probability larger than 0.8. These results do not consider congestion effects: inclusion of upper bounds on the number of robots that can traverse each edge of a map (that is, edge capacities) will be studied in future research.

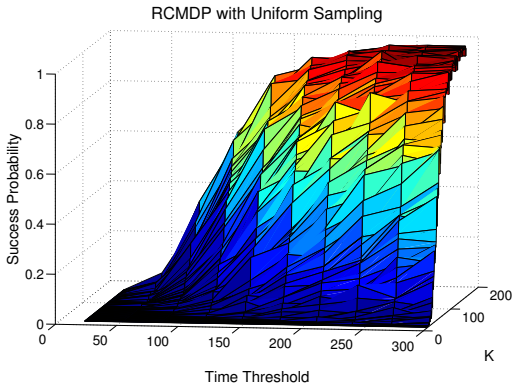

Fig. 4: Success rate as a function of the number of robots for different temporal deadlines using random uniform assignment.

We next test the strategy whereby targets are assigned to robots by executing Algorithm 1 (presented in Section 5). Results are presented in Figure 5. As in the previous case,

the success probability increases with the number of robots and time threshold, but it does so much faster. For example, with a time threshold equal to 175, one would require at least 69 robots (instead of 118 as in the previous case) in order to achieve a success probability larger than 0.8. Visually, this improved performance is evidenced by the steeper shape of the surface when compared with the analogous one in Figure 4.

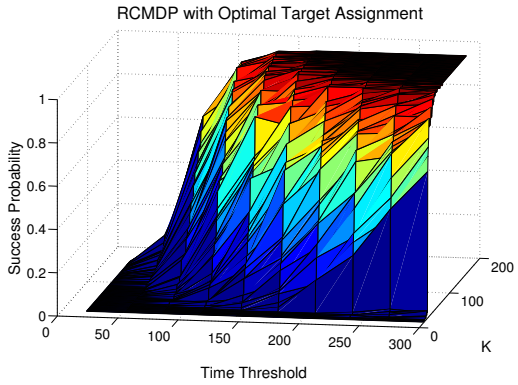

Fig. 5: Success rate as a function of the number of robots for different temporal deadlines using optimal target assignment.

8 Conclusions

In this paper we studied the rapid multi-robot deployment problem, where there is an inherent trade-off between mission success and speed of execution. We showed how the rapid deployment problem can be formulated within the theory of constrained Markov Decision Processes, whereby one seeks to compute policies that maximize the probability of successful deployment while ensuring that the expected duration of the deployment task is bounded by a given deadline. To account for uncertainties in the problem parameters, we considered a robust formulation and we proposed efficient solution algorithms, which are novel and of independent interest. Numerical experiments corroborated our findings and showed how the algorithmic machinery we introduced can be used to address relevant design questions. For example, it is possible to anticipate the performance of a team of a given size, or to decide the size of a team in order to achieve certain performance objectives.

Future research will develop in two directions. From the point of view of deployment, methods relying on explicit communication and coordination between the agents are of course of interest and will be investigated. From the point of view of modeling, additional or different types of uncertainties will be considered. In particular, it is of interest to study the impact of uncertainty in the objective costs and/or on the transition probabilities. This would for example be of interest when the safety functions characterizing the edges are only coarsely estimated.

Acknowledgements

Stefano Carpin is partially supported by ARL under contract MAST-SUPP-13-6-CNC. Any opinions, findings, and conclusions or recommendations expressed in these materials are those of the authors and should not be interpreted as representing the official policies, either expressly or implied, of the funding agencies of the U.S. Government.

The authors thank the reviewers for numerous thoughtful comments that helped to improve the quality of this paper.

References

[1] Bonabeau, E., Dorigo, M., and Theraulaz, G., 1999. *Swarm Intelligence: from Natural to Artificial Systems*, Vol. 4. Oxford University Press New York.

[2] Bullo, F., Cortés, J., and Martínez, S., 2009. *Distributed Control of Robotic Networks*. Princeton.

[3] Pavlic, T., and Passino, K., 2009. "Foraging Theory for Autonomous Vehicle Speed Choice". *Engineering Applications of Artificial Intelligence,* **22**(3), pp. 482–489.

[4] Cortes, J., Martinez, S., Karatas, T., and Bullo, F., 2004. "Coverage Control for Mobile Sensing Networks". *IEEE Transactions on Robotics and Automation,* **20**(2), pp. 243–255.

[5] Schwager, M., McLurkin, J., and Rus, D., 2006. "Distributed Coverage Control with Sensory Feedback for Networked Robots". In Proceedings of Robotics: Science and Systems.

[6] Morlok, R., and Gini, M., 2004. "Dispersing Robots in an Unknown Environment". In International Symposium on Distributed Autonomous Robotic Systems (DARS).

[7] Pearce, J., Rybski, P., Stoeter, S., and Papanilolopoulos, N., 2003. "Dispersion Behaviors for a Team of Multiple Miniature Robots". In IEEE International Conference on Robotics and Automation, pp. 1158–1163.

[8] Purohit, A., Zhang, P., Sadler, B., and Carpin, S., 2014. "Deployment of swarms of micro-aerial vehicles: from theory to practice". In Proceedings of the IEEE International Conference on Robotics and Automation, pp. 5408–5413.

[9] Pavone, M., Arsie, A., Frazzoli, E., and Bullo, F., 2011. "Distributed Algorithms for Environment Partitioning in Mobile Robotic Networks". *IEEE Transactions on Automatic Control,* **56**(8), pp. 1834–1848.

[10] Carpin, S., Chung, T., and Sadler, B., 2013. "Theoretical Foundations of High-Speed Robot Team Deployment". In IEEE International Conference on Robotics and Automation, pp. 2025–2032.

[11] Kloetzer, M., and Belta, C., 2007. "Temporal Logic Planning and Control of Robotic Swarms by Hierarchical Abstractions". *IEEE Transactions on Robotics,* **23**(2), pp. 320–330.

[12] Ding, X., Kloetzer, M., Chen, Y., and Belta, C., 2011. "Automatic Deployment of Robotic Teams". *IEEE Robotics & Automation Magazine,* **18**(3), pp. 75–86.

[13] Batalin, M., and Sukhatme, G., 2007. "The Design and

Analysis of an Efficient Local Algorithm for Coverage and Exploration Based on Sensor Network Deployment". *IEEE Transactions on Robotics,* **23**(4), pp. 661–675.

[14] Fink, J., Ribeiro, A., and Kumar, V., 2013. "Robust Control of Mobility and Communications in Autonomous Robot Teams". *IEEE Access,* **1**, pp. 290–309.

[15] Matignon, L., Jeanpierre, L., and Mouaddib, A., 2012. "Coordinated Multi-Robot Exploration Under Communication Constraints Using Decentralized Markov Decision Processes". In AAAI Conference on Artificial Intelligence, pp. 2017–2023.

[16] Ding, X., Pinto, A., and Surana, A., 2013. "Strategic Planning under Uncertainties via Constrained Markov Decision Processes". In IEEE International Conference on Robotics and Automation, IEEE, pp. 4568–4575.

[17] Napp, N., and Klavins, E., 2011. "A Compositional Framework for Programming Stochastically Interacting Robots". *International Journal of Robotics Research,* **30**(6), pp. 713–729.

[18] Altman, E., 1996. "Constrained Markov decision Processes with Total Cost Criteria: Occupation Measures and Primal LP". *Mathematical Methods of Operations Research,* **43**(1), pp. 45–72.

[19] Bertsekas, D., 2005. *Dynamic Programming & Optimal Control*, Vol. 1 and 2. Athena Scientific.

[20] Puterman, M., 1994. *Markov Decision Processes – Discrete Stochastic Dynamic Programming*. Wiley-Interscience.

[21] Altman, E., 1999. *Constrained Markov Decision Processes*. Stochastic modeling. Chapman & Hall/CRC.

[22] Ben-Tal, A., El Ghaoui, L., and Nemirovski, A., 2009. *Robust Optimization*. Princeton University Press.

[23] Bertsimas, D., and Sim, M., 2003. "Robust Discrete Optimization and Network Flows". *Mathematical Programming,* **98**(1-3), pp. 49–71.

[24] Rausand, M., and Høyland, A., 2004. *System Reliability Theory: Models, Statistical Methods, and Applications*, Vol. 396. John Wiley & Sons.

[25] Purohit, A., and Zhang, P., 2011. "Controlled-mobile Sensing Simulator for Indoor Fire Monitoring". In Wireless Communications and Mobile Computing Conference, pp. 1124–1129.

[26] Kumar, V., and Michael, N., 2012. "Opportunities and Challenges with Autonomous Micro Aerial Vehicles". *The International Journal of Robotics Research,* **31**(11), pp. 1279–1291.

[27] Kushleyev, A., Mellinger, D., and Kumar, V., 2012. "Towards a Swarm of Agile Micro Quadrotors". In Robotics: Science and Systems.

[28] Mellinger, D., Michael, N., and Kumar, V., 2012. "Trajectory Generation and Control for Precise Aggressive Maneuvers with Quadrotors". *The International Journal of Robotics Research,* **31**(5), pp. 664–674.

[29] Nesterov, Y., Nemirovskii, A., and Ye, Y., 1994. *Interior-point Polynomial Algorithms in Convex Programming*, Vol. 13. Society for Industrial and Applied Mathematics.

[30] Luenberger, D., 2003. *Linear and Nonlinear Programming*. Kluwer Academic Press.

Appendix

Proof of Lemma 2.2. Fix a policy π and an initial distribution β. First, we note that

$$\sup_{\varepsilon\in\mathcal{U}} d_\varepsilon(\pi,\beta) = \sum_{t=0}^{\infty} E_{\pi,\beta}\left[d(X_t,A_t)\right] + \sup_{\varepsilon\in\mathcal{U}} \sum_{t=0}^{\infty} E_{\pi,\beta}\left[\varepsilon(X_t,A_t)\right], \tag{14}$$

which follows from the additivity property of the series and expectation operator, and the fact that both series converge given our standing assumption of $\mathbf{X}'$-transient CMDP. Next we define $\psi^{\pi,\beta}(\varepsilon) := \sum_{t=0}^{\infty} E_{\pi,\beta}\left[\varepsilon(X_t,A_t)\right]$ and we show that it is a linear functional. Indeed, one has for all $\varepsilon_1,\varepsilon_2 \in \mathbb{R}^{|\mathcal{K}'|}$:

$$\begin{aligned}\psi^{\pi,\beta}(\varepsilon_1+\varepsilon_2) &= \sum_{t=0}^{\infty} E_{\pi,\beta}\left[\varepsilon_1(X_t,A_t)+\varepsilon_2(X_t,A_t)\right]\\ &= \sum_{t=0}^{\infty} E_{\pi,\beta}\left[\varepsilon_1(X_t,A_t)\right] + \sum_{t=0}^{\infty} E_{\pi,\beta}\left[\varepsilon_2(X_t,A_t)\right]\\ &= \psi^{\pi,\beta}(\varepsilon_1)+\psi^{\pi,\beta}(\varepsilon_2),\end{aligned}$$

where the second equality again follows from the additivity property of the series and expectation operators and the fact that both series converge. Analogously, one has for all $\varepsilon \in \mathbb{R}^{|\mathcal{K}'|}$ and $\alpha \in \mathbb{R}$

$$\begin{aligned}\psi^{\pi,\beta}(\alpha\varepsilon) &= \sum_{t=0}^{\infty} E_{\pi,\beta}\left[\alpha\varepsilon(X_t,A_t)\right]\\ &= \alpha \sum_{t=0}^{\infty} E_{\pi,\beta}\left[\varepsilon(X_t,A_t)\right] = \alpha\psi^{\pi,\beta}(\varepsilon).\end{aligned}$$

Hence, the functional $\psi^{\pi,\beta}$ is linear. Since $\mathcal{U}$ is a finite dimensional set and $\psi^{\pi,\beta}$ is a linear functional, it follows that $\psi^{\pi,\beta}$ is continuous. Note that $\mathcal{U}$ is closed and bounded, hence by the extreme value theorem the functional $\psi^{\pi,\beta}$ achieve its maximum. Combining this fact with equation (14), one obtains the claim.

Proof of theorem 2.3. As in the proof of Lemma 2.2, let $\psi^{\pi,\beta}(\varepsilon) := \sum_{t=0}^{\infty} E_{\pi,\beta}\left[\varepsilon(X_t,A_t)\right]$. As shown in Lemma 2.2, $\psi^{\pi,\beta}(\varepsilon)$ is a linear functional with respect to ε. The constraint $\max_{\varepsilon\in\mathcal{U}} d_\varepsilon(\pi,\beta) \le D$ is clearly equivalent to the constraint

$$\sum_{t=0}^{\infty} E_{\pi,\beta}\left[d(X_t,A_t)\right] + \max_{\varepsilon\in\mathcal{U}} \psi^{\pi,\beta}(\varepsilon) \le D.$$

Note that $\mathcal{U}$ is the intersection of a hyper-rectangle, i.e., $\{\varepsilon : 0 \le \varepsilon(x,a) \le \bar{\varepsilon}(x,a), \forall (x,a) \in \mathcal{K}'\}$, and a simplex, i.e., $\{\varepsilon : \sum_{(x,a)\in\mathcal{K}'} \varepsilon(x,a) \le \Gamma\}$. Thus, $\mathcal{U}$ has a finite number of vertices. Let $\mathcal{V}$ be the set of vertices of the polytopic set

$\mathcal{U}$ and let $|\mathcal{V}|$ be its cardinality. Since the maximum for a linear optimization problem over a bounded polytopic set is achieved at one of the vertices, then the constraint

$$\sum_{t=0}^{\infty} E_{\pi,\beta}\left[d(X_t,A_t)\right] + \max_{\varepsilon\in\mathcal{U}} \psi^{\pi,\beta}(\varepsilon) \leq D,$$

is equivalent to

$$\sum_{t=0}^{\infty} E_{\pi,\beta}\left[d(X_t,A_t)\right] + \psi^{\pi,\beta}(\varepsilon_j) \leq D \quad \text{for all } \varepsilon_j \in \mathcal{V}.$$

Therefore, problem RCMDP is equivalent to the problem:

$$\begin{aligned}
&\min_{\pi\in\Pi_M} c(\pi,\beta)\\
&\text{s.t.} \sum_{t=0}^{\infty} E_{\pi,\beta}\left[d(X_t,A_t)+\varepsilon_j(X_t,A_t)\right] \leq D \quad \text{for all } \varepsilon_j \in \mathcal{V},
\end{aligned}$$

where the number of constraints is $|\mathcal{V}|$. According to Theorem 8.1 in [18], this problem is equivalent to the problem

$$\begin{aligned}
\min_{\rho} \quad & \sum_{(x,a)\in\mathcal{K}'} \rho(x,a)c(x,a)\\
\text{s.t.} \quad & \sum_{y\in\mathbf{X}'}\sum_{a\in A(y)} \rho(y,a)\left[\delta_x(y) - \mathcal{P}_{yx}^a\right] = \beta(x), \forall x \in \mathbf{X}'\\
& \sum_{(x,a)\in\mathcal{K}'} \rho(x,a)(d(x,a)+\varepsilon_j(x,a)) \leq D \text{ for all } \varepsilon_j \in \mathcal{V}\\
& \rho(x,a) \geq 0,\ \forall (x,a) \in \mathcal{K}'.
\end{aligned}$$

Equivalency is in the sense that the solution to the above linear optimization problem induces an optimal policy, see equation (3) in section 2.2.

Finally, the above problem can be rewritten as

$$\begin{aligned}
\min_{\rho} \quad & \sum_{(x,a)\in\mathcal{K}'} \rho(x,a)c(x,a)\\
\text{s.t.} \quad & \sum_{y\in\mathbf{X}'}\sum_{a\in A(y)} \rho(y,a)\left[\delta_x(y) - \mathcal{P}_{yx}^a\right] = \beta(x), \forall x \in \mathbf{X}'\\
& \max_{\varepsilon\in\mathcal{U}} \sum_{(x,a)\in\mathcal{K}'} \rho(x,a)(d(x,a)+\varepsilon(x,a)) \leq D\\
& \rho(x,a) \geq 0,\ \forall (x,a) \in \mathcal{K}'.
\end{aligned}$$

This concludes the proof.