Rapid Deployment of Mobile Robots under Temporal, Performance, Perception, and Resource Constraints

Jose Luis Susa Rincon[1], Pratap Tokekar[2], Vijay Kumar[3], and Stefano Carpin[1]

Abstract—**We consider the problem where a team of mobile robots is tasked with collecting information about a set of stationary targets. There is a temporal deadline to complete the task, and the objective is to determine a control policy maximizing the probability of successfully completing the task within the assigned deadline. In addition, robots use imprecise sensors, and are subject to noisy dynamics. Furthermore, there are more targets than robots, so load sharing between robots is necessary. We model this problem using the theory of Constrained Markov Decision Processes (CMDP) and split the solution into two steps. First, policies to observe small subsets of targets are computed, and the proposed model and algorithm allow one to extract accurate information characterizing the performance of the computed control policies. In the second stage a subset of the computed policies is assigned to the robots for execution with the objective of maximizing a collective team performance function. To this end, we introduce a submodular objective function and a greedy approximation algorithm to solve this nonlinear assignment problem. Simulations demonstrate how these models can be used in practice to appropriately tune the parameters characterizing this problem and show how the approach favorably scales with the complexity of the problem.**

Index Terms—**Autonomous Agents; Cooperating Robots; Motion and Path Planning.**

I. Introduction

IN this paper we extend our former work on the *rapid deployment* problem where a team of robots is tasked with gathering information about multiple stationary targets under temporal and performance-based constraints. We consider a stochastic scenario with uncertainty affecting both sensing and control. Due to these disturbances, there is no guarantee that the task will be successfully completed. For example, robots may crash due to uncertainty in control, or may fail to observe a target due to erroneous sensor readings. Ideally, given a temporal deadline, the objective is to plan the actions for each robot in the team so that the probability of completing the task within the given deadline is maximized. We consider the case where the number of robots is smaller than the number of targets to observe. Therefore, to observe all targets, it is advantageous to split the task among the robots. This work significantly extends our former research in this domain. In [7], [9] we considered the case where the size of the team is much larger than the number of targets. In such scenario a *swarm approach* is possible, i.e., multiple robots are assigned to each target and robustness is obtained through redundancy. Moreover, in [7], [9] we did not consider uncertainty in sensing. In [21] we considered a persistent monitoring application where the number of targets to be observed exceeded the number of robots. However, no uncertainty was considered either in the dynamics of the robots or in the sensing process. Moreover, no performance constraints were given. In this paper we consider the situation where the number of targets is larger than the number of robots, uncertainty affects both the dynamics and perception, and the robots must complete the task within a given temporal deadline. For sensing, we assume that the outcome of observations is uncertain. In particular we suppose that even when a target can be observed, there is some probability of missed detection. This probability is not constant but is a function of the point from which the observation is made. With regards to the dynamics of the system, we consider a scenario where accuracy and speed vary. At each stage the robot is presented with a set of possible motion primitives to choose from. Each primitive is characterized in terms of execution time and success probability. For a wheeled robot one could assume that faster motions are associated with higher failure probabilities. On the other hand, for a small-size quadrotor it could also be that too slow motions result in higher failure rates. Our framework can accommodate both these cases, as long each primitive can be stochastically characterized. At the single robot level, the control policy has to select the actions to execute in each state with the objective of observing a subset of assigned targets within the given temporal deadline. At the team level it is instead necessary to split the targets between the robots. As in our previous work, we assume that robots do not communicate while the mission is executed. Therefore, replanning to re-balance the target assignment on the fly (e.g., to respond to individual failures) is not feasible because each robot is unaware of how

Manuscript received: Feb, 15, 2017; Revised May, 5, 2017; Accepted May, 29, 2017.

This paper was recommended for publication by Editor Nak Young Chong upon evaluation of the Associate Editor and Reviewers' comments. This research is partially supported by the Army Research Lab under contract W911NF-08-2-0004 (MAST-CNC-15-4-4). P. Tokekar is partially supported by National Science Foundation through award #1566247. V. Kumar is partially supported by grant ONR Grant N00014-15-1-2115. 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.

[1]Jose Luis Susa Rincon and Stefano Carpin are with the School of Engineering, University of California, Merced, CA 95343 USA. jsusarincon@ucmerced.edu; scarpin@ucmerced.edu

[2]Pratap Tokekar is with the Department of Electrical and Computer Engineering, Viginia Tech, Blacksburg, VA 24061 USA. tokekar@vt.edu

[3]Vijay Kumar is with the GRASP Laboratory, University of Pennsylvania, Philadelphia, PA 19104 USA kumar@seas.upenn.edu

Digital Object Identifier (DOI): see top of this page.

the mission unfolds at the team level. In rapid deployment the task is successfully completed if and only if all targets are observed within the deadline, irrespective of which robot observes a given target.

Rapid deployment finds application in various domains, including for example search and rescue where survivors need to be located quickly. In such a case the team of robots is tasked with gathering information (e.g., taking a picture) about a set of relevant locations of interest. The scenario we consider implies that robots in the team will in general be tasked with observing more than one target. This assumption requires us to solve two problems. First, it is necessary to compute a control policy for a robot to observe multiple targets while satisfying the given constraints and accounting for uncertain perception. Second, it is necessary to solve a task assignment problem to allocate subsets of targets to the robots in the team. The main contributions of this paper are the following:

- In Section III, we study the case of a single robot and multiple targets. This case is challenging, since it generalizes the orienteering problem [6], that is NP-Hard, by incorporating multiple costs and constraints. We show how the simpler problem of observing preassigned sequences of targets can be solved using the theory of CMDP and present a linear programming formulation to find the optimal solution.
- In Section IV we combine the single robot solutions to solve the rapid deployment problem at the team level.
- In Section V we consider the problem of splitting the targets among the robots. We formulate a submodular objective function leading to a greedy algorithm achieving a $1 - \frac{1}{e}$ approximation.
- In Section VI, we present various simulations studying the CMDP formulation. We assess the performance of our algorithm and study how it scales with the complexity of the problem.

In addition, related work is presented in Section II and conclusions are given in Section VII.

II. Related work

The rapid deployment problem was formally introduced in our former papers [7], [9]. Therein we considered the *swarm* approach where the number of robots was much larger than the number of locations to be observed. The original problem formulation considered stochastic models for state evolution but not for sensing. In [21] we studied the problem of persistent monitoring with robot teams where a multi-robot system is tasked with monitoring a set of locations larger than the size of the team. Therefore, each robot is assigned multiple targets to observe. However, in [21] we neither considered sensing errors nor stochastic motion models.

The problem we consider in this paper is related to the *team orienteering* problem [8]. Given a weighted graph with values associated to vertices, the problem is to find M paths, from a given start to a goal location, whose length does not exceed a given travel budget B while maximizing the sum of the values for the visited vertices. In this case M is the number of team members. This problem generalizes the orienteering problem [13] that is known to be NP-Hard and also APX-hard to approximate. These problems however feature only one cost per edge, whereas in our setting each edge in the graph is associated with multiple costs, i.e., time to complete the transition and failure probability. Moreover, the problem we consider features stochastic transitions between vertices, whereas in orienteering, and also in the related traveling salesman problem (TSP) [3], transitions are deterministic. It then follows that the problem we consider in this paper is significantly more complex and general than orienteering, and we will therefore have to settle for a suboptimal solution. The problem of finding tours to visit sites with stochastic transitions has been studied extensively in the operations research community [11], [4]. Laporte et al. [16] were among the first to study a variant of TSP known as the vehicle routing problem (VRP) [12] under stochastic travel as well as service times. The service time refers to the amount of time spent at each site. Since then, a number of algorithms for solving many variants of stochastic have been proposed [11], [4] typically based on chance-constrained optimization [18]. These works address not only stochastic times, but also cases where the sites themselves are stochastic [5], [10]. The problem considered in this paper differs from these works in that we do not have specific sites that must be visited by the robot. Instead, the algorithm must optimize for the sites to be visited based on the visibility sets of the targets and other input parameters. Our problem formulation is based on a graph abstraction of an occupancy map with stochastic motion primitives. Graph abstractions can be readily extracted from occupancy grid maps [15] and the recent work by Karydis et al. has shown how stochastic primitives compatible with the assumptions made in this paper can be generated using a data-driven approach [14].

III. Observing Multiple Targets with One Robot

We present a model for the problem we consider, and then provide a recap about CMDPs. We then show how CMDPs can be used to solve the problem of observing multiple targets with one robot under temporal and probabilistic constraints, and with uncertain dynamics and sensing.

A. Environment model

The environment is modeled by a graph $G = (X, E)$ where the vertex set X is a set of locations and an edge $e = (x, y)$ indicates that there exist one or more stochastic motion primitives to move from x to y (see Fig. 1).

Let a be one of the stochastic motion primitives associated with edge connecting x to y. Motion primitive a will succeed or fail with a certain probability, and will take a certain time to execute. Different primitives between the same vertices x and y are characterized by different success probabilities and time. These primitives model the different ways in which a robot can move between two locations. In the following, $\mathcal{P}^a_{xy}$ is the probability that the motion primitive a succeeds in going from x to y whereas $1 - \mathcal{P}^a_{xy}$ is the probability it fails. Noisy observations are added as follows. Let $T = \{t_1, \ldots, t_M\} \subset X$ be a set of M *stationary* targets. An observation action performed by a robot can detect these targets. For each target

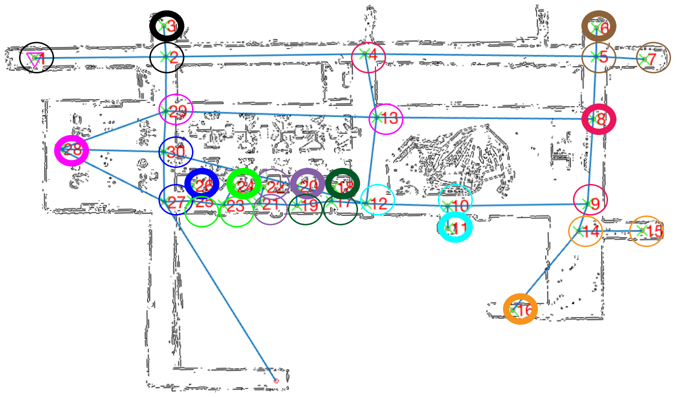

Fig. 1: A possible setup for rapid deployment. Ten different targets are outlined by solid thick circles (visibility sets are displayed by thin circles of the same color.)

$t \in T$ we define a visibility set $\mathcal{V}(t) \subset X$, i.e., a set of vertices from which the target can be detected if the robot performs the *observation* action. For simplicity, we assume that the visibility sets are disjoint, i.e., $t_i \neq t_j \Rightarrow \mathcal{V}(t_i) \cap \mathcal{V}(t_j) = \varnothing$. For $x \in \mathcal{V}(t_i)$, let $\mathcal{P}(x, o)$ be the probability that the robot will detect t_i when executing the observation action o in x. Note that since the visibility sets are assumed to be disjoint, this probability is unique and by definition larger than 0, otherwise x would not be in $\mathcal{V}(t_i)$. Note however that this assumption is introduced only to simplify the presentation and that the algorithms do not critically hinge on this hypothesis.

B. Background on CMDPs

The reader is referred to [7], [9] and to [1] for a general introduction to CMDPs. In the following we provide just the minimum definitions needed for our model. A finite, stationary, discrete time CMDP is defined by a tuple $(\mathbf{X}, A, \beta, c, \mathcal{P}, \{c_i\}_{i=1}^{L}, \{D_i\}_{i=1}^{L})$ where:

- $\mathbf{X}$ is a finite set with n states.
- A is a finite set of actions, where $A(x)$ is the set of actions that can be executed when the system is in state $x \in \mathbf{X}$. We define $\mathcal{K} = \{(x, a) : x \in \mathbf{X}, a \in A(x)\}$ as the set of allowable state/action pairs.
- β is a mass distribution over $\mathbf{X}$ for the initial state, i.e., $\beta(x) = \Pr[X_0 = x]$.
- $c : \mathcal{K} \to \mathbb{R}_{\geq 0}$ is the non-negative cost incurred when applying action $a \in A(x)$ while in state $x \in \mathbf{X}$.
- $c_i : \mathcal{K} \to \mathbb{R}_{\geq 0}$ are L additional non-negative costs incurred when applying action $a \in A(x)$ while in state $x \in \mathbf{X}$.
- D_i are L upper bounds for the sum of the costs c_i.
- $\mathcal{P}^a_{xy}$ is the one step transition probability from state x to state y when action a is applied, i.e., $\mathcal{P}^a_{xy} = \Pr[X_{t+1} = y | X_t = x, A_t = a]$.

In the following we indicate with X_t the random variable for the state at time t, and A_t for the action taken at time t. A randomized, Markov policy π is a function assigning to each state $x \in \mathbf{X}$ a mass distribution over $A(x)$. While in the theory of MDPs deterministic policies suffice for optimality, in CMDPs the optimal policy is in general randomized.

In the following we focus on *transient* CMDPs, defined as follows. The state space $\mathbf{X}$ is partitioned into sets $\mathbf{X}'$ and $\mathbf{M}$. A policy π is transient in $\mathbf{X}'$ if

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

where $\Pr^{\pi}_{x_0}[X_t = x]$ is the probability that $X_t = x$ given the initial state is $x_0 \in \mathbf{X}'$ and the policy π is followed. The first condition implies that eventually the state will enter the absorbing set $\mathbf{M}$, and the second condition ensures that once the state enters $\mathbf{M}$ it will remain there. A CMDP for which all policies are transient in $\mathbf{X}'$ is called a $\mathbf{X}'$-transient CMDP. In the following we assume that $c(x, a) = c_i(x, a) = 0$ for each $x \in \mathbf{M}$ and each $1 \leq i \leq L$. For a transient CMDP, the following total costs are then defined and finite:

$$c(\pi, \beta) = \mathbb{E}\left[\sum_{t=0}^{\infty} c(X_t, \pi(X_t))\right]$$

$$c_i(\pi, \beta) = \mathbb{E}\left[\sum_{t=0}^{\infty} c_i(X_t, \pi(X_t))\right].$$

In the definitions, expectations are taken with respect to both π and β. Let Π_M be the set of all randomized, Markov policies. The total cost CMDP problem is therefore defined as follows:

$$\pi^* \in \arg\min_{\pi \in \Pi_M} c(\pi, \beta) \qquad (1)$$
$$\text{s.t. } c_i(\pi, \beta) \leq D_i, \quad 1 \leq i \leq L.$$

C. A model based on CMDPs

Throughout this work we assume that the state of the robot is its location in the map, i.e., the vertex in the graph, and that it is observable. Known location is a realistic assumption when robots are operating outdoors and GPS is available. Moreover, if a map of the environment is given, localization can be solved using various existing algorithms [20]. Our solution is then formulated as a feedback policy π mapping the location onto a mass distribution of the set of possible actions. Using a plain MDP model, for a given target t_i it would be straightforward to compute a policy π determining for each state the primitive to execute to maximize the probability of eventually observing t_i. However, this approach is inadequate because it does not consider the time taken to complete the observation task, and also because some robots will be necessarily tasked with observing more than one target (as there are more targets than robots). To this end, it is necessary to extend the state space with some *memory* so that a robot can track the targets it has already seen. One possible approach is to assign a certain subset of targets to the planning algorithm and let the planner determine a policy to visit all of them in an unspecified order. However, it is easy to see that if the robot is assigned k targets to observe, keeping track of those observed already in an arbitrary order would entail an exponential growth of the state space (by 2^k to be precise). To limit the growth in the state space, the planner is therefore not only given a set of targets to observe, but also *the order* in which they should be observed. This assumption implies that given a sequence, if the robot cannot observe the ith target in the sequence, it will not move on to observe the successive ones. This is consistent with our definition of the problem, whereby the task is successfully completed if and only if all targets are observed and therefore skipping one target would lead to failure. Alternatively, if the definition of the problem is changed to allow skipping targets, the same construction we present in the following can incorporate a counter so that the robot will skip a target after a series failed detection attempts. To be specific, if the robot is assigned k targets to be observed in sequence, say $t_1, t_2, \ldots, t_k$, the state space X grows by a factor $k+1$ since to every state we associate a string with k bits indicating which targets have been observed already. Note that k bits are sufficient (instead of 2^k) because the order is specified. For example, if $k = 3$ it means the planner needs to observe 3 targets and the state $(000, x)$ indicates that the robot is in state x and has not seen yet any target. From such state the robot can either remain there, transit to $(000, y)$ for $y \neq x$ or to state $(100, x)$. In the first case it means that it moves to a different state y without making any observation, whereas in the second case it means that it has successfully observed the first target while in x. Observe that moving to state $(010, x)$ is not permitted because this would mean observing the second target before the first one, and this is is not consistent with our assumptions. To reward the planner when a target is observed, we introduce *augmented* states of the form (n, x') where n is a string of k bits satisfying the constraints we formerly described (see Figure 2). A reward function taking advantage of these additional states will be introduced later on. When the robot makes an observation from a state x_j belonging to the visibility set of a target, it will detect the target with probability $P(x_j, o)$. In this case it makes the two transitions indicated in Figure 2 to mark that it has seen the target. If the robot is in state (n, x_j) and $x_j \in \mathcal{V}(t_k)$ but t_k is not the next target to be observed in the sequence encoded by n, then the observation action cannot be executed.

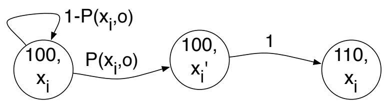

Fig. 2: When a successful observation from state x_i is made, the state goes through an augmented state x'_i and from there it executes its only action which with probability 1 goes back to the state x_i but with the corresponding bit modified.

Starting from $G = (X, E)$ and an ordered sequence of k targets T, we can then build a CMDP as follows. Let $\mathbb{N}_T$ be the set of k bit strings we formerly defined. From X, a state space $X' = \mathbb{N}_T \times X$ with $\mathbb{N}_T \times |X|$ states is built attaching k bits to each state in X. The expanded states are defined as follows. Let, $\mathcal{T} = \cup_{j=1\ldots M} V(t_j)$, i.e., the union of all the vertices from which one of the targets can be seen. Next, we define $\mathcal{T}' = \mathbb{N}_T \times \mathcal{T}$. To distinguish elements of $\mathcal{T}'$ from elements of X', elements of the former set will be indicated as (n, x') whereas elements for the latter will be indicated as (n, x). Finally we introduce a sink state $\mathcal{S}$ and a final state $\mathcal{F}$. The sink state $\mathcal{S}$ is used to represent *failure*, i.e., to model the event of a robot ceasing to correctly function. Consequently, once it is entered, no more actions can be taken by the robot. The full state space is $\mathbf{X} = X' \cup \mathcal{T}' \cup \{\mathcal{S}, \mathcal{F}\}$, and the absorbing set is $\mathbf{M} = \{\mathcal{F}\}$. We next define the action set of each state. For each state of type (n, x) we add an action a if motion primitive a is available in vertex x. Moreover, for states belonging to the visibility sets, we add an action o for the *observe* action, but only if observing a target from that state is compatible with the assigned sequence of targets. These two types of actions apply only to states in X'. To each state in $\mathcal{T}'$ we add a single action going back to the corresponding state with the flipped bit, as we illustrated in Figure 2. Finally, to the sink state $\mathcal{S}$ we add a single action $a_{\mathcal{S}}$ going to $\mathcal{F}$. The mass distribution β is set to 1 to the deployment vertex where the robots start from, and 0 elsewhere. The transition function is defined as follows. When an observation action is executed, it is defined as shown in Figure 2, i.e., it remains in the same state with probability $1 - P(x, o)$ (no detection), or it goes to the augmented state with probability $P(x, o)$ (detection). If a is a motion primitive between x and y, then $\mathcal{P}^a_{xy}$ is given by the success probability of the primitive, and we set $\mathcal{P}^a_{x\mathcal{S}} = 1 - \mathcal{P}^a_{xy}$, i.e., if the primitive fails the state goes to the sink state $\mathcal{S}$. Actions for the states $(n, x') \in \mathcal{T}$ move the state back to (n', x) with probability 1, where n' differs from n for the flipped bit. The only action in $\mathcal{S}$ is $\mathcal{P}_{\mathcal{S}\mathcal{F}} = 1$, i.e., it deterministically moves the state to the absorbing state. To conclude, we define two different costs and a reward (hence $L = 2$ in our model.) First, we define a cost $c(x, a)$, such that $c(\mathcal{S}, a_{\mathcal{S}}) = 1$ and is 0 everywhere else. The purpose of this cost is to accurately determine failure probabilities, as we did in [7], [9]. Next we define a cost $d(x, a)$ for all state actions. d models the time to take an action. If a is a motion primitive, then $d(x, a)$ is the time to complete the primitive. If a is an observation action, we set this to the cost of making an observation, e.g., the robot may have to stop to take a picture. For all other actions the d cost is 0. Finally we introduce a reward function $e(x, a)$. The reward function is a constant, say 1, for all actions associated with states of the type (n, x') and 0 elsewhere. As it will be seen in the following, by posing the question as an optimization problem, the planner will produce policies maximizing the expected number of observed targets in the sequence. Starting from the above definitions, we can then formulate the following single-robot multiple target problem.

Single robot, multiple target observation problem (SRMTO): Given an environment $G = (V, E)$, an ordered sequence of targets $\mathcal{G} \subset \{t_1, \ldots, t_M\}$ with visibility regions $\mathcal{V}(t_i)$, parameters $\mathbf{X}, A, \mathcal{P}, c, d, e, \beta$ as per the above definition, a temporal deadline T_f and a probability bound P_f, determine an optimal randomized policy $\pi^* : \mathbf{X}' \rightarrow \mathbb{P}(A)$ mapping state into actions maximizing the expected number of observed targets in the given order, while ensuring the temporal and performance bounds are met in expectation.

The SRMTO formulation, and the following theorem, are an instance of the total cost CMDP problem formulated in Eq. (1).

Theorem 1: Let $\mathbf{X}' = \mathbf{X} \setminus \{\mathcal{F}\}$ and $\mathcal{K}' = \{(x, a) : x \in \mathbf{X}', a \in A(x)\}$. The following constrained linear program has a solution if and only if the SRMTO problem has a solution and there is a one-to-one correspondence between the solution vector $\rho(x,a)$ and the optimal policy π^*.

$$\begin{aligned}
\max & \sum_{(x,a)\in\mathcal{K}'} \rho(x,a)e(x,a) \\
\text{s.t.} & \sum_{(x,a)\in\mathcal{K}'} \rho(x,a)c(x,a) \leq P_f \\
& \sum_{(x,a)\in\mathcal{K}'} \rho(x,a)d(x,a) \leq T_f \\
& \sum_{y\in\mathbf{X}'} \sum_{a\in A(y)} \rho(y,a)(\delta_x(y) - \mathcal{P}^a_{yx}) = \beta(x)\ \forall x \in \mathbf{X}' \\
& \rho(x,a) \geq 0
\end{aligned}$$

where $\delta_x(y)$ is 1 when $x = y$ and 0 otherwise.

Proof Sketch: the proof builds upon the fact [1] that the optimization variables $\rho(x,a)$ are interpreted as *occupancy measures*, defined as:

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

Because of the way we defined the costs e, these are 1 only on the outgoing arc from states of type (n,x'), i.e., states associated with the observation of a target. Hence the objective function is the expected number of observed targets. In our previous work [9] we have proved that $\sum_{(x,a)\in\mathcal{K}'} \rho(x,a)c(x,a)$ is the expected failure probability and this is bounded by P_f in the first constraint. Similarly, in [9] we have shown that $\sum_{(x,a)\in\mathcal{K}'} \rho(x,a)d(x,a)$ is the expected time to complete a strategy, and this is bounded in expectation by T_f in the second constraint. □

Note that an occupancy measure is in general not a probability, but it is a probability if it sums the probabilities of mutually disjoint events. For example, in our former papers we showed that $\rho(\mathcal{S}, a_\mathcal{S})$ is the probability of a trajectory going through the sink state and therefore by definition the probability that a robot fails during its mission. Following the same reasoning one can show for a given target $t_i \in T$ the solution to the SRMTO problem provides the probability that the target is observed. This probability is

$$p_i = \sum_{x\in\mathcal{V}'(t)} \rho(x,a) \tag{2}$$

where $\mathcal{V}'(t)$ is the set of states of the type (n,x') with $x \in \mathcal{V}(t)$. To see why this relationship holds, it suffices to observe that for each target t_i each trajectory will go through only one of the states in $\mathcal{V}'(t)$. We can find the solution vector $\rho(x,a)$ by solving the linear program formulated in Theorem 1. The robot can execute the optimal policy using the corresponding occupancy measures (see [7], [9] for details.).

IV. Solving the rapid deployment problem

In the rapid deployment problem one is given N robots, a set of M targets T, and a temporal deadline T_f. The objective is to split the targets among the robots and compute the associated control policy for each robot so that the team as a whole maximizes the probability of completing the task within the target temporal deadline. The technique we presented in the previous section computes the control policy for a single robot under the assumption that the sequence of targets is given, as well as a probability bound P_f. However, both these elements are not part of the input. To fill this gap, an iterative algorithm seeking for the maximum probability of success can be developed, where the optimal value for P_f is determined with a sequential search.

At each iteration, the algorithm determines a set of $K > N$ sequences of targets together with the optimal policy for each sequence, solving the corresponding SRMTO problem. Note that the algorithm may not necessarily find an optimal solution for each of the K sequences. The algorithm then assigns to each robot one of the sequences for which a solution was found, and computes the success probability P' for the whole team. Upon termination, the algorithm returns the policies and the assignment of policies to robots that yields the best group performance. Algorithm 1 sketches this approach.

```
Data: $G = (X,E)$, set of targets $T = \{t_1, t_2, \ldots, t_M\}$,
      temporal deadline $T_f$, number of robots $N$
Result: A policy for each robot, and success probability
      $P'$
1 Generate $K > N$ sequences of targets $S_1, \ldots, S_K$;
2 for $P_f \leq P_f^{MIN}$ to $P_f^{MAX}$ do
3     Solve SRTMO for each sequence with $P_f$ and $T_f$;
4     Assign one sequence to each robot and compute $P'$;
5     if $P'$ is best then
6         Save Assignment and $P'$;
7 return Assignment and $P'$
```

Algorithm 1: Rapid deployment

Algorithm 1 handles two different probabilities, i.e., P_f and P'. P_f is the failure probability bound used in the CMDP described in Theorem 1. This is the bound for the failure probability for each policy computed for the K sequences. Since this is not part of the input, a linear search is performed over the discretized range $[P_f^{MIN}, P_f^{MAX}]$. P_f is the failure probability for a single robot executing one of the policies. Once a policy for all sequences is computed and an assignment is selected, the *group success* probability P' is computed, and this is a a simple algebraic exercise building upon Eq. (2). Finally, note that the relationship between P_f and P' is not necessarily monotone, i.e., while increasing the individual robot failure probability P_f the group failure probability P' may increase or decrease.

Two problems still need to be solved, i.e., generating the sequences of targets (line 1), and assigning policies to the robots (line 4). Generating all target sequences is practically unfeasible as soon as T features more than an handful of targets, so it is necessary to consider only a *small* set of sequences. Sequences are generated as follows. First we solve the TSP problem on the complete graph of all M targets. The cost between vertices x and y is the average time to execute all motion primitives going from x to y. In our implementation we use the *Concorde* approximate solver [2] to quickly determine a tour. Next, we split the tour into subsequences of up to i targets with $3 \leq i \leq M$. It therefore follows that with this strategy the number of generated sequences K bounded by $\sum_{i=3}^{M} = \lceil \frac{M}{i} \rceil$, i.e., K is $\mathcal{O}(M \log M)$. The reason to discard sequences of length 1 and 2 comes after having observed that they rarely contribute to the optimal solution. The last remaining problem is group assignment, i.e., deciding which sequence of targets to assign to each robot. This is discussed in the next section.

V. Group Assignment Problem

Let $S_1, \ldots, S_K$ be the groups generated by Algorithm 1 for which the SRMTO problem admits a solution. For a solution to the rapid deployment problem to exist, it must be that $\cup_{i=1}^{K} S_i = T$. If this is not the case, then there exists at least one target that cannot be reached.

For each sequence S_j let ρ_j be the vector computed by solving SRMTO for fixed values of T_f and P_f. As formerly stated, the ρ_j vector encodes valuable information. For example, it allows to determine the expected number of targets seen by the optimal policy π^* solving SRMTO(S_j), as well as the probability that each of the targets in S_j will be observed while executing π^* (Eq. (2)). That is to say that if $t_i \in S_j$, then ρ_j, combined with the underlying model, gives the probability p_{ij} that t_i will be observed by the robot while executing the optimal policy π_j^* returned when solving SRMTO(S_j). More specifically, assuming $t_i \in S_j$, for each $v \in \mathcal{V}(t_i)$ define

$$\mathcal{X}_v = \{(n, x') \in \mathcal{T}' \mid x' = v\}$$

Then it is easy to verify that

$$p_{ij} = \sum_{v \in \mathcal{V}_i} \sum_{y \in \mathcal{X}_v} \rho_j(y, a) \tag{3}$$

where $y \in \mathcal{X}_v$ indicates a composite state (n, x'). The validity of this relationship is immediate to verify by writing down the explicit formula for occupancy measures considering that states in $\mathcal{T}'$ have just one action, and that each target can be observed just once due to the structure of the underlying state space. If $t_i \notin S_j$, then we set $p_{ij} = 0$.

The objective of the group assignment step is to pick N of the K sets $S_1, \ldots, S_K$. Different objective functions can be considered when making the selection. In the following we cast the problem as an instance of a nonlinear assignment problem related to the "static weapon target assignment" (SWTA) problem (see [19], ch. 3). For each target $t_i \in T$, let V_i be its importance, i.e., a weighting factor indicating how important it is to reach it. If V_i is not provided, than we can set all weights to 1. Moreover, as per Eq. (3), let us define the *survival rate* as $q_{ij} = 1 - p_{ij}$. This is the probability that target t_i will not be observed if group S_j is assigned to one robot for execution. For each of the K groups, we define a binary variable x_j indicating whether S_j is selected or not. The objective function we consider is then

$$\sum_{i=1}^{M} V_i [\Pi_{j=1}^{K} (q_{ij})^{x_j}] \tag{4}$$

subject to the constraint that we pick at most N groups. The interpretation of the formula is as follows. For each target t_i and each assignment of variables x_j, $\Pi_{j=1}^{K}(q_{ij})^{x_j}$ is the probability that t_i is not observed by any of the robots. Note that if $t_i \notin S_j$ then $q_{ij} = 1$. The objective function is then the sum of these probabilities for all targets, weighted by the corresponding importance V_j. If all V_js are one, then Eq. (4) is the expected number of targets not observed by any robot. We can then formulate the Group Assignment problem.

Group Assignment (GA). Let $S_1, \ldots, S_K$ be K sequences of targets and for each target i and group j, let q_{ij} be the survival rate for i-th target and the j-th group. Select N groups of targets to minimize Eq. (4).

SWTA is known to be NP-complete. Therefore, to solve the GA problem one can follow at least two strategies. First, GA could be formulated as the following integer optimization problem:

$$\min_{x_1, \ldots, x_K} \sum_{i=1}^{M} V_i [\Pi_{j=1}^{K} (q_{ij})^{x_j}]$$
$$\text{s.t.} \sum_{j=1}^{K} x_j = N \quad x_j \in \{0, 1\} \quad 1 \le j \le K.$$

For problems of moderate size, this can be exactly solved using a branch-and-bound approach and one could then take advantage of commercially available solvers. Alternatively, and this is the approach we present in the following, an approximate solution can be determined using a greedy approach for the case where an exact method is too onerous.

Let A be a group of sequences from $S_1, \ldots, S_K$, and let us define the function

$$f(A) = -\sum_{i=1}^{M} V_i [\Pi_{j \in A} q_{ij}]. \tag{5}$$

The GA problem is then equivalent to the problem of selecting N elements from $S_1, \ldots, S_K$ to maximize $f(S_{i_1} \cup \ldots S_{i_N})$.

Theorem 2: Function f as defined in Eq. (5) is submodular. The proof of the theorem is given in the appendix. According to this theorem one can use the well known greedy algorithm due to Nemhauser et al. [17] and obtain an approximated solution with a constant $1 - \frac{1}{e}$ approximation factor.

VI. Simulations

In this section we present some simulations showing how our solution can not only be used to determine the optimal control policy, but also to infer interesting design and analysis properties. For example it is possible to assess a priori what is the probability of successfully completing a rapid deployment task for a given temporal deadline T_f and a certain number of robots N.

First, for the environment shown in Figure 1 we study how the probability of successfully completing the task varies with the number of robots and the temporal deadline T_f. This is shown in Figure 3.

The different curves are not only associated with control strategies, but allow us to anticipate the performance of the team. This way, based on the desired probability of success P', one can either alter the temporal deadline T_f or allocate more robots to the task. A comment is in order to explain why there is a drop in performance for $T_f = 500$. This is

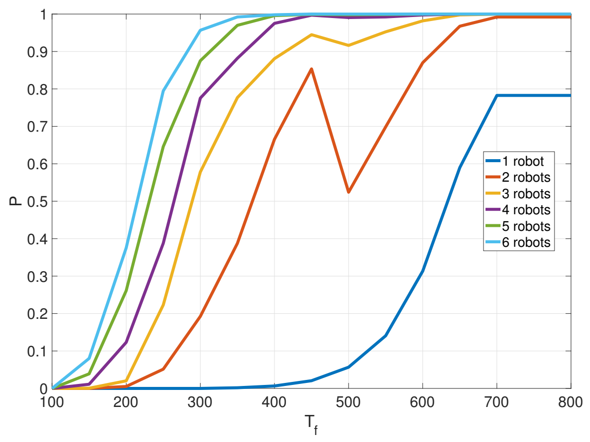

Fig. 3: Success probability for different temporal deadlines for different number of robots in the team for the case where 10 targets are considered.

particularly evident for the case where two robots are considered. The reason is that for the target assignment problem we rely on a greedy allocation algorithm that is known to be suboptimal. For that specific combination of temporal deadline and number of robots, the greedy strategy picks a suboptimal target assignment that yields a performance drop. The trend of the curves can be used by a human operator (e.g., the incident commander in a search and rescue task) to decide how to allocate resources (how many robots) or how to pick the temporal deadline T_f to achieve a desired confidence level in terms of probability of success.

An important question to ask is how the method scales with the number of robots N and the number of targets M. The algorithm relies on the solution of two subproblems, namely SRMTO (policy calculation) and GA (group assignment). The greedy solution to the GA problem is a function of both the number of targets and the number of robots. In particular, it is $\mathcal{O}(KN^2)$. This bound follows because we have to iterate N times over the K sequences, and it takes $\mathcal{O}(N)$ to evaluate each candidate sequence. Considering that in Section IV we clarified that the number of sequences K is $\mathcal{O}(M \log M)$, it follows that the complexity of the greedy algorithm for GA is $\mathcal{O}(M \log MN^2)$. If instead one opts for the exact method, then it is necessary to solve an integer optimization problem with K binary variables.

The SRMTO problem is instead by definition independent from the number of robots because it deals with the generation of a single robot policy. We therefore consider only the dependency between the time to solve SRMTO and the number of targets M. More targets mean more subsequences to consider, with the associated solution of various linear programs to solve the SRMTO problem instance. As the number of targets grows, there is an increase in the size of the linear programs we need to solve because there will be subsequences with more targets. Moreover, the number of subsequences of targets grows as well. Figures 4 and 5 analyze this growth as a function of the number of targets. In particular, Figure 4 displays the average time (with standard deviation bars) to solve a single linear program[1] as a function of the number of targets. As it can be seen, the growth trend is roughly linear. Figure 5 instead plots the trend for the sum of the time to solve all the linear programs associated with a specific temporal deadline. The curve shows mean and standard deviation of the trends for different temporal deadlines. The small standard deviation confirms that this is largely independent from the T_f value and unveils a quadratic growth.

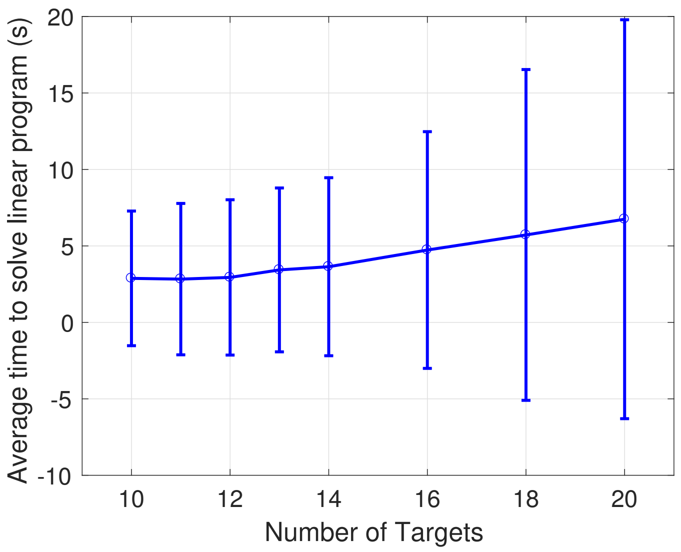

Fig. 4: Average time to solve a single linear program for the SRMTO problem as a function of the number of targets (average taken over all linear programs to be solved for a specific number of targets).

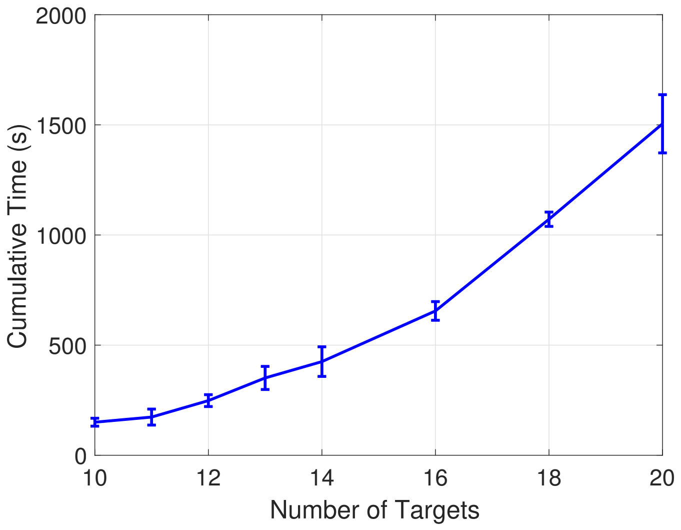

Fig. 5: Average cumulative time to solve all linear programs to solve the SRMTO problem as a function of the number of targets (average taken over all temporal deadlines considered for a given number of targets).

[1]In our implementation, we rely on Matlab's function `linprog`.

VII. Conclusions and future work

In this paper we have extended our previous work on the rapid deployment problem by considering two important generalizations, namely that the robots are subject to imperfect observations and that there are more targets to observe than robots available. This problem is significantly more complex than the one we considered before and is associated with some hard problems such as orienteering. In particular, it features multiple costs associated for every edge, such as the probability of success and the time to complete a primitive. We have shown that the CMDP framework we formerly formulated can still be extended to tackle a constrained robot where a single robot is tasked with observing a sequence of targets under temporal and probabilistic constraints. This building block can then be used to solve the original problem, i.e., maximizing the probability that the task is solved within a given temporal deadline. The CMDP model, in particular, allows to precisely predict success and failure probabilities at the single robot and at the team level. In addition, we have shown that the problem of assigning groups of targets to robots is related to the SWTA problem that can be approximately solved using a greedy algorithm. Our simulations show how the results presented in this paper can be used by a human supervisor to make informed decisions about how to allocate resources like the number of robots or the time to allocate. The method we proposed scales quadratically with the number of robots and targets. In terms of future work, we are currently working on determining motion primitives for both wheeled and microUAVs to test this planner on the field.

Appendix

Proof: Given a finite set N, a function $g : 2^N \to \mathbb{R}$ is submodular if for every $X, Y \subset N$ with $X \subseteq Y$ and every $z \in N \setminus Y$ the following condition holds:

$$g(X \cup \{z\}) - g(X) \geq g(Y \cup \{z\}) - g(Y). \tag{6}$$

First note that for $X = Y$ the submodularity condition is trivially verified. Let $X = \{S_{i_1}, \dots, S_{i_k}\}$ and $Y = \{S_{i_1}, \dots, S_{i_k}, S_{i_{k+1}}, \dots, S_{i_{k+p}}\}$ be two subsets of $\{S_1, \dots, S_K\}$ with $X \subset Y$, and let $S_z \notin Y$. The left hand side of Eq. (6) is then written as:

$$\begin{aligned}
&f(X \cup \{S_z\}) - f(X) = \\
&\quad - V_1[q_{1i_1} \dots q_{1i_k} q_{1z}] \dots - V_M[q_{Mi_1} \dots q_{Mi_k} q_{Mz}] \\
&\quad + V_1[q_{1i_1} \dots q_{1i_k}] \dots + V_M[q_{Mi_1} \dots q_{Mi_k}] = \\
&\quad V_1[q_{1i_1} \dots q_{1i_k}(1 - q_{1z})] + \dots \\
&\quad + V_M[q_{Mi_1} \dots q_{Mi_k}(1 - q_{Mz})].
\end{aligned}$$

Similarly, the right side of Eq. (6) is

$$\begin{aligned}
&f(Y \cup \{S_z\}) - f(Y) = \\
&\quad - V_1[q_{1i_1} \dots q_{1i_k} q_{1i_{k+1}} \dots q_{1i_{k+p}} q_{1z}] \dots \\
&\quad - V_M[q_{Mi_1} \dots q_{Mi_k} q_{Mi_{k+1}} \dots q_{Mi_{k+p}} q_{Mz}] \\
&\quad + V_1[q_{1i_1} \dots q_{1i_k} q_{1i_{k+1}} \dots q_{1i_{k+p}}] \dots \\
&\quad + V_M[q_{Mi_1} \dots q_{Mi_k} q_{Mi_{k+1}} \dots q_{Mi_{k+p}}] \\
&\quad = V_1[q_{1i_1} \dots q_{1i_k} q_{1i_{k+1}} \dots q_{1i_{k+p}}(1 - q_{1z})] + \dots \\
&\quad + V_M[q_{Mi_1} \dots q_{Mi_k} q_{Mi_{k+1}} \dots q_{Mi_{k+p}}(1 - q_{Mz})].
\end{aligned}$$

The submodularity condition is then verified because the weights V_j are positive and $q_{ij} \leq 1$ for each i, j. $\square$

References

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

[2] D. Applegate, R. Bixby, V. Chvatal, and W. Cook. Concorde TSP solver. `www.math.uwaterloo.ca/tsp/concorde.html`, 2006.

[3] S. Arora. Polynomial time approximation schemes for euclidean traveling salesman and other geometric problems. *Journal of the ACM*, 45(5):753–782, 1998.

[4] E. Berhan, B. Beshah, D. Kitaw, and A. Abraham. Stochastic vehicle routing problem: A literature survey. *Journal of Information & Knowledge Management*, 13(03):1450022, 2014.

[5] D. J. Bertsimas. A vehicle routing problem with stochastic demand. *Operations Research*, 40(3):574–585, 1992.

[6] A. Blum, S. Chawla, D. R. Karger, T. Lane, A. Meyerson, and M. Minkoff. Approximation algorithms for orienteering and discounted-reward TSP. *SIAM Journal on Computing*, 37(2):653–670, 2007.

[7] S. Carpin, M. Pavone, and B.M. Sadler. Rapid multirobot deployment with time constraints. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1147–1154, 2014.

[8] I.-M. Chao, B.L. Golden, and E.A. Wasil. The team orienteering problem. *Eurupean Journal of Operational Research*, 88:464–474, 1996.

[9] Y-L. Chow, M. Pavone, B.M. Sadler, and S. Carpin. Trading safety versus performance: Rapid deployment of robotic swarms with robust performance constraints. *ASME Journal of Dynamical Systems, Measurements and Control*, 137(3):031005, 2015.

[10] M. Dror, G. Laporte, and P. Trudeau. Vehicle routing with stochastic demands: Properties and solution frameworks. *Transportation science*, 23(3):166–176, 1989.

[11] M. Gendreau, O. Jabali, and W. Rei. Stochastic vehicle routing problems. *Vehicle Routing: Problems, Methods, and Applications*, 18:213, 2014.

[12] B. L. Golden, S. Raghavan, and E. A Wasil. *The vehicle routing problem: latest advances and new challenges*, volume 43. Springer Science & Business Media, 2008.

[13] B.L. Golden, L. Levy, and R. Vohra. The orienteering problem. *Naval Research Logistics*, 34(3):307–318, 1987.

[14] K. Karydis, D. Zarrouk, I. Poulakakis, R.S. Fearing, and H.G. Tanner. Planning with the STAR(s). In *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3033–3038, 2014.

[15] A. Kolling and S. Carpin. Extracting surveillance graphs from robot maps. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2323–2328, 2008.

[16] G. Laporte, F. Louveaux, and H. Mercure. The vehicle routing problem with stochastic travel times. *Transportation Science*, 26(3):161–170, 1992.

[17] G.L. Nemhauser, L.A. Wolsey, and M.L. Fisher. An analysis of approximations for maximizing submodular set functions - I. *Mathematical Programming*, 14:265–294, 1978.

[18] A. Nemirovski and A. Shapiro. Convex approximations of chance constrained programs. *SIAM Journal on Optimization*, 17(4):969–996, 2006.

[19] P.M. Pardalos and L.S. Pitsoulis, editors. *Nonlinear assignment problems*. Kluwer Academic Publishing, 2000.

[20] S. Thrun, W. Burgard, and D. Fox. *Probabilistic Robotics*. MIT Press, 2006.

[21] P. Tokekar and V. Kumar. Visibility-based persistent monitoring with robot teams. In *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3387 – 3394, 2015.