Explorative Probabilistic Planning with Unknown Target Locations

Farhad Nawaz[1] and Melkior Ornik[2]

***Abstract*— Motion planning in an unknown environment demands synthesis of an optimal control policy that balances between exploration and exploitation. In this paper, we present the environment as a labeled graph where the labels of states are initially unknown, and consider a motion planning objective to fulfill a generalized reach-avoid specification given on these labels in minimum time. By describing the record of visited labels as an automaton, we translate our problem to a Canadian traveler problem on an adapted state space. We propose a strategy that enables the agent to perform its task by exploiting possible a priori knowledge about the labels and the environment and incrementally revealing the environment online. Namely, the agent plans, follows, and replans the optimal path by assigning edge weights that balance between exploration and exploitation, given the current knowledge of the environment. We illustrate our strategy on the setting of an agent operating on a two-dimensional grid environment.**

I. INTRODUCTION

This paper tackles the classical problem of *reach-avoid planning*: construction of an optimal path to reach a target while avoiding obstacles [1], [2]. A significant amount of recent research, motivated by autonomous missions in remote or hostile areas, focuses on solving it in uncertain and dynamic environments [3], [4]. Our paper aims to solve a generalization of such a problem — encoded as a linear temporal logic (LTL) specification [5] — in an environment where the agent's dynamics are known and simple, but the locations of the targets and obstacles are unknown until an agent reaches their vicinity. Such a problem naturally arises in scientific or military missions where an area of interest has not been previously mapped [6].

The problem considered in our work draws from two areas of prior research: (i) automated synthesis of policies for high-level mission specifications and (ii) optimal planning in uncertain environments. Various tools such as NuSMV [7] and Spin [8] have been developed in the formal verification community to check whether an agent's path satisfies a LTL specification. Though past work [9], [10] utilizes LTL specifications for path planning, it focuses on a priori known state space, solely verifying whether a given path satisfies a specification, and not operating in an uncertain environment.

In the direction of planning in uncertain environments, a common approach [11] is to produce online planning policies which combine *exploration* — learning more about the environment for the sake of success later in the mission — with *exploitation*, i.e., taking actions that seem to contribute towards completing the mission. With notable exceptions of [12] and [13], such work primarily focuses on exploring the environment to learn unknown dynamics, often in a Markov decision process environment [11], [14]. However, our work seeks to locate unknown states of interest, possibly under some prior structural information about their relative locations (tactical formation of military units [15]).

The setting of our work is most similar to [12] and [13]; the latter work considers a reach specification, but just for one particular environment modeling the game of Battleship, where the agent always takes greedy actions for mission completion with no active exploration. The former work jointly considers progress towards an LTL specification and motion of the agent in a *product transition system* similar to our own, but is not able to make use of any prior knowledge about the environment. The planning procedure in [12] consists of multiple motion phases used alternately for exploration and exploitation. We use possible prior information about the environment to interpret the resulting problem as an instance of a Canadian Traveler Problem (CTP), and utilize a more sophisticated planning procedure that keeps track of the probability distribution on all possible environment configurations and seeks to optimally *concurrently* explore the environment and progress towards mission completion.

As in [12], our work rests on interpreting an LTL specification as an automaton [16]. The unknown locations of the states of interest, interpreted as an unknown labeling function of the states, thus result in unknown automaton transitions. We then use tools from automaton theory [16] to construct a product transition system, with deterministic but a priori partly unknown transitions, representing the LTL specification as well as the agent's environment, where transitions are deterministic. Unlike the setting of [12], allowing for a prior distribution on possible labelings now results in an instance of a CTP [17], with additional structure stemming from the construction of an underlying graph as a product space of an automaton and a transition system. The specific structure of the problem allows us to propose a novel approach of solving it tailored to this structure, as opposed to generic heuristic solutions to the CTP [18], [19]. Our approach relies on assigning weights to each possibly existing edge depending on their value for the agent's exploration and exploitation, subsequently planning and replanning — upon gaining new information about the environment — an optimal path towards completing the mission. We first proceed to define the preliminary mathematical concepts used in this paper.

* This work was supported by an Early Stage Innovations grant from NASA's Space Technology Research Grants Program, grant no. 80NSSC19K0209.

[1] F. Nawaz is with the Department of Aerospace Engineering and the Coordinated Science Laboratory, University of Illinois at Urbana-Champaign, Urbana, IL 61801, USA. Email: fns3@illinois.edu

[2] M. Ornik is with the Department of Aerospace Engineering and the Coordinated Science Laboratory, University of Illinois at Urbana-Champaign, Urbana, IL 61801, USA. Email: mornik@illinois.edu

II. Preliminaries and Notation

We introduce some definitions to formally describe the environment and objectives considered in the paper. Throughout the paper, notation 2^X indicates all subsets of a set X.

A. Transition System

A transition system [16] describes the motion of an agent operating in a finite state space. We provide two equivalent definitions, allowing us to switch seamlessly between a formal transition system [12] and a labeled directed graph.

Definition 1. *A* deterministic finite transition system *is a tuple $\mathcal{T} = (S, s_0, Act, \Delta, AP, L)$, where S is a finite set of states, $s_0 \in S$ is the initial state, Act is a finite set of actions, $\Delta \subseteq S \times Act \times S$ is the set of transitions, AP is a set of finite atomic propositions (labels), and $L : S \to 2^{AP}$ is a labeling function.*

We assume that all actions in Act are available at all states $s \in S$. In general, we could define $Act : S \to 2^A$, where A is the set of all actions that are available at some state in S.

A transition relation from state s to s' with action α can be written as $s \xrightarrow{\alpha} s'$ or $(s, \alpha, s') \in \Delta$. The labeling function L assigns each state a set of atomic propositions, $L(s)$, which are considered to be *true* in that state. A *finite path* of a transition system is a finite sequence $\pi = s_0 s_1 ... s_n$, where $s_k \xrightarrow{\alpha_k} s_{k+1} \ \forall\ 0 \leq k \leq n-1$, $\alpha_k \in Act$. The finite path π generates a finite trace $trace(\pi) = l_0 l_1 ... l_n$, where $l_k = L(s_k) \ \forall\ 0 \leq k \leq n$.

We define the labeled directed graph for the deterministic transition system $\mathcal{T}$ as $\mathcal{G} = (S, \Delta_E, AP, L)$, with vertices S and edges $\Delta_E \subseteq S \times S$, where L is the same labeling function as in $\mathcal{T}$. The labels describe regions of interest in the environment. A set of neighbouring vertices for a vertex $s \in S$ is defined as $S'_s = \{s' \in S \mid (s, s') \in \Delta_E\}$. We now proceed to formalize the objective of the agent.

B. Linear Temporal Logic

Linear temporal logic (LTL) [16] is a formal language used to express high level task specifications for the agent to satisfy. An *LTL specification* is a formula defined over a set of atomic propositions, with logical connectors and temporal modal operators. The atomic propositions take Boolean values, either *true* or *false*. The logical connectors come from extending propositional logic, such as negation, disjunction, and conjunction, denoted by $\neg$, $\vee$, and $\wedge$, respectively. The temporal modal operators describe modalities related to time. The basic operators are *next*, *until*, *eventually*, and *always*, denoted by $\bigcirc$, $\mathcal{U}$, $\Diamond$, and $\Box$, respectively. For instance, the specification — which will become our running example — of "eventually visit T_1 and T_2 while avoiding O" can be translated to the LTL specification: $\Diamond T_1 \wedge \Diamond T_2 \wedge \Box \neg O$.

Since the labels of states in a transition system are sets of atomic propositions, the meaning of a path from Section II-A satisfying an LTL formula φ is natural. Formally, we write $trace(\pi) \models \varphi$, which means that φ evaluates as *true* with respect to $trace(\pi)$ [16]. We proceed to encode an agent's progress at its specification using the notion of an automaton.

C. Automaton

A deterministic finite automaton gives a formal way of describing the "history" of a path with respect to an LTL specification that it satisfies. The automaton for our running example is given in Fig. 1a. Every LTL formula defined over a finite set of atomic propositions can be described by a deterministic finite automaton [16].

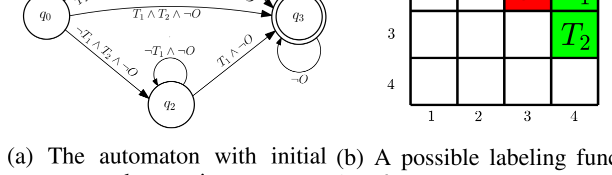
Definition 2. *A* deterministic finite automaton *(DFA) is a tuple $\mathcal{A} = (Q, q_0, \Sigma, \delta, F)$, where Q is a finite set of states, $q_0 \in Q$ is the initial state, $F \subseteq Q$ is a set of accepting states, Σ is the input alphabet, and $\delta : Q \times \Sigma \to Q$ is the transition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

We define a transition in an automaton as $q \xrightarrow{\sigma} q'$ if $q' = \delta(q, \sigma)$, where $\sigma \in \Sigma$ is an atomic proposition. All transitions which violate the LTL formula are not present in the automaton, as in Fig. 1a. An *accepting run* $\pi_{\mathcal{A}}$ of a DFA on a finite sequence of sets of atomic propositions $\sigma_0 \sigma_1 ... \sigma_n$ over Σ is a sequence of states $q_0 q_1 ... q_{n+1}$ such that $q_{n+1} \in F$, and $q_{k+1} = \delta(q_k, \sigma_k) \ \forall\ 0 \leq k \leq n$. Combining these notions with Section II-A, we use an automaton to describe an agent's satisfaction of its LTL specification. The transition relation δ describes the state transitions in $\mathcal{A}$ as the agent transitions from one state in S to another, with a new set of labels. Hence, each set $L(s)$ in $\mathcal{T}$ is an input Σ for $\mathcal{A}$. As in Fig. 1a, a transition $q_0 \to q_1$ at time t indicates that $T_1 \in l_t$, with $T_1 \notin l_{\bar{t}} \ \forall\ \bar{t} < t$ and $T_2, O \notin l_t$. It thus makes natural sense to consider an agent's motion in the transition system $\mathcal{T}$ and its "motion" in the DFA $\mathcal{A}$ in parallel.

D. Product Transition System

A product transition system [16] combines a transition system with an automaton.

Definition 3. *Given a deterministic finite transition system $\mathcal{T} = (S, s_0, Act, \Delta, AP, L)$ and a DFA $\mathcal{A} = (Q, q_0, \Sigma, \delta, F)$, their* product transition system *$\mathcal{T} \otimes \mathcal{A}$ is defined as the following transition system: $\mathcal{T} \otimes \mathcal{A} = (Q_p, Act, \Delta_p, Q_{p0}, F_p)$, where, $Q_p = S \times Q$ is the set of states, $\Delta_p \subseteq Q_p \times Q_p$ where $(s, q) \to (s', q')$ if $s \xrightarrow{\alpha} s'$ and $q' = \delta(q, L(s'))$, $Q_{p0} = \{(s_0, q) | q = \delta(q_0, L(s_0))\}$ is the set of initial states, $F_p = S \times F$ is the set of final states.*


$\neg T_1 \wedge \neg T_2 \wedge \neg O$
q_0
q_1
$T_1 \wedge \neg T_2 \wedge \neg O$
$\neg T_2 \wedge \neg O$
$T_2 \wedge \neg O$
$T_1 \wedge T_2 \wedge \neg O$
q_3
$\neg T_1 \wedge T_2 \wedge \neg O$
$\neg T_1 \wedge \neg O$
$T_1 \wedge \neg O$
$\neg O$
q_2
1
2
3
4
O
T_1
T_2
1
2
3
4

(a) The automaton with initial state q_0, and accepting state q_3. (b) A possible labeling function $l \in \mathcal{L}$, on a grid-world.

Fig. 1: Automaton and a possible labeling function for our running example: $\Diamond T_1 \wedge \Diamond T_2 \wedge \Box \neg O$.

Q_{p0} is a set of pairs of (i) the initial state of $\mathcal{T}$ and (ii) the state of $\mathcal{A}$ which is connected to the initial state of $\mathcal{A}$, with the input as the label of the initial state of $\mathcal{T}$. The transition relation Δ_p defines that a transition between the states of $\mathcal{T}$ and the states of $\mathcal{A}$ exists if and only if (i) there exists a valid transition between the current and the successor state in $\mathcal{T}$ and (ii) label of the successor state in $\mathcal{T}$ is in the input alphabet for a valid transition in $\mathcal{A}$.

Analogous to a transition system $\mathcal{T}$, we can interpret a product transition system as a directed graph. Having defined the relevant preliminaries, we proceed to formally define the optimal planning problem with unknown label locations.

III. Problem Formulation

We consider an agent operating on a labeled directed graph (deterministic transition system) $\mathcal{G} = (S, \Delta_E, AP, L)$. At every time step t, the agent transitions from its current state s_t to a state s_{t+1} such that $(s_t, s_{t+1}) \in \Delta_E$. To formally pose our problem, we make the following assumptions about the agent's knowledge as it moves on $\mathcal{G}$:

1) graph vertices and edges are a priori known to the agent,
2) set AP of possible labels is a priori known to the agent, but not the labeling function L
3) the agent knows its state s_t at every time t,
4) at time t, the agent can see (and memorize for future use) the labels of state s_t and all its neighboring states s_t'.

While the agent does not know exactly the labeling function L, it may have *some* prior information about L. In other words, it has a probability distribution $P : \mathcal{L} \to [0,1]$ over the finite set $\mathcal{L} = \{\overline{L} : S \to 2^{AP}\}$. If there is no knowledge whatsoever, we may consider P to be a uniform distribution.

We consider the following problem statement.

Problem 1. *Let an agent operate on a labeled directed graph $\mathcal{G}$, with $AP = \{T_{ij}\} \cup \{O_i\}$,* where $1 \leq i \leq N, 1 \leq j \leq p_i$, *$N \geq 1$, and p_i are positive integers. Under assumptions 1)-4), find a path π which satisfies a given LTL specification*

$$\varphi = \vee_{i=1}^{N} \left(\left(\wedge_{j=1}^{p_i} \Diamond T_{ij} \right) \wedge (\Box \neg O_i) \right) \quad (1)$$

in minimum time, where T_{ij} are the labels of the target locations, and O_i are the labels of the obstacle locations.

In less formal terms, the LTL specification (1) requires the agent to satisfy at least one of the N specifications $\varphi_i = \left(\left(\wedge_{j=1}^{p_i} \Diamond T_{ij} \right) \wedge (\Box \neg O_i) \right)$. Each specification φ_i is a reach-avoid problem [2], where the agent should eventually visit at least one state labeled with T_{ij} for every j, while avoiding states labeled with O_i. Our method can handle more general co-safe LTL specifications; we present the problem in the current fashion because of its obvious motivational scenario of seeking targets with unknown locations.

Note that our running example in Fig. 1 is an instance of the specification (1) for $N = 1$ and $p_i = 2$. We introduce a grid-world environment given in Fig. 1b as the transition system for our running example. A prior probability distribution on $\mathcal{L}$ may come, for instance, from the knowledge that the pairwise distance between the targets and obstacle is always less than 2 units: any labeling function L that do not satisfy such a property has probability 0, while we consider all other labeling functions to be equally probable.

Throughout the paper, we assume that there indeed exists a path π satisfying φ. Nonetheless, given that the agent does not a priori know the labels of states, it is naturally impossible for it to immediately find a π. However, given assumptions 3) and 4), the agent can "explore" the environment and continually re-plan its path during its mission. Our approach is to adapt the methodology of previous work [11], [14] on optimal planning in unknown environments and try to find an ideal mixture of active exploration — movement for the sake of finding out more about the environment — and progress towards mission completion (exploitation), given the current knowledge about the environment. Our approach differs from [12] in a similar setting; [12] explicitly switches between phases of exploration and exploitation, but not attempting to explore and exploit at the same time might not be maximally time-effective, and [12] does not make use of possible prior knowledge about L.

With notable exception of [12], existing work on planning for unknown environments largely considers unknown system dynamics rather than unknown target locations. The first step of our approach is to consider the motion on a product transition system as described in Section II-D, instead of the previously considered class of problems, which considers motion solely on the transition system $\mathcal{T}$.

IV. Canadian Traveler Problem

We begin by constructing an automaton $\mathcal{A}$ for the specification φ, as in Section II-C. Consequently, the product transition system $\mathcal{T} \otimes \mathcal{A}$ is constructed as in Section II-D.

Let us consider the transitions existing in $\mathcal{T} \otimes \mathcal{A}$. Each of those transitions is a pair of a fully known state transition $s_t \to s_{t+1}$ in $\mathcal{T}$ and a state transition in $\mathcal{A}$. As described in Section II-C, the state transition in $\mathcal{A}$ depends on the labels of the subsequent element of the agent's path, i.e., on $L(s_{t+1})$. Thus, the agent's path in $\mathcal{T} \otimes \mathcal{A}$ depends on the labeling function L, which, by assumption 2), is a priori unknown. Since transitions in $\mathcal{T} \otimes \mathcal{A}$ are uniquely defined by the choice of labeling L, the probability distribution P described in Section III yields a prior probability distribution on the existence of each possible transition in $\mathcal{T} \otimes \mathcal{A}$. By an abuse of notation, we define $P(\delta_p)$ as the prior probability that the transition δ_p exists in $\mathcal{T} \otimes \mathcal{A}$, where $\delta_p \in \Delta_p$.

We emphasize that the agent's motion in $\mathcal{T} \otimes \mathcal{A}$, given its actions, is deterministic, i.e., for a given action $\alpha \in Act$ and state (s_t, q_t), the agent will always transition to the same state in $\mathcal{T} \otimes \mathcal{A}$. The stochastic element of this problem is solely over whether a particular transition (i.e., an edge in the product graph) exists or not. We thus proved the following:

Theorem 1. *Problem 1 is equivalent to a shortest path problem on $\mathcal{T} \otimes \mathcal{A}$ with edges whose existence is a priori uncertain.*

The shortest path problem on a finite deterministic transition system (directed graph) is a *Canadian traveler problem* (CTP) [17]. The CTP asks to find the shortest path on a directed graph $\mathcal{G} = (V, E)$ with vertices V and edges E, where $v_0 \in V$ is the initial vertex, $V_F \subseteq V$ is a set of final vertices, and E is unknown, but there exists a prior probability distribution over the set $\mathcal{E}$ of possible edge configurations, with $E \in \mathcal{E}$.

Finding an optimal policy that yields the expected shortest path for the CTP, is a provably #P-hard problem [17], and heuristic methods for finding a sub-optimal solution are used instead [19]. In addition to computational issues, a general approach to the CTP might not be useful for our problem, in which graph $\mathcal{G}$ is constructed as the product of $\mathcal{T}$ and $\mathcal{A}$, and thus the edges have a particular sparse structure. In the subsequent section, we propose our own approach to the CTP conscious of the special structure of our problem, and compare it to a standard approach of [19].

V. Exploration and Exploitation

Interpreting assumption 4) in terms of the graph $\mathcal{G} = (V, E)$ derived from $\mathcal{T} \otimes \mathcal{A}$ ensures that, at every time step, the agent is aware of all the edges $e \in E$ which are neighbors of its current state. As the agent visits more states in $\mathcal{T}$, i.e., more vertices in $\mathcal{G}$, it changes its probability distribution over $\mathcal{E}$ by removing the sets which do not fit the information about existence of edges that it collected until then. To solve the CTP, we thus propose the following approach: we compute the best path according to some metric of optimality with a particular assumed set E' of existing edges, and proceed along that path until either we see that $E' \neq E$, or the agent completes its mission. If the model $E' \neq E$, we then build a new model E' consistent with current information about E, re-plan for the new best path, and proceed. Our metric of optimality aims to balance between *exploration* — finding out more about the environment — and *exploitation* — moving in a way that seems to help with completing the mission, given the current knowledge about the environment.

We note that the approach of replanning once a model of existing edges turns out to be incorrect is the same as in [19] for the general CTP; however, as we will discuss later, our two approaches differ significantly in the metric of quality that we assign to a path. A similar method is used in [13], but does not explicitly pose the original problem as a CTP and uses a significantly different metric of quality. The technique of removing impossible edge configurations is used in [18], where two paths are computed at every iteration, one each for exploration and exploitation, while we compute only one path based on exploration and exploitation weights.

Let us briefly describe our method presented in Algorithm 1. Lines 1–5 run at the beginning of agent's mission convert the state space S, transition edges Δ_E, and automaton $\mathcal{A}$ derived from the specification φ into a graph with partly unknown edges E; they use the distribution P over the labeling function L to construct a probability distribution $\hat{P}$ over E. Then, a *model* — graph $\overline{\mathcal{G}}$ — is constructed on which a conventional path planning technique can be implemented.

Lines 6–10 utilize **Path-Plan**, to be defined later, as a metric of path quality, to find the best path on $\overline{\mathcal{G}}$ to a set of final vertices V_f. Lines 11–16 describe the agent's

Algorithm 1 Path Planning with Unknown Target Locations

```
 1: procedure INITIALIZATION($S, s_0, \Delta_E, AP, P, \varphi$)
 2:     Construct Automaton $\mathcal{A}$ for LTL formula $\varphi$
 3:     Construct $(V, v_0, V_f)$ for $\mathcal{T} \otimes \mathcal{A}$
 4:     Construct a distribution $\hat{P}$ for $E$ from $\mathcal{T} \otimes \mathcal{A}$
 5:     $\overline{\mathcal{G}} \leftarrow$ Deterministic-Graph$(V, \hat{P})$
 6: procedure SEARCH-PATH($\overline{\mathcal{G}}$)
 7:     repeat
 8:         $\mathcal{P} \leftarrow \min(\mathcal{P}, \textbf{Path-Plan}(\overline{\mathcal{G}}, v_0, V_f))$
 9:         $v_0 \leftarrow$ TRAVERSE-PATH($\mathcal{P}$)
10:     until $\varphi$ is satisfied
11: procedure TRAVERSE-PATH($\mathcal{P}$)
12:     repeat
13:         Visit next vertex $v$ in $\mathcal{P}$
14:         $\hat{P} \leftarrow$ Update-Possible-Edges$(\hat{P})$
15:         $\overline{\mathcal{G}} \leftarrow$ Deterministic-Graph$(V, \hat{P})$
16:     until $\overline{\mathcal{G}}$ changes or end of path
```

traversal of the environment. At each time step, the agent collects new information about state labels, and changes the possible set of graphs to derive a new model graph. If this model differs from the previously computed model, traversal stops, and the path is re-planned in line 8. As long as the mission is not satisfied, the agent keeps visiting new states, until the environment becomes entirely known. Algorithm 1 thus trivially ensures the following result, paralleling the one obtained in [12] using a different planning method.

Theorem 2. *Algorithm 1 results in the agent eventually satisfying its specification, if the specification is satisfiable.*

We now proceed to describe how functions **Deterministic-Graph** and **Path-Plan** construct a graph $\overline{\mathcal{G}}$ and find an optimal path on it, respectively.

Our first step is to distill a probability distribution over the set of possible graphs into a single "guessed" graph $\overline{\mathcal{G}}$. Naturally, $\overline{\mathcal{G}}$ will likely not equal the correct graph $\mathcal{G}$. Nonetheless, replacing a set of graphs with a single graph allows us to efficiently compute a best path on such a graph.

We construct the graph $\overline{\mathcal{G}}$ as follows. Since vertices V of $\mathcal{G}$ are known, those remain the vertices of $\overline{\mathcal{G}}$. The edges of $\overline{\mathcal{G}}$ are those that have non-zero probability of existence.

We proceed with path planning on $\overline{\mathcal{G}}$. We assign weight $w(e)$ to each edge $e \in \overline{E}$ of $\overline{\mathcal{G}}$ as follows:

$$w(e) = w_1(e) + w_2(e), \tag{2}$$

where w_1 and w_2 are exploration and exploitation weights, respectively. An edge whose traversal may reduce the uncertainty about $\mathcal{G}$ will have a low exploration weight. An edge whose traversal will seemingly push the agent towards a vertex in V_f will have a low exploitation weight.

Assuming e as an edge from $v_1 \in V$ to $v_2 \in V$, we define

$$w_1(e) = 1 + \gamma \frac{\sum_{v' \in V'_{v_2}} \left(1 - \frac{1}{(P(v_2, v') - 0.5)^2 + 1}\right)}{\#(V'_{v_2})}, \tag{3}$$

where γ is a tunable parameter. The set of neighboring vertices for v_2 is V'_{v_2}, and $\#(V'_{v_2})$ is the cardinality of V'_{v_2}.

If v_2 has uncertain adjacent edges, then $w_1(e)$ is small. The existence of each edge $e' = (v_2, v')$ is Bernoulli distributed with probability $\mathbb{P}(e')$. The uncertainty is maximum when $\mathbb{P}(e') = 0.5$, with the lowest value of $1 - 1/((\mathbb{P}(e') - 0.5)^2 + 1)$, and uncertainty is minimum when $\mathbb{P}(e') = 0$ or 1, with the highest value of $1 - 1/((\mathbb{P}(e') - 0.5)^2 + 1)$.

To define the exploitation weight, we introduce the following assumptions, and notation based on our LTL formula (1):

i) equation (1) can be modified to repeat some targets (T_{ij}) or obstacles (O_i) among the N specifications,
ii) a target label in one specification i cannot be an obstacle label in another specification j, and vice-versa,
iii) q_v is the automaton "coordinate" of vertex v in $\overline{\mathcal{G}}$,
iv) $O(q_v)$ and $T(q_v)$ are the total number of visited obstacles and targets, respectively, in automaton state q_v,
v) n_O and n_T are the number of obstacles and targets, respectively, visited by the agent until the current time.

We define $w_2(e)$ as follows:

$$w_2(e) = \begin{cases} 1 \quad \text{if } q_{v_2} = q_{v_1} \\ 1 + (1 + n_O)\mathbb{P}(e) \text{ if } O(q_{v_2}) = O(q_{v_1}) + 1 \\ \frac{(1-\mathbb{P}(e))}{1+n_T} \quad \text{if } T(q_{v_2}) = T(q_{v_1}) + 1 \end{cases} \tag{4}$$

From assumption ii) above, each state in S is either not labeled, a target, or an obstacle. Hence, a state transition will either increase the number of visited obstacles or targets by 1, or have both of those numbers remain same. Thus, the piecewise definition of (4) covers the three possible cases, assuming a state in the transition system can have only one label, or none. We could accordingly add additional conditions to $w_2(e)$ if a state can have multiple labels.

If a state transition does not lead to any change in visited labels, the weight is 1. If vertex v_2 corresponds to an increase in obstacle count from v_1, we increase the weight of e, making it less desirable, depending on the number of obstacles already visited, and the probability of the edge actually existing. If e leads to a vertex with an increase in target count, we analogously reduce the weight of e, making it more desirable.

Function **Path-Plan** now simply implements a shortest path algorithm (Bellman-Ford [20]) on a graph $\overline{\mathcal{G}}$ with weights w defined in (2), (3) and (4). We proceed to validate this approach on a classical grid-world environment.

VI. Numerical Example

We consider an agent that operates on a 8×8 grid-world. Some cells are not labeled; those that are labeled have exactly one label from the set $AP = \{T_1, T_{21}, T_{22}, O_{11}, O_{12}, O_{21}, O_{22}\}$. It is a priori known that there is one cell with each label: cells labeled with $O_{(\cdot)}$ are obstacles, while $T_{(\cdot)}$ are targets. The agent can move to the north, west, south, or east of its current cell, except, at the edges of the environment. Its knowledge about the environment satisfies assumptions 1)-4) above Problem 1 and property ii) before (4).

The actual labeled environment which is unknown to the agent is shown in Fig. 2a. The agent has the following structural information about possible label configurations:

- the obstacles always lie on the corners of a 4×4 square,
- O_{11} is the top left corner of the square, O_{21} top right, O_{22} bottom right, and O_{12} bottom left,
- three targets always lie in the square, including edges,
- the Euclidean distance between the goals is at least 2.

While we recognize that the described structural information is highly specific, information of such type is often available in urban or military planning, where units or facilities may be arranged in a specific formation [21], [22].

The mission of the agent is given as follows: if the agent visits any one of the obstacles, then it has to visit all of the targets T_1, T_{21} and T_{22}. If the agent does not visit an obstacle, then the mission is complete if it visits a state labeled T_1, and at least one of the states labeled T_{21} or T_{22}. This mission can be expressed in the following specification:

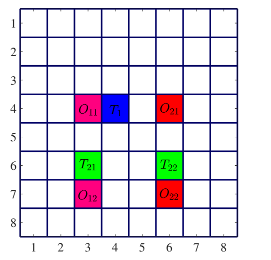
$$\begin{aligned}
&\varphi = \\
&(\Diamond T_1 \wedge \Diamond T_{21} \wedge \neg\Box O_{11} \wedge \neg\Box O_{12} \wedge \neg\Box O_{21} \wedge \neg\Box O_{22}) \vee \\
&(\Diamond T_1 \wedge \Diamond T_{22} \wedge \neg\Box O_{11} \wedge \neg\Box O_{12} \wedge \neg\Box O_{21} \wedge \neg\Box O_{22}) \vee \\
&(\Diamond T_1 \wedge \Diamond T_{21} \wedge \Diamond T_{22} \wedge \neg\Box O_{12} \wedge \neg\Box O_{21} \wedge \neg\Box O_{22}) \vee \\
&(\Diamond T_1 \wedge \Diamond T_{21} \wedge \Diamond T_{22} \wedge \neg\Box O_{11} \wedge \neg\Box O_{21} \wedge \neg\Box O_{22}) \vee \\
&(\Diamond T_1 \wedge \Diamond T_{21} \wedge \Diamond T_{22} \wedge \neg\Box O_{11} \wedge \neg\Box O_{12} \wedge \neg\Box O_{22}) \vee \\
&(\Diamond T_1 \wedge \Diamond T_{21} \wedge \Diamond T_{22} \wedge \neg\Box O_{11} \wedge \neg\Box O_{12} \wedge \neg\Box O_{21}) .
\end{aligned} \tag{5}$$

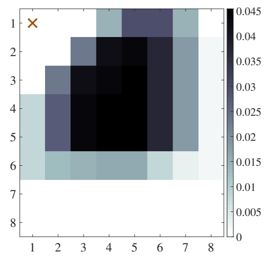
Specification (5) falls within the class of (1), where the agent needs to satisfy at least one of six reach-avoid tasks φ_i; to obtain the form of (1), we can formally combine all obstacles mentioned in task φ_i into a new obstacle label O_i.

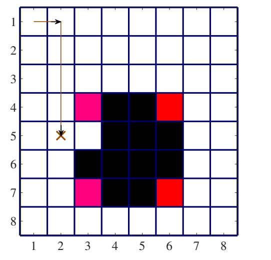
The agent starts at the top left corner of the grid. Assuming that all labelings that satisfy the known structural information are equally probable, Fig. 2b shows the initial probabilities of each cell for label type T_1. For implementation, we choose $\gamma = 5$, resulting in roughly equal scaling of the exploration and exploitation weights. Figs. 2c and 2d present some time instances of the agent's motion according to Algorithm 1 for the LTL specification (5); the agent completes its task, by following the first reach-avoid specification, in 10 time steps. We compare the performance of our algorithm with Westphal's reposition algorithm [19] for the CTP as illustrated in Fig. 3. The algorithm from [19] requires 20 time steps to complete the mission, choosing to complete the fifth reach-avoid specification. We note that, when implementing Westphal's algorithm [19], the agent proceeds to satisfy the mission only once it has discovered exactly where the labels are. In our algorithm, the agent incorporates exploration and exploitation from the start of the mission.

VII. Conclusions

In this paper, we formulated a path planning problem in a labeled environment with unknown label locations, and mission objectives given as generalized reach-avoid specifications in LTL formula. We proposed a solution by constructing a product transition system from the agent's


(a) True environment, a priori unknown to the agent.


(b) A priori probability that a cell is T_1.


(c) Agent's path at $t = 7$.

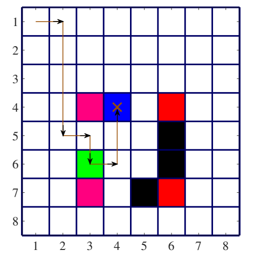

(d) Agent's path at $t = 10$.

Fig. 2: Illustration of Algorithm 1. The brown cross is the agent, while the arrows denote the direction of motion. The labels of black cells are not completely known to the agent, white cells are known to be vacant, while other colored cells denote the discovered label types as in Fig. 2a, at that time.

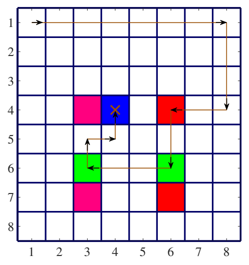


Fig. 3: Agent's path produced by Westphal's algorithm [19], with the same coloring scheme as in Fig. 2a.

transition system and an automaton representing its mission; a probability distribution on label locations which translated into a probability distribution on transitions in the product transition system. The resulting problem is a highly structured instance of a Canadian traveler problem. We proposed to solve this problem by a path planning procedure on a deterministic graph modeling the agent's knowledge about the environment, with graph weights balancing between exploring the environment and exploiting the objective. Finally, we implemented the algorithm for an agent operating on a grid-world with local sensor information and prior structural knowledge about target and obstacle locations.

While this work proposes an approach to solve an optimal planning problem for unknown labels, much remains in terms of providing theoretical guarantees on the optimality of the proposed solution: the current algorithm only guarantees eventual satisfaction of the mission, but there is no developed theory on the time to complete a mission, nor on the optimal choice of graph weights (including the parameter γ). As indicated in previous sections, the considered LTL specifications readily generalize to a wider class of co-safe LTL formulae; however, these formulae can give rise to large automata, resulting in large product transition systems. Future practical work with such automaton thus likely requires automating the conversion into a product environment.

References

[1] S. M. LaValle, *Planning Algorithms*. Cambridge University Press, 2006.

[2] Z. Zhou, J. Ding, H. Huang, R. Takei, and C. Tomlin, "Efficient path planning algorithms in reach-avoid problems," *Automatica*, vol. 89, pp. 28–36, 2018.

[3] N. E. Du Toit and J. W. Burdick, "Robot motion planning in dynamic, uncertain environments," *IEEE Transactions on Robotics*, vol. 28, no. 1, pp. 101–115, 2011.

[4] P. D. Triantafyllou, G. A. Rovithakis, and Z. Doulgeri, "Constrained visual servoing under uncertain dynamics," *International Journal of Control*, vol. 92, no. 9, pp. 2099–2111, 2019.

[5] G. E. Fainekos, H. Kress-Gazit, and G. J. Pappas, "Temporal logic motion planning for mobile robots," in *IEEE International Conference on Robotics and Automation*, pp. 2020–2025, 2005.

[6] S. M. Milkovich, R. Lange, K. Williford, T. L. Wagner, M. Heverly, and M. Ono, "Mars 2020 surface mission performance analysis: Part 1. Science exploration and soil type modeling," in *AIAA SPACE and Astronautics Forum and Exposition*, 2018.

[7] A. Cimatti, E. Clarke, F. Giunchiglia, and M. Roveri, "NuSMV: a new symbolic model checker," vol. 2, pp. 410–425, Springer, 2000.

[8] G. Holzmann, *The SPIN Model Checker: Primer and Reference Manual*. Addison-Wesley, 2003.

[9] E. Vitolo, C. Mahulea, and M. Kloetzer, "Path-planning in Discretized Environments with Optimized Waypoints Computation," in *2018 IEEE 23rd International Conference on Emerging Technologies and Factory Automation (ETFA)*, vol. 1, pp. 729–735, IEEE, 2018.

[10] M. Kloetzer and C. Mahulea, "Path planning for robotic teams based on LTL specifications and Petri net models," *Discrete Event Dynamic Systems*, vol. 30, no. 1, pp. 55–79, 2020.

[11] M. Ornik, J. Fu, N. T. Lauffer, W. K. Perera, M. Alshiekh, M. Ono, and U. Topcu, "Expedited learning in MDPs with side information," in *57th IEEE Conference on Decision and Control*, pp. 1941–1948, 2018.

[12] A. I. Medina Ayala, S. B. Andersson, and C. Belta, "Temporal logic motion planning in unknown environments," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 5279–5284, 2013.

[13] N. Berry. http://www.datagenetics.com/blog/december32011/, 2011.

[14] A. Strehl, L. Li, and M. Littman, "Reinforcement learning in finite MDPs: PAC analysis," *Journal of Machine Learning Research*, vol. 10, pp. 2413–2444, 2009.

[15] J. E. Kline, "A tactical doctrine for distributed lethality," in *Distributed Lethality 2016: A CIMSEC Compendium* (D. Filipoff, M. Merighi, J. Stryker, and S. DeBoer, eds.), pp. 4–9, 2016.

[16] C. Baier and J. Katoen, *Principles of Model Checking*. MIT Press, 2008.

[17] A. Bar-Noy and B. Schieber, "The Canadian Traveller Problem.," in *SODA*, vol. 91, pp. 261–270, 1991.

[18] Z. W. Lim, D. Hsu, and W. S. Lee, "Shortest Path under Uncertainty: Exploration versus Exploitation," in *Conference on Uncertainty in Artificial Intelligence*, 2017.

[19] S. Westphal, "A note on the k-Canadian traveller problem," *Information Processing Letters*, vol. 106, no. 3, pp. 87–89, 2008.

[20] R. Bellman, "On a routing problem," *Quarterly of applied mathematics*, vol. 16, no. 1, pp. 87–90, 1958.

[21] Department of the Army, "Civil disturbance operations," Tech. Rep. FM 3-19.15, 2005.

[22] J. Lang, *Urban Design: The American Experience*. Wiley, 1994.