Journal Title
XX(X):1–19

DOI: 10.1177/ToBeAssigned
www.sagepub.com/
SAGE

Planning to Chronicle: Optimal Policies for Narrative Observation of Unpredictable Events

Hazhar Rahmani[1] and Dylan A. Shell[2] and Jason M. O'Kane[1]

Abstract

One important class of applications entails a robot scrutinizing, monitoring, or recording the evolution of an uncertain time-extended process. This sort of situation leads to an interesting family of active perception problems that can be cast as planning problems in which the robot is limited in what it sees and must, thus, choose what to pay attention to. The distinguishing characteristic of this setting is that the robot has influence over what it captures via its sensors, but exercises no causal authority over the process evolving in the world. As such, the robot's objective is to observe the underlying process and to produce a 'chronicle' of occurrent events, subject to a goal specification of the sorts of event sequences that may be of interest. This paper examines variants of such problems in which the robot aims to collect sets of observations to meet a rich specification of their sequential structure. We study this class of problems by modeling a stochastic process via a variant of a hidden Markov model, and specify the event sequences of interest as a regular language, developing a vocabulary of 'mutators' that enable sophisticated requirements to be expressed. Under different suppositions on the information gleaned about the event model, we formulate and solve different planning problems. The core underlying idea is the construction of a product between the event model and a specification automaton. Using this product, we compute a policy that minimizes the expected number of steps to reach a goal state. We introduce a general algorithm for this problem as well as several more efficient algorithms for important special cases. The paper reports and compares performance metrics by drawing on some small case studies analyzed in depth via simulation. Specifically, we study the effect of the robot's observation model on the average time required for the robot to record a desired story. We also compare our algorithm with a baseline greedy algorithm, showing that our algorithm outperforms the greedy algorithm in terms of the average time to record a desired story. In addition, experiments show that the algorithms tailored to specialized variants of the problem are rather more efficient than the general algorithm.

Keywords

Planning, Story-telling, Reconnoitering, Raconteuring

1 Motivation and Introduction

This paper is about robotic planning problems in which the goals are expressed as time-extended sequences of discrete events whose occurrence the robot cannot causally influence. As a concrete motivation for this sort of setting, consider the proliferation of home videos. These videos are, with remarkably few exceptions, crummy specimens of the cinematic arts. They fail, generally, to establish and then bracket a scene; they often founder in emphasizing the importance of key subjects within the developing action, and are usually unsuccessful in attempts to trace an evolving narrative arc. And the current generation of autonomous personal robots and video drones, in their roles as costly and glorified 'selfie sticks,' are set to follow suit. The trouble is that capturing footage to tell a story is challenging. A camera only records what you point it toward and part of the difficulty stems from the fact that you can't know exactly how the scene will unfold before it actually does. Moreover, what constitutes structure isn't easily summed up with a few trite quantities. Another part of the challenge, of course, is that one has only limited time to capture video footage.

Many applications can be cast as the problem of producing a finite-length sensor-based recording of the evolution of some process. As the video example emphasizes, one might be interested in recordings that meet rich specifications of the event sequences that are of interest. It is easy to look beyond pure vanity as a motivator: consider cases where a robot is auditing or inspecting some occurrence, perhaps to summarize or provide select evidence of some specific property or portray the manifestation of some pattern. When the evolution of the event-generating process is uncertain/non-deterministic and sensing is local (necessitating that it be directed actively), then one encounters an instance from this class of problem. The broad class encompasses many monitoring and surveillance scenarios. An important characteristic of such settings is that the robot has influence over what it captures via its sensors, but cannot control the process of interest.

Our approach to this class of problem involves two lines of attack. The first is a wide-embracing formulation in which we pose a general stochastic model, including aspects of

[1]University of South Carolina, Department of Computer Science and Engineering, US
[2]Texas A&M University, Department of Computer Science and Engineering, US

hidden/latent state, simultaneity of event occurrence, and various assumptions on the form of observability. Secondly, we specify the sequences of interest via a deterministic finite automaton (DFA), and we define several language mutators, which permit composition and refinement of specification DFAs, allowing for rich descriptions of desirable event sequences. The two parts are brought together via our approach to planning to satisfy the specifications as quickly as possible: we show how to compute an optimal policy via a form of product automaton. Empirical evidence from simulation experiments attests to the feasibility of this approach.

Beyond the pragmatics of planning, a theoretical contribution of the paper is to prove a result on representation independence of the specifications. That is, though multiple distinct DFAs may express the same regular language and despite the specific DFA being involved directly in constructing the product automaton used to solve the planning problem, it turns out that it is the language expressed, not how it is expressed, that affects the resulting optimal solution. This means that one can reduce the automaton size (say, using the famous Myhill-Nerode result) to reduce the planning time and know that this does not reduce the quality of the solution produced. Returning to mutators that transform DFAs, enabling easy expression of sophisticated requirements, we distinguish when mutators preserve representational independence too.

A preliminary version of this paper (Rahmani et al., 2020) appeared in the 14th International Workshop on the Algorithmic Foundations of Robotics (WAFR-XIV). The following results are new to this version:

- We generalize the event model, introducing an occurrence probability for each state-event pair.
- We address the case in which no policy can guarantee that the robot will successfully capture a desired story.
- We discuss several special cases, showing how they can be solved more efficiently than the general problem.
- We added new case studies comparing our general approach to both a greedy baseline and to our specialized algorithms for the newly-introduced special cases.

2 Related Work

In research closely related to the present article, the work of Shell et al. (2019) (having some authors in common with the present paper) introduces the idea of using a team of autonomous robots, coordinated by a planner, to capture a sequence of events meeting some narrative structure. That work raised (but did not answer) several questions, among them: how the robot can formulate effective plans to capture events relevant a given story specification. Here we build upon that prior effort showing how such plans can be formed in a principled way.

Also related is work in the area of autonomous cinematography, which has recently become an active research topic in robotics. Mademlis et al. (2019a) focus on vision-based UAVs (Unmanned Aerial Vehicles), and specifically multi-UAV, cinematography. They review the research topic and make a taxonomy of shot types based on framing types (long short, close-up, etc.) and camera motion types (orbit, fly-by, etc.), which they extend for multi-UAV cinematography. A more comprehensive version of their taxonomy, including several ones for target tracking, can be found in (Mademlis et al., 2019b). Sabetghadam et al. (2019) propose for autonomous UAV cinematography, an optimal trajectory planning algorithm in which the control of the drone and the control of the gimbal are decoupled. They also consider in (Alcántara et al., 2021), optimal trajectory planning for multi-UAV cinematography. All these works assume that the robot has already been assigned what it is to capture, while in our work it is the robot that needs to predict which events will occur and which it should try to capture.

Treatment of narrative in terms of structured sequences of discrete events, as done in this article, is not unique. The *story validation problem* (Yu and LaValle, 2010, 2011) can be viewed as an inverse of our problem. The aim there is to determine whether a given story is consistent with a sequence of events captured by a network of sensors in the environment. In our problem, it is the robot that needs to capture a sequence of events that constitute a desired story.

Video summarization is the problem of making a 'good' summary of a given video by prioritizing sequences of frames based on some selection criterion (importance, representativeness, diversity, etc.). Various approaches include identifying important objects (Lee et al., 2012), finding interesting events (Gygli et al., 2014), selection using supervised learning (Gong et al., 2014), and finding inter-frame connections (Lu and Grauman, 2013). For a survey on video summarization see (Truong and Venkatesh, 2007), which one might augment with more recent results (Mahasseni et al., 2017; Plummer et al., 2017; Zhang et al., 2018; Ji et al., 2019; Narasimhan et al., 2021).

Specifically with robot-based observation in mind, Girdhar and Dudek (2012) considered the related *vacation snapshot problem* in which the goal is to retain a diverse subset from data observed by a mobile robot. However, in such summarization techniques, the problem is essentially to post-process a collection of images already recorded. This paper, by contrast, addresses the problem of deciding which video segments the robot should attempt to capture in the first place.

Also, for text-based and interactive narratives, a variety of methods are known for narrative planning and generating natural language stories (Riedl and Young, 2010; Robertson and Young, 2017).

Our work builds on the theory of Markov decision processes (MDPs) and its extension to partially observable Markov decision processes (POMDPs), which are surveyed in (LaValle, 2006; Shani et al., 2013; Ross et al., 2008; Bonet and Geffner, 2009). In what follows, we tackle our problem by constructing a product of the event model and the story specification, which together yield a specific POMDP.

Our own previous work (Chaudhuri et al., 2021b) considered a multi-robot extension of the problem we study here. In that work, we compared, in terms of quality of solution and efficiency, a planning approach in which planning for all of the robots is conducted jointly against a sequential approach. We also studied an extension in which the planning process comprises not only to what events to try to capture but also which recording styles the robot

should use to capture events those events, ensuring that the final result conforms to a given specification of acceptable styles (Chaudhuri et al., 2021a).

The empirical results we shall present show that as the robot's power to perceive the world increases, the robot can record events that form a desirable narrative more quickly. As such, the ideas of active perception (Bajcsy, 1988; Bajcsy et al., 2018) could be useful for the robot to strategically increase its knowledge about the world and the events happening around it; of course, the present work also constitutes a planned form of active perception itself.

3 The Problem

To begin, we introduce the problem formalization, starting with the most basic elements of the model.

3.1 Events and observations

The essential objects of interest are *events*, that is, atomic occurrences situated at specific times and places. We treat each event as a letter drawn from a finite alphabet E, a set which contains all possible events. Any finite sequence of events, in particular a *story* ξ the robot wants to record from the events that occur in the system, is a word in E^*. (The set of finite sequences is written here using the Kleene star.)

We model the occurrence of events using a structure defined as follows.

Definition 1. [event model] An *event model* $\mathcal{M} = (S, \mathbf{P}, s_0, E, g)$ is a tuple in which

- S, a nonempty finite set, is the *state space* of the model;
- $\mathbf{P} : S \times S \to [0, 1]$ is the *transition probability function* of the model, such that for each state $s \in S$, $\sum_{s' \in S} \mathbf{P}(s, s') = 1$;
- $s_0 \in S$ is the *initial state*;
- E is the set of all possible events;
- $g : S \times E \to [0, 1]$ is an *event 'going-on'* function defined so that for state s and event e, value $g(s, e)$ is the probability that event e happens at state s. We assume that for each $e \in E$, $g(s_0, e) = 0$.

An execution of the model starts from the initial state s_0 and then, at each time step k, the system makes a transition from state s_k to state s_{k+1}, the latter being chosen randomly based on $\mathbf{P}$ from those states for which $\mathbf{P}(s_k, \cdot) > 0$. This execution specifies a path $s_0 s_1 \cdots$. For every time step k, when the system enters state s_k, some (possibly empty) set of events occurs simultaneously, each event $e \in E$ occurring with probability $g(s_k, e)$, independently from other events.

We are interested in scenarios in which a robot is tasked with recording certain sequences of events. We model the state of the event model as only partially observable to the robot. That is, the current state s_k of the event model is hidden from the robot, but the system instead emits an output observable to the robot at each time step. The next definition formalizes the idea.

Definition 2. [observation model] For a given event model $\mathcal{M} = (S, \mathbf{P}, s_0, E, g)$, an *observation model* $\mathcal{B} = (Y, h)$ is a pair in which

- Y is a nonempty set of *observations* or outputs;
- $h : S \times Y \to [0, 1]$ is the *emission probability function* of the model, such that for each state $s \in S$, $\sum_{y \in Y} h(s, y) = 1$.

At each time step, when the system enters a state s_k, it emits an output y_k, drawn according to $h(s_k, \cdot)$. The emitted output y_k is observable to the robot. An event model and observation model can be depicted together as a directed graph (e.g., see Figure 1a), where we show each state's events as an attached set (in braces in the figure) and display observations from Y along with their emission probabilities (in brackets). We consider, as important special cases, two particular types of observation models.

Definition 3. Given an event model $\mathcal{M} = (S, \mathbf{P}, s_0, E, g)$ with observation model $\mathcal{B} = (Y, h)$, we say that $\mathcal{B}$ makes $\mathcal{M}$ *fully observable* if (1) $Y = S$, and (2) $h(s, y) = 1$ if and only if $s = y$.

We write $\mathcal{B}_{obs}(\mathcal{M})$ to denote the unique observation model that makes $\mathcal{M}$ fully observable. At the other extreme, another special event model is one in which the emitted outputs do not help at all to reduce uncertainty.

Definition 4. Given an event model $\mathcal{M} = (S, \mathbf{P}, s_0, E, g)$ with observation model $\mathcal{B} = (Y, h)$, then $\mathcal{B}$ causes the event model to be *fully hidden* if the observation space Y is a singleton set.

Since the particular single observation comprising Y is unimportant, by $\mathcal{B}_{hid}(\mathcal{M})$ we denote some observation model making $\mathcal{M}$ fully hidden.

3.2 Story specifications, belief states, and policies

As the system evolves along a path $s_0 s_1 s_2 \cdots$, the robot attempts to record some of the events that actually occur in the world to form a story $\xi \in E^*$. We specify the desired story using a deterministic finite automaton (DFA) $\mathcal{D} = (Q, E, \delta, q_0, F)$, where Q is its state space, E is its alphabet, $\delta : Q \times E \to Q$ is its transition function, q_0 its initial state, and $F \subseteq Q$ is the set of all final (accepting) states of the automaton. In other words, we want the robot to make a story ξ in the language of $\mathcal{D}$, denoted $\mathcal{L}(\mathcal{D})$, which is the set of all strings in E^* that when are tracked from q_0, the automaton reaches an accepting state.

The semantics of event capture are as follows. At each step $k \geq 0$, the robot chooses one event e from E to attempt to record in the next step, $k + 1$. If any of the actual events that do happen at step $k + 1$ (an event e can happen at s_{k+1} only if $g(s_{k+1}, e) > 0$) match the robot's prediction, then the robot successfully records this event; otherwise, it records nothing. The robot is aware of the success or failure of each of its attempts. The robot stops making guesses and observations once it has recorded a desired story—a story in $\mathcal{L}(\mathcal{D})$.

To estimate the current state, the robot maintains, at each time step k, a belief state $b_k : S \to [0, 1]$, in which $\sum_{s \in S} b_k(s) = 1$. For each $s \in S$, $b_k(s)$ represents the probability that the event model is in state s at time step k,

according to the information available to the robot, including both the observations emitted directly by the event model, and the sequence of successes or failures in recording events. It also maintains, for each time k, the sequence ξ_k of events it has recorded until time step k, and the (unique) DFA state q_k obtained by ξ_k.

The robot's predictions are governed by a policy $\pi : \Delta(S) \times Q \to E$ that depends on the belief state and the state of the DFA. At time step $k+1$, the robot may append a recorded event to ξ_k via the following formula:

$$\xi_{k+1} = \begin{cases} \xi_k \pi(b_k, q_k) & \pi(b_k, q_k) \text{ happened at } s_{k+1} \\ \xi_k & \pi(b_k, q_k) \text{ did not happen at } s_{k+1}. \end{cases} \tag{1}$$

The initial condition is that $\xi_0 = \epsilon$, in which ϵ is the empty string. The robot changes the value of variable q_k only when the guessed event actually happened:

$$q_{k+1} = \begin{cases} \delta(q_k, \pi(b_k, q_k)) & \pi(b_k, q_k) \text{ happened at } s_{k+1} \\ q_k & \pi(b_k, q_k) \text{ did not happen at } s_{k+1}. \end{cases} \tag{2}$$

The robot stops when $q_k \in F$.

3.3 Optimal recording problems

The robot's goal is to record a story (or video) as quickly as possible. We consider this problem in three different settings: a general setting without any restriction on the observation model, a setting in which the observation model makes the event model fully observable, and a final one in which the event model becomes fully hidden.

Definition 5. [correct policy] For a given event set E, event model $\mathcal{M} = (S, \mathbf{P}, s_0, E, g)$ with observation model $\mathcal{B} = (Y, h)$, DFA $\mathcal{D} = (Q, E, \delta, q_0, F)$, and policy $\pi : \Delta(S) \times Q \to E$, we say π is a *correct policy* if it ensures, with probability 1, that a story within $\mathcal{L}(\mathcal{D})$ will eventually be captured.

First, the general setting.

Problem: Recording Time Minimization (RTM)

Input: An event set E, an event model $\mathcal{M} = (S, \mathbf{P}, s_0, E, g)$ with observation model $\mathcal{B} = (Y, h)$, and a DFA $\mathcal{D} = (Q, E, \delta, q_0, F)$.

Output: A correct policy that minimizes the expected number of steps k until $\xi_k \in \mathcal{L}(\mathcal{D})$; or 'NO SOLUTION' if no correct policy exists.

Note that k is not necessarily the length of the resulting story ξ_k, but rather is the number of steps the system runs to capture that story. In fact, since the robot captures at most one event in each time step, $|\xi_k| \leq k$.

The second setting constrains the system to be fully observable.

Problem: RTM with Fully Observable Model (RTM/FOM)

Input: An event set E, an event model $\mathcal{M} = (S, \mathbf{P}, s_0, E, g)$, and a DFA $\mathcal{D} = (Q, E, \delta, q_0, F)$.

Output: A correct policy that, under observation model $\mathcal{B}_{obs}(\mathcal{M})$, minimizes the expected number of steps k until $\xi_k \in \mathcal{L}(\mathcal{D})$; or 'NO SOLUTION' if no correct policy exists.

In this setting, because states are fully observable to the robot, we might have defined the policy as a function over $S \times Q$ rather than over $\Delta(S) \times Q$. Nonetheless, our current definition does not pose any problem. Any reachable belief state in this setting considers only a *single* outcome (i.e., given any k, $b_k(s) = 1$ for exactly one $s \in S$) and thus, we are interested in the optimal policy only for those reachable beliefs.

The third setting assumes a fully hidden event model state.

Problem: RTM with Fully Hidden Model (RTM/FHM)

Input: An event set E, an event model $\mathcal{M} = (S, \mathbf{P}, s_0, E, g)$, and a DFA $\mathcal{D} = (Q, E, \delta, q_0, F)$.

Output: A correct policy that, under observation model $\mathcal{B}_{hid}(\mathcal{M})$, minimizes the expected number of steps k until $\xi_k \in \mathcal{L}(\mathcal{D})$; or 'NO SOLUTION' if no correct policy exists.

4 Algorithm Description

Next we give an algorithm for RTM, which also solves RTM/FOM and RTM/FHM, essentially special cases of RTM.

4.1 The Goal POMDP

The first step of the algorithm constructs a specific partially observable Markov decision process (POMDP), which we term the Goal POMDP, as follows:

Definition 6. [Goal POMDP] For an event model $\mathcal{M} = (S, \mathbf{P}, s_0, E, g)$ with observation model $\mathcal{B} = (Y, h)$, and a DFA $\mathcal{D} = (Q, E, \delta, q_0, F)$, the associated *Goal POMDP* is a tuple $\mathcal{P}_{(\mathcal{M},\mathcal{B};\mathcal{D})} = (X, A, b_0, \mathbf{T}, X_G, Z, \mathbf{O}, c)$, in which

1. $X = S \times Q$ is the state space;
2. $A = E$ is the action space;
3. $b_0 \in \Delta(X)$ is the initial belief state, in which $b_0(x) = 1$ if and only if $x = (s_0, q_0)$;
4. $\mathbf{T} : X \times A \times X \to [0, 1]$ is the transition probability function such that, assuming $\mathbb{1}_{\{A\}}(\cdot)$ to be set A's indicator function, for each $e \in E$ and $(s, q), (s', q') \in X$,

$$\mathbf{T}((s,q),e,(s',q')) = \begin{cases} \mathbf{P}(s,s')\cdot g(s',e) & \text{if } q \notin F, q' = \delta(q,e), \\ & \text{and } q' \neq q \quad (4.a) \\ \mathbf{P}(s,s')\cdot g(s',e)\cdot & \\ \quad \mathbb{1}_{\{q'\}}(\delta(q,e)) + & \text{if } q \notin F, \\ \mathbf{P}(s,s')\cdot(1-g(s',e)) & \text{and } q' = q \quad (4.b) \\ 1 & \text{if } q \in F, q' = q, \\ & \text{and } s = s' \quad (4.d) \\ 0 & \text{otherwise;} \end{cases}$$

5. $X_G = S \times F$ is the set of goal states;
6. $Z = (\{\text{True}, \text{False}\} \times Y) \cup \{\perp\}$, in which $\perp$ is used for the observation that the robot has completed recording a desired story, is the set of observations;
7. $\mathbf{O} : A \times X \times Z \to [0,1]$ is the observation probability function such that for each $e \in E$, $s \in S$, $q \in Q$, and $y \in Y$:
   (a) $\mathbf{O}(e,(s,q),(\text{True},y)) = h(s,y)\cdot g(s,e)$ if $q \notin F$,
   (b) $\mathbf{O}(e,(s,q),(\text{False},y)) = h(s,y)\cdot(1-g(s,e))$ if $q \notin F$,
   (c) $\mathbf{O}(e,(s,q),\perp) = 1$ if $q \in F$;
8. $c : X \times A \to \mathbb{R}_{\geq 0}$ is the cost function such that for each $x \in X$ and $a \in A$, $c(x,a) = 1$ if $x \notin X_G$, and $c(x,a) = 0$ otherwise.

Figure 1 illustrates this construction for an elementary example. Each state of this POMDP is a pair (s,q) indicating the situation where, under an execution of the system, the current state of the event model is s and the current state of the DFA is q. For each $x, x' \in X$ and $a \in A$, $\mathbf{T}(x,a,x')$ gives the probability of transitioning from state x to state x' under performance of action a. In the context of our event model, each transition corresponds to a situation where the robot chooses an event e to observe and the event model makes a transition from a state s to s'. If e happens at s' and $\delta(q,e) \neq q$, then the robot records e and then changes the current state of the DFA to $\delta(q,e)$; otherwise, it does nothing and the DFA remains in state q. These correspond to cases (4.a) and (4.b) above, respectively. Note that the term $\mathbf{P}(s,s')\cdot g(s',e)\cdot \mathbb{1}_{\{q'\}}(\delta(q,e))$ in case (4.b) corresponds to the situation where $\delta(q,e) = q$ and the predicted event e happens at s', while term $\mathbf{P}(s,s')\cdot(1-g(s',e))$ corresponds to the situation where the predicted event e, regardless of whether $\delta(q,e) = q$ or not, does not happen at s'. Case (4.c) makes all the goal states of the POMDP *absorbing* states. The goal states of the POMDP are those in which the robot has recorded a story, i.e., the current state of the specification DFA is accepting.

For each $a \in A$, $x \in X$, and $z \in Z$, the function $\mathbf{O}(a,x,z)$ is an observation model, its value being the probability of observing z given that the system has entered

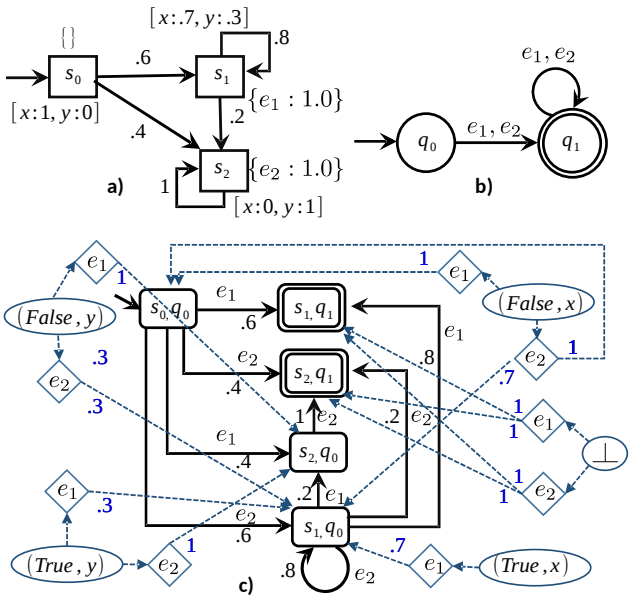

Figure 1. **a)** An event model $\mathcal{M}$ with its observation model $\mathcal{B}$. **b)** A DFA $\mathcal{D}$, specifying event sequences that contain at least one event. **c)** The Goal POMDP $\mathcal{P}_{(\mathcal{M},\mathcal{B},\mathcal{D})}$, constructed by Definition 6. (Self-loop transitions of the goal states have been omitted to try reduce visual clutter.)

state x via action a. The POMDP has a special observation, $\perp$, which is observed only when a goal state is reached. Any other observation is a pair (r,y) where $r \in \{\text{True}, \text{False}\}$ discloses whether the robot's prediction was correct—the event did happen—or not, and y indicates the sensed observation the robot made (as per $\mathcal{B}$). Rules 7a–7b ensure that the first element of the observation pair informs the robot whether its prediction was correct. To see this, if the robot has predicted e to occur, the event model has entered state s such that e has happened at s, and the robot has made an observation y, then the probability of observing (True, y) by entering to state (s,q) via action e is equal $h(s,y)g(s,e)$ (case 7a). If event e has not happened at s, then the robot's prediction has to be wrong, and thus, the probability of observing (False, y) in state (s,q) when it is reached via action e is $h(s,y)(1-g(s,a))$ (expressed in case 7b). Case 7c indicates the observation that the robot has completed recording of a story in $\mathcal{L}(\mathcal{D})$.

After making the product automaton in Definition 6, our algorithm first checks whether or not there is a policy that assures a desired story will be captured. Then, if there exists such a policy, it computes a policy minimizing the expected number of steps to record such a policy. We first focus on computing such a policy, assuming such a policy exists, in Section 4.2 (and Section 4.3 as well for special inputs of the problem), and then we discuss how our algorithm can check if such a policy exists or not in Section 4.4.

4.2 Solving the Goal POMDP

A POMDP is commonly treated as a fully observable MDP called a *belief MDP* whose (continuous) state space consists of the belief space of the POMDP. For details, see (Astrom, 1965; Sondik, 1978; Bonet and Geffner, 2009; Shani et al., 2013; Ross et al., 2008). Accordingly, in the belief MDP from Goal POMDP $\mathcal{P} = (X, A, b_0, \mathbf{T}, X_G, Z, \mathbf{O}, c)$, for each belief state $b \in \Delta(X)$, action $a \in A$, and observation

$z \in Z$, we denote the updated belief state of b after action a and observation z by b_z^a. It is computed as follows:

$$b_z^a(x) = Pr(x|z,a,b) = \frac{\mathbf{O}(a,x,z)\sum_{x' \in X}\mathbf{T}(x',a,x)b(x')}{Pr(z|a,b)}, \quad (3)$$

in which

$$Pr(z|a,b) = \sum_{x \in X}\mathbf{O}(a,x,z)\sum_{x' \in X}\mathbf{T}(x',a,x)b(x'). \quad (4)$$

For this belief MDP, the cost of each action a at belief state b is $c'(b,a) = \sum_{x \in X} b(x)c(x,a)$. In our case, $c'(b,a) = 1$ if b is a not a goal belief state, and otherwise $c'(b,a) = 0$. An optimal policy $\pi'^* : X \to A$ for this MDP is formulated as a solution to the Bellman recurrences

$$V'^*(b) = \min_{a \in A}\big(c'(b,a) + \sum_{z \in Z} Pr(z|a,b)V'^*(b_z^a)\big), \quad (5)$$

$$\pi'^*(b) = \arg\min_{a \in A}\big(c'(b,a) + \sum_{z \in Z} Pr(z|a,b)V'^*(b_z^a)\big). \quad (6)$$

Any standard technique may be used to solve these recurrences. For surveys on methods, see (Bonet and Geffner, 2009; Shani et al., 2013; Ross et al., 2008). Note that, in general, solutions to POMDPs are approximate solutions because it is intractable to provide exact solutions for POMDPs. An optimal policy computed via these recurrences prescribes, for any belief state reachable from b_0, an optimal action to execute. Hence, the robot executes, at each step, the action given by the optimal policy, and then updates its belief state via (3). One can show, via induction, that at each step i, there is a unique $q_i \in Q$ such that belief state b_i has outcomes only for (but probably not all) $x_j = (s_j, q_i) \in X, j = 1, 2, \cdots |S|$. As such, function $\beta : \Delta(X) \to \Delta(S) \times Q$ maps each b_i of those belief states to a tuple (d, q_i), where for each $s \in S$, $d(s) = b((s, q_i))$. Subsequently, the optimal policy π'^* computed for $\mathcal{P}_{(\mathcal{M},\mathcal{B},\mathcal{D})}$ can be mapped to an optimal solution $\pi^* : \Delta(S) \times Q \to A$ to RTM, by interpreting $\pi^*(\beta(b_i)) = \pi'^*(b_i)$, for each reachable belief state $b_i \in \Delta(X)$.

4.3 Solving RTM/FOM via a Goal MDP

The previous construction can be used to solve RTM/FOM instances too. But, since the event model fed into a RTM/FOM is fully observable, it seems rather more sensible, especially in terms of solution tractability, to construct a Goal MDP. To do so, for the event model and the DFA in Definition 6, the Goal MDP $\mathcal{M} = (X, A, b_0, \mathbf{T}, X_G, c)$ embedded in the POMDP $\mathcal{P}$ in that definition is extracted and then an optimal policy for $\mathcal{M}$ is solved. An optimal policy π''^* for the MDP is a function over $X = S \times Q$, which is computed via the Bellman equations

$$V''^*(x) = \min_{a \in A}\big(c(x,a) + \sum_{x' \in X} V''^*(x')\mathbf{T}(x,a,x')\big), \quad (7)$$

$$\pi''^*(x) = \arg\min_{a \in A}\big(c(x,a) + \sum_{x' \in X} V''^*(x')\mathbf{T}(x,a,x')\big). \quad (8)$$

These equations may be solved by a variety of methods. For a survey see (LaValle, 2006, Chp. 10). In the evaluation reported below, we use standard value iteration. After computing π''^*, for each $x = (s,q) \in X$, we make a belief state $b \in \Delta(S)$ such that $b(s') = 1$ if and only if $s' = s$, and then set $\pi^*(b,q) = \pi''^*((s,q))$, where π^* is an optimal solution to RTM/FOM. Observe that π^* for RTM/FOM is only computed for finitely many pairs (b,q), those in which b is a single outcome.

This section provided an algorithm to solve RTM and the two variants of it, RTM/FOM and RTM/FHM, assuming that there exists a policy that can guarantee that a desired story will be captured. However, for some input DFAs and event models to the problem, no such policy exists. Therefore, before using the algorithm in this section, one might want to check whether such a policy exists or not. The next section discusses how to answer that decision problem.

4.4 Deciding if a Policy Exists

In this section, we discuss how to check if there exists a policy that guarantees, for any execution of the event model, a story within the language of the DFA will be captured.

We first consider the RTM/FOM problem, recalling that we can compute a policy for the Goal MDP underlying the Goal POMDP in Definition 6 rather than the Goal POMDP itself. To provide an answer to the decision question, we can check whether or not there exists a policy for the MDP that guarantees that the goal states are reachable with probability 1. To find one, we check if there exists any policy that avoids the MDP's *dead-end* states, namely those states from which no goal state is reachable. If no policy can avoid these dead-ends, then no policy will ensure that a desired story will be captured.

To illustrate, consider Figure 2a, which shows a simple DFA specifying all stories that start with e_1. Figure 2b shows a simple event model in which e_1 happens at s_1 and e_2 occurs at s_2. Figure 2c shows the Goal MDP obtained from the product of this DFA and event model. In this example, no policy guarantees that a desired story will be captured with probability 1 because no policy assures that the goal states of the MDP are almost surely reachable. Should a policy choose event e_1 to capture in the first time step, then the probability that a desired story will be captured by this policy is 0.6, which happens when the event model enters state s_1 in the next time step. Similarly, if a policy chooses event e_2 in the same configuration, then the probability that a desired story is captured is again 0.6 if the policy chooses event e_1 when the DFA is in state q_0 and the event model is in state s_1. Observe that the Goal MDP constructed for this DFA and event model, which is shown in Figure 2c, has dead-end states (q_0, s_2) and (q_2, s_2) that are unavoidable.

Where there exists no policy that can guarantee a story will be captured, several strategies can be used to compute a reasonable policy that, nevertheless, may still be useful in practice. Generally, these maintain some balance between maximizing the probability of reaching a goal state and minimizing the expected number of steps to reach these goal states. See the discussion of Kolobov et al. (2012) for several such options.

An alternative strategy, and one specific to our context, would be to alter the specification by expanding the language described by the DFA. One seeks to modify the DFA to obtain a language which will ensure the existence of some policy guaranteeing a story be captured almost surely. One

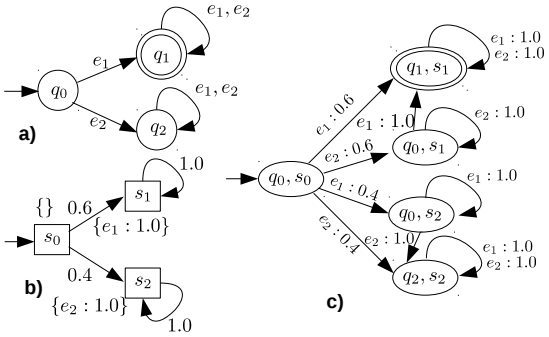

Figure 2. **a)** A DFA specifying all stories that contain at least an event over the event set $\{e_1, e_2\}$. **b)** A sample event model, in which event e_1 happens at s_1 and event e_2 happens at s_2. **c)** The Goal MDP obtained from the product of the DFA and the event model in parts (a) and (b). component.

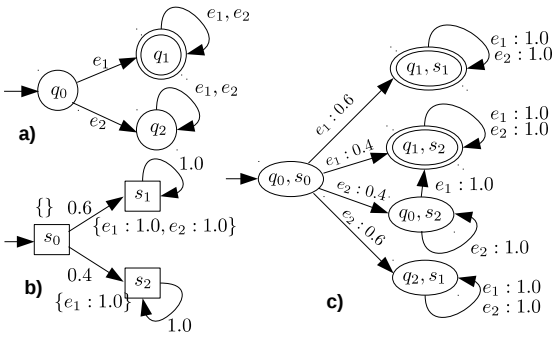

Figure 3. **a)** A DFA specifying all stories that contain at least an event over the event set $\{e_1, e_2\}$. **b)** A sample event model, in which event e_1 happens at s_1 and s_2 while event e_2 happens at s_1. **c)** The Goal MDP obtained from the product of the DFA and the event model in parts (a) and (b). component.

desires, naturally, that the altered DFA be "close" to the original in terms of some metric for the distance between two DFAs. In Section 8, we examine one such metric, the Levenshtein distance, and discuss how, given a DFA, to construct another that is within a desired Levenshtein distance.

Note, however, that the existence of dead-ends does not mean that no policy ensures that the robot is able to capture a desired story under any execution of the event model. If all the dead-ends are avoidable and some goal states are almost surely reachable, then there will exist such a policy. For illustration consider the example in Figure 3. In this example, the MDP, part (c) of the figure, has only a single dead-end state (q_2, s_1). But this dead-end is avoidable because the policy can avoid taking action b in state (q_0, s_0). Doing so, the probability of reaching this dead-end becomes zero, ensuring that all executions under this policy always end in goals. For the standard value iteration method to work, we first prepossess the MDP to identify dead-end states. The value of each dead-end is infinite as no goal state from that dead-end is reachable. For more discussions about MDPs with dead-ends, including algorithms to detect dead-ends and strategies to deal with them, we refer the reader to (Kolobov et al., 2012; Little and Thiébaux, 2007; Kolobov et al., 2010; Keyder and Geffner, 2008; Bonet and Geffner, 2003). In our breadth-first-like implementation, we search backward from the goal states, finding all states that are reachable from those states, avoiding dead-ends found so-far.

For RTM and RTM/FOM, to determine whether there exists any policy guaranteeing that a desired story will be captured, we need to ascertain, for the Goal POMDP in Definition 6, whether there exists a policy whose execution ends in the goal states with probability 1 or not. One must checked whether any belief state in the reachable part of the (infinite) belief MDP has an unavoidable dead-end belief state. In this case, we treat a belief state to be a dead-end if it arises from a dead-end state. This belief MDP has an infinite state space and we cannot deal with it directly, but instead one forms the finite support-belief MDP (Junges et al., 2021), whose states are the support for belief states. Note that, in general, the size of this support-belief MDP is exponential in the size of the MDP underlying the POMDP.

5 Representation-invariance of expected time

The event selected by the policy π^* at each step depends, in part, on the current state of the specification DFA. Because a single regular language may be represented with a variety of distinct DFAs with different sets of states —and thus, their optimal policies cannot be identical— one might wonder whether the expected execution time achieved by their computed policies depends on the specific DFA, rather than on the language. The question is particularly relevant in light of the language mutators we examine in Section 8. Here, we show that the expected number of steps required to capture a story within a given event model does indeed depend only on the language specified by the DFA, and not on the particular representation of that language.

For a DFA $\mathcal{D} = (Q, E, \delta, q_0, F)$, we define a function $f : Q \to \{0, 1\}$ such that for each $q \in Q$, $f(q) = 1$ if $q \in F$, and otherwise, $f(q) = 0$. Now consider the well-known notion of bisimulation, defined as follows:

Definition 7. [bisimulation] Given DFAs $\mathcal{D} = (Q, E, \delta, q_0, F)$ and $\mathcal{D}' = (Q', E, \delta', q_0', F')$, a relation $R \subseteq Q \times Q'$ is a *bisimulation relation* for $(\mathcal{D}, \mathcal{D}')$ if for any $(q, q') \in R$: (1) $f(q) = f'(q')$; (2) for any $e \in E$, $(\delta(q, e), \delta'(q', e)) \in R$.

Bisimulation implies language equivalence and vice versa.

Proposition 1. *(Rot et al., 2016)* For two DFAs $\mathcal{D} = (Q, E, \delta, q_0, F)$, $\mathcal{D}' = (Q', E, \delta', q_0', F')$, we have $\mathcal{L}(\mathcal{D}) = \mathcal{L}(\mathcal{D}')$ iff $(q_0, q_0') \in R$ for a bisimulation relation R for $(\mathcal{D}, \mathcal{D}')$.

Bisimulation is preserved for any reachable pairs. The state to which a DFA with transition function δ reaches by tracking an event sequence r from state q is denoted $\delta^*(q, s)$.

Proposition 2. If (q, q') are related by a bisimulation relation R for $(\mathcal{D}, \mathcal{D}')$, then for any $r \in E^*$, $(\delta^*(q, r), \delta'^*(q', r)) \in R$.

We now define a notion of equivalence for a pair of belief states.

Definition 8. [equivalence of belief states] Given an event model $\mathcal{M} = (S, \mathbf{P}, s_0, E, g)$, an observation model $\mathcal{B} = (Y, h)$ for $\mathcal{M}$, DFAs $\mathcal{D} = (Q, E, \delta, q_0, F)$ and $\mathcal{D}' = (Q', E, \delta', q_0', F')$ such that $\mathcal{L}(\mathcal{D}) = \mathcal{L}(\mathcal{D}')$, let $\mathcal{P}_{(\mathcal{M},\mathcal{B};\mathcal{D})} = (X, A, b_0, \mathbf{T}, X_G, Z, \mathbf{O}, c)$ and $\mathcal{P}'_{(\mathcal{M},\mathcal{B};\mathcal{D}')} = (X', A, b_0', \mathbf{T}', X_G', Z, \mathbf{O}', c')$. For two reachable belief states $b \in \Delta(X)$ and $b' \in \Delta(X')$, with $\beta(b) = (d, q)$ and $\beta'(b') = (d', q')$, we say that b' is *equivalent* to b, denoted $b \equiv b'$, if (1) (q, q') are related by a bisimulation relation for $(\mathcal{D}, \mathcal{D}')$ and that (2) $d = d'$, i.e. for each $s \in S$, $d(s) = d'(s)$.

Equivalence is preserved for updated belief states.

Lemma 1. Given the structures in Definition 8, let $b \in \Delta(X)$ and $b' \in \Delta(X')$ be two reachable belief states such that $b \equiv b'$. For any action $a \in A$ and observation $z \in Z$, it holds that $b_z^a \equiv {b'}_z^a$ and that $Pr(z|a, b) = Pr(z|a, b')$.

Proof. Let $b_2 = b_z^a$ and $b_2' = {b'}_z^a$, and assume $\beta(b_2) = (d_2, q_2)$ and $\beta'(b_2') = (d_2', q_2')$. The case where b and b' are both goal belief states, that is, where $f(q) = f'(q') = 1$ is immediately implied from the fact that the goal states of the POMDPs from in Definition 6 are absorbing, i.g., $b_2 = b$ and $b_2' = b'$. Therefore, we consider the case where $f(q) = f'(q') = 0$. Let b_a and b_a' be respectively the belief states resulted from doing action a (but before any observation) at belief states b and b'. These belief states are computed by the following formulas:

$$b_a(t) = \sum_{x \in X} \mathbf{T}(x, a, t) b(x), \tag{9}$$

and

$$b_a'(t') = \sum_{x' \in X'} \mathbf{T}'(x', a, t') b'(x'). \tag{10}$$

Given that $\beta(b) = (d, q)$, belief state b can have outcomes only for those states that are among $x_1 = (s_1, q), x_2 = (s_2, q), \cdots, x_n = (s_n, q)$, and similarly, b' can have outcomes only for states that are among $x_1' = (s_1, q'), x_2' = (s_2, q'), \cdots, x_n' = (s_n, q')$, where $n = |S|$. Also, because it is assumed that $d = d'$, for each integer $1 \leq j \leq n$, $b((s_j, q)) = b'((s_j, q'))$, or in other words, $b(x_j) = b'(x_j')$. Now, with (9) and given the construction of the transition probability function $\mathbf{T}$ in Definition 6, the only states for which b_a can have outcomes are among $t_1 = (s_1, q), t_2 = (s_2, q) \cdots, t_n = (s_n, q)$ and $t_{n+1} = (s_1, \delta(q, a)), t_{n+2} = (s_2, \delta(q, a)), \cdots, t_{2n} = (s_n, \delta(q, a))$. Similarly, the only states for which b_a' can have outcomes are among $t_1' = (s_1, q'), t_2' = (s_2, q') \cdots, t_n' = (s_n, q')$ and $t_{n+1}' = (s_1, \delta'(q', a)), t_{n+2}' = (s_2, \delta'(q', a)), \cdots, t_{2n}' = (s_n, \delta'(q', a))$.

Now, we claim that for each integer $1 \leq k \leq 2n$, $b_a(t_k) = b_a'(t_k')$. To prove this, consider that by assumption, q and q' are related by a bisimulation relation for $(\mathcal{D}, \mathcal{D}')$, and this, by Definition 7, means that $q \in F$ iff $q' \in F'$. As a result, by the construction of the transition probability function in Definition 6, for each pair of integers $1 \leq i, j \leq n$, $\mathbf{T}((s_j, q), a, (s_i, q)) = \mathbf{T}'((s_j, q'), a, (s_i, q'))$ and $\mathbf{T}((s_j, q), a, (s_i, \delta(q, a))) = \mathbf{T}'((s_j, q'), a, (s_i, \delta'(q', a)))$, which together mean that for integers $1 \leq j \leq n$ and $1 \leq k \leq 2n$, $\mathbf{T}(x_j, a, t_k) = \mathbf{T}'(x_j', a, t_k')$. We use this, the assumption that $b(x_j) = b'(x_j')$ for all $1 \leq j \leq n$, (9) and (10) to prove our claim as follows:

$$\begin{aligned} b_a(t_k) &= \sum_{1 \leq j \leq n} \mathbf{T}(x_j, a, t_k) b(x_j) \\ &= \sum_{1 \leq j \leq n} \mathbf{T}'(x_j', a, t_k') b'(x_j') = b_a'(t_k'). \end{aligned} \tag{11}$$

To prove that $Pr(z|a, b) = Pr(z|a, b')$, consider the following formulas:

$$Pr(z|a, b) = \sum_{t \in X} \mathbf{O}(a, t, z) b_a(t), \tag{12}$$

and

$$Pr(z|a, b') = \sum_{t' \in X'} \mathbf{O}'(a, t', z) b_a'(t'). \tag{13}$$

By the construction of the observation function in Definition 6 and that $q \in F \iff q' \in F'$, it follows that for each integer $1 \leq j \leq n$, $\mathbf{O}(a, (s_j, q), z) = \mathbf{O}'(a, (s_j, q'), z)$. Similarly, for each integer $1 \leq j \leq n$, $\mathbf{O}(a, (s_j, \delta(q, a)), z) = \mathbf{O}'(a, (s_j', \delta'(q', a), z)$. Together, these two mean that for each integer $1 \leq k \leq 2n$, $\mathbf{O}(a, t_k, z) = \mathbf{O}'(a, t_k', z)$. This, combined with (11), proves a part of the lemma as follows:

$$\begin{aligned} Pr(z|a, b) &= \sum_{t \in X} \mathbf{O}(a, t, z) b_a(t) \\ &= \sum_{1 \leq k \leq 2n} \mathbf{O}(a, t_k, z) b_a(t_k) \\ &= \sum_{1 \leq k \leq 2n} \mathbf{O}'(a, t_k', z) b_a'(t_k') \\ &= \sum_{t' \in X'} \mathbf{O}'(a, t', z) b_a'(t') \\ &= Pr(z|a, b'). \end{aligned} \tag{14}$$

To prove that $b_2 \equiv b_2'$, consider that for each $x \in X$ and $x \in X'$, $b_2(x)$ and $b_2'(x')$ are computed as follows:

$$b_2(x) = \mathbf{O}(a, x, z) b_a(x) / Pr(z|a, b), \tag{15}$$

and

$$b_2'(x') = \mathbf{O}'(a, x', z) b_a'(x') / Pr(z|a, b'). \tag{16}$$

These two formulas combined with (11) and (14), and the fact that $\mathbf{O}(a, t_k, z) = \mathbf{O}'(a, t_k', z)$ for all $1 \leq k \leq 2n$, imply that $b_2(t_k) = b_2'(t_k')$ for all $1 \leq k \leq 2n$. Therefore, $d_2 = d_2'$. We now only need to prove that q_2 and q_2' are related by a bisimulation relation for $(\mathcal{D}, \mathcal{D}')$. Observe that if the robot's prediction of occurring event a was wrong, then $q_2 = q$ and $q_2' = q'$, and otherwise, $q_2 = \delta(q, a)$ and $q_2' = \delta'(q', a)$. In the former case, by definition, q_2 and q_2' are related by a bisimulation relation R for $(\mathcal{D}, \mathcal{D}')$, and in the later case, by Proposition 2, q_2 and q_2' are related by the same bisimulation relation q and $q\prime$ were related by. Thus, we conclude $b_z^a \equiv {b'}_z^a$. □

Note that for a Goal POMDP $\mathcal{P}$ with initial belief state b_0, $V^*(b_0)$ is the expected cost of reaching a goal belief state via an optimal policy for $\mathcal{P}$. We now present our result.

Theorem 1. For the structures in Definition 8, it holds that $V^*(b_0) = V'^*(b'_0)$.

Proof. For a belief MDP $\mathcal{M}$, let $\mathrm{Tree}(\mathcal{M})$ to be its tree-unravelling—the tree whose paths from the root to the leaf nodes are all possible paths in $\mathcal{M}$ that start from the initial belief state. A policy π for $\mathcal{M}$ chooses a fixed set of paths over $\mathrm{Tree}(\mathcal{M})$, and the expected cost of reaching a goal belief state under π is equal to $\sum_{p\in \mathrm{GoalPaths}(\pi,\mathrm{Tree}(\mathcal{M}))} C(p)W(p)$, where $\mathrm{GoalPaths}(\pi, \mathrm{Tree}(\mathcal{M}))$ is the set of all paths that are chosen by π and reach a goal belief state from the root of $\mathrm{Tree}(\mathcal{M})$, $C(p)$ is the sum of costs of all transitions in path p, and $W(p)$ is the product of the probability values of all transitions in p. The idea is that if we can overlap the tree-unravellings of the belief MDPs $\mathcal{P}_{(\mathcal{M},\mathcal{B};\mathcal{D})}$ and $\mathcal{P}'_{(\mathcal{M},\mathcal{B};\mathcal{D}')}$ in such a way that each pair of overlapped belief states are equivalent in the sense of Definition 8 and that each pair of overlapped transitions have the same probability and the same cost, then for each pair of overlapped belief states $b \in \Delta(X)$ and $b' \in \Delta(X')$, if we use $\pi^*(b)$ as the decision at the belief state b' then, because those fixed paths are overlapped, we know $V^*(b_0) \geq V'^*(b'_0)$. And, in a similar fashion, $V^*(b_0) \leq V'^*(b'_0)$, and thus, $V^*(b_0) = V'^*(b'_0)$. The following construction makes those trees and shows how they can be overlapped.

For an integer $n \geq 1$, we can make two trees T_n and T'_n as follows: (1) Set b_0 as the root of T_n and set b'_0 as the root of T'_n; make a relation R and set $R \leftarrow \{(b_0, b'_0)\}$. (2) While $|T_n| < n$, extract a pair (b, b') from R that has not been checked yet and in which b and b' are not goal belief states; for each action a and observation z, compute b^a_z and b'^a_z, add node b^a_z and edge (b, b^a_z) to T, and add node b'^a_z and edge (b', b'^a_z) to T'; label both edges (a, z). Also assign to edge (b, b^a_z), $Pr(z|a, b)$ as its probability value, and set the probability value of (b', b'^z_a), $Pr(z|a, b')$; the cost of each edge is set 1. Finally, add (b^a_z, b'^a_z) to R.

Given that $\mathcal{L}(\mathcal{D}) = \mathcal{L}(\mathcal{D}')$, by Proposition 1, states q_0 and q'_0 are related by a bisimulation relation for $(\mathcal{D}, \mathcal{D}')$, which (by Definition 8 and the construction in Definition 6) implies that $b_0 \equiv b'_0$. This combined with Lemma 1 implies that for each pair $(b, b') \in R$, $b \equiv b'$. We now match T_n and T'_n so that each pair (b, b') that are related by R overlap. By Lemma 1, each pair of overlapped edges have the same probability value and the same cost value. Since for any integer $n \geq 0$ we can overlap trees T_n and T'_n in the desired way, we can overlap the tree-unravellings of the belief MDPs of $\mathcal{P}_{(\mathcal{M},\mathcal{B};\mathcal{D})}$ and $\mathcal{P}'_{(\mathcal{M},\mathcal{B};\mathcal{D}')}$ in the desired way too; this completes the proof. □

The upshot of this analysis is that we need attend only to the story specification language (given indirectly via $\mathcal{D}$), the specific presentation of that language does not impact the expected number of steps to capture an event sequence satisfying that specification.

6 Faster Algorithms for Special Structures

In this section, we consider several special cases of event models and DFAs for which new algorithms are feasible. Though less general than the approaches introduced in the prior sections, these new algorithms exploit the structure of the special cases to run significantly faster.

6.1 The DFA is loop-omitted acyclic

The first case is when the DFA adheres to the following requirement.

Definition 9. A DFA $\mathcal{D} = (Q, E, \delta, q_0, F)$ is called a *loop-omitted acyclic DFA* if for every state $q \in Q$ and string $\eta \in E^*$ for which $\delta^*(q, \eta) = q$, it holds for every prefix η' of η that $\delta^*(q, \eta') = q$.

Intuitively, a DFA is a loop-omitted acyclic DFA if it does not have any cycle except self-loops on the states. We speculate that many DFAs of interest for this sort of problem will have this property. In fact, all the DFAs in this paper's case studies are loop-omitted acyclic DFAs; likewise for the case studies in our other extensions to this work (Chaudhuri et al., 2021a,b)

For this kind of DFA, regardless of the form of the event model, the graph underlying the Goal POMDP in Definition 6 will be a layered directed acyclic graph where each layer is, in fact, a strongly connected component (SCC). Note that all states $x = (q, s)$ within a single SCC share a single DFA state q and all those states represent a situation where either the event predicted by the robot does not happen in the next time step, making the DFA stay in the same state q, or the predicted event did happen but state q transitions back to itself with that predicted event.

For an example, see the DFA in Figure 4a and the event model in Figure 4b. The set of possible events consists of e_1 and e_2. Event e_1 happens with probability 1 at state s_1, while event e_2 happens with probability 1 at state s_2, and at each state of the event model no more than one event happens. Figure 4c shows the state space and the transition function of the Goal POMDP constructed from the product of the DFA and the event model based on Definition 6. This product, which is decomposed into its set of SCCs in Figure 4d, has five SCCs C_0, C_1, C_2, C_3, and C_4. There is only a single topological ordering of these SCCs, namely $(C_0, C_1, C_3, C_2, C_4)$.

Now, we consider solving the RTM/FOM problem where the input DFA is loop-omitted acyclic. Recall that to solve that problem we need to compute a policy that minimizes the expected number of steps to reach a goal state, and observe that, in this case, the Goal MDP underlying the Goal POMDP in Definition 6 is a layered DAG. Thus, to compute an optimal policy for such a Goal MDP, we can use the *topological value iteration algorithm* of Dai and Goldsmith (2007).

Their algorithm considers each SCC as a *metastate* and then computes an optimal policy for those metastates based on a reverse ordering of a topological ordering of the metastates. The optimal action for states within each metastate is computed using value iteration. In Dai and Goldsmith's algorithm, each metastate is solved only once because the graph connecting the metastates is acyclic. Note that when a metastate (SCC) is solved, only the values of states within that metastate are backed up, whereas in the classical value iteration, at each step, the values of all states are backed up until their values converge. To

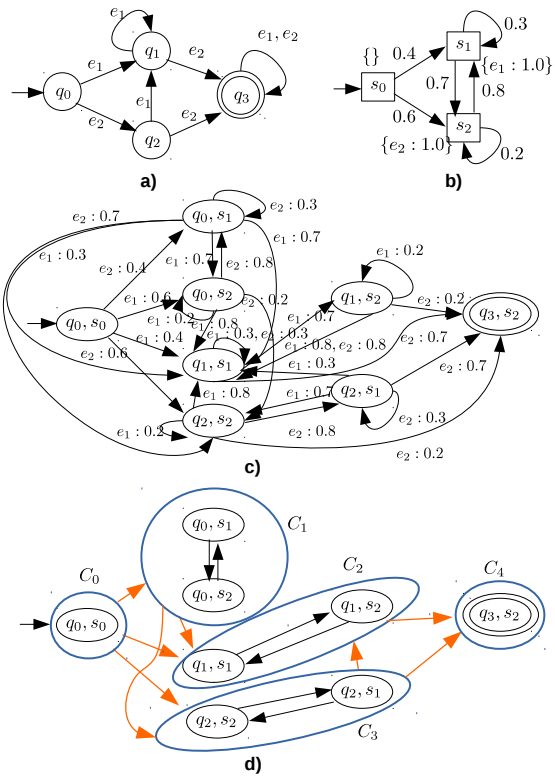

Figure 4. **a)** A loop-omitted acyclic DFA. **b)** A sample event model. **c)** The Goal MDP obtained from the product of the DFA and the event model in parts (a) and (b). **d)** It shows the strongly connected components of graph underlying the MDP in part c. Each blue circle is a strongly connected component. The self-loops and the edges between states of different SCCs have been omitted to reduce visual clutter.

illustrate their algorithm, consider again the Goal MDP in Figure 4c. Recall that there was only one topological ordering, $(C_0, C_1, C_3, C_2, C_4)$, between the SCCs of the MDP. Accordingly, their algorithm first computes an optimal action for the single state in C_4, then for the states in C_2, then for the states in C_3, then for the states in C_1, and finally for the only state of C_0.

For solving the RTM and the RTM/FHM problems with input DFAs which are loop-omitted acyclic, we need to solve a Goal POMDP that has a layered DAG structure, and for solving those Goal POMDPs, one can use the algorithm of Dibangoye et al. (2009), which computes a policy using a point-based method. Their algorithm constructs the layered acyclic graph for the belief points by utilizing the layered acyclic structure of the POMDP.

6.2 The Goal MDP is a directed acyclic graph

Another special case is where the graph underlying the Goal MDP is a DAG if all the self-loops are removed from it.

We clarify this special case in the following definition.

Definition 10. For a given Goal MDP $\mathcal{M} = (X, A, b_0, \mathbf{T}, X_G, c)$, let $\mathbf{G}(\mathcal{M}) = (V, E)$ be the graph underlying $\mathcal{M}$, in which $V = X$ and $E = \{(x,t) \in X^2 \mid \mathbf{T}(x,t) > 0\}$, and let $\mathbf{Loop}^-(\mathbf{G}(\mathcal{M})) = (V, E')$ be the graph obtained from $\mathbf{G}(\mathcal{M})$ by removing the self-loops from it, that is, $E' = E \setminus \{(x,x) \mid x \in X\}$. We say that $\mathcal{M}$ is a *loop-omitted directed acyclic graph*, or loop-omitted DAG for short, if $\mathbf{Loop}^-(\mathbf{G}(\mathcal{M}))$ is a directed acyclic graph.

Note that this is stronger than the previous case, as it corresponds to the circumstance where each SCC is a single vertex. Nevertheless, it is possible for a Goal MDP to be a loop-omitted DAG, while its underlying graph has self-loops. Figure 5 provides one such example. This kind of MDPs arises, in particular, in applications where the DFA specifying all desired stories is a loop-omitted acyclic DFA and the graph underlying the event model is also a DAG if the self-loops removed from that graph. A special case of that kind of event model is where the event model has only a single state, which might be created by collapsing all the states of an original event model in order to make an approximation of the original event model.

In the rest of this section, we only consider RTM/FOM problems for this kind of MDP. Observe that because any Goal MDP that is a loop-omitted DAG is a layered DAG where each strongly connected component of the graph underlying the Goal MDP has only one state, we can use the algorithm of the previous section, the topological value iteration algorithm for MDPs, to compute an optimal policy for the MDP. This requires iteration to update the value of a state using the Bellman equation until the value of the state converges. Though this algorithm is faster than the original value iteration algorithm for MDPs, for solving large MDPs, it still may require a considerable amount of time for the state values to converge, and therefore, we propose a faster algorithm that does not require value iteration at all, and the Bellman equation for each state is solved by solving several simple, single-variable equations.

In this algorithm we first choose a topological ordering of the non-goal states (state values of all the goal states are zero) of the MDP. Then, at each step, we take a state from the MDP based on the reverse of the topological ordering to compute an optimal action for that state and the value of that state. To do so, for each state x and action $a \in A$, we introduce a variable $t_{x,a}$ to denote the expected number of steps from x to reach a goal state of the MDP if the policy assigns action a to state x. Also, for each state x, we introduce a variable t_x to denote the expected number of steps to reach a goal state from x under an optimal policy. To compute the optimal action for each state x, if x is a goal state, then $t_x = 0$, meaning that the expected number of steps to reach a goal state from x under an optimal policy is zero. If state x is not a goal state, then for each action $a \in A$, we solve the following equation

$$t_{x,a} = \mathbf{T}(x,a,x)(1 + t_{x,a}) + \sum_{\substack{x' \in X \\ x' \neq x}} \mathbf{T}(x,a,x')(1 + t_{x'}). \tag{17}$$

Then, we compute t_x simply as $t_x = \min_{a \in A}\{t_{x,a}\}$. For each $x \in X \setminus X_G$, we set $\pi^*(x) = \arg\min_{a \in A}\{t_{x,a}\}$. By doing so, we compute an optimal policy π^*.

We end this section by illustrating this algorithm via an example for the MDP in Figure 5. There is a single topological ordering of the non-goal states: $x_0 \to x_1 \to x_2$. To compute an optimal policy, we pick this (the sole) choice of ordering. Next, we compute the optimal action for x_2. For this purpose, for actions a and b, we need to solve the

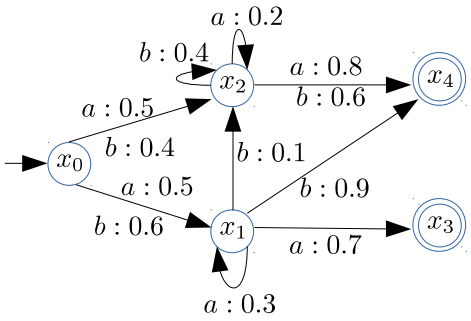

Figure 5. A sample of an MDP that is a directed acyclic graph when all the self-loops are removed.

following equations,

$$t_{x_2,a} = 0.2(1 + t_{x_2,a}) + 0.8(1 + t_{x_4}) \tag{18}$$

and

$$t_{x_2,b} = 0.4(1 + t_{x_2,a}) + 0.6(1 + t_{x_4}). \tag{19}$$

Based on these two equations, we have $t_{x_2,a} = 1.25$ and $t_{x_2,b} = 1.67$. So, $t_{x_2} = 1.25$, and thus, we set $\pi^*(x_2) = a$. Then, we solve the following equations for x_1.

$$t_{x_1,a} = 0.3(1 + t_{x_1,a}) + 0.7(1 + t_{x_3}) = 1 + 0.3t_{x_1,a} \tag{20}$$

and

$$t_{x_1,b} = 0.1(1 + t_{x_2}) + 0.9(1 + t_{x_4}) = 1.125. \tag{21}$$

So, $t_{x_1} = 1.125$, and hence, we set $\pi^*(x_1) = b$. To compute the optimal action for x_0, we solve the following equations,

$$t_{x_0,a} = 0.5(1 + t_{x_2}) + 0.5(1 + t_{x_1}) \tag{22}$$

and

$$t_{x_0,b} = 0.4(1 + t_{x_2}) + 0.6(1 + t_{x_1}). \tag{23}$$

As such, $t_{x_0} = 2.175$, and therefore, we let $\pi^*(x_0) = b$.

7 Greedy algorithm

In this section, we consider a greedy algorithm for solving RTM and RTM/FHM, which we also specially adapt for RTM/FOM. This greedy algorithm will serve as a baseline for comparison for the case studies in Section 9.

The idea is, at each time step, simply to choose an event to capture that has the highest probability of occurring in the next time step. To do so, the robot first uses the event model $\mathcal{M} = (S, \mathbf{P}, s_0, E, g)$ and the DFA $\mathcal{D} = (Q, E, \delta, q_0, F)$, to construct the Goal POMDP $\mathcal{P}_{(\mathcal{M},\mathcal{B};\mathcal{D})} = (X, A, b_0, \mathbf{T}, X_G, Z, \mathbf{O}, c)$ based on the construction in Definition 6. Then, at each time step, it uses this POMDP to compute an event that has the highest probability to be chosen in the next time step. Importantly, in this greedy approach, the robot considers only a single step, and does not compute a policy that minimizes the expected number of steps to reach a goal state for this POMDP.

For each time step k, the robot maintains the current belief state $b_k \in \Delta(X)$ of the POMDP and the current state q_k of the DFA. Then, for all events e for which $\delta(q_k, e) \neq q_k$ and from $\delta(q_k, e)$ at least one accepting state is reachable, we compute the probability that e happens in the next time step given b_k as follows:

$$Pr(e \text{ happens in the next time step} \mid b_k) = \sum_{x=(s,q)\in X} b_k[x] \cdot \sum_{s'\in S} \mathbf{P}(s, s')g(s', e). \tag{24}$$

Hence, the robot hopes to record in the next time step, an event that has the greatest probability computed by this equation. In the case that several such events exist, the robot chooses one of them arbitrarily.

Given that RTM/FOM is a special form of RTM, the process described so far in this section applies for RTM/FOM too, but for RTM/FOM we can avoid constructing the product of the event model and the DFA. For RTM, the robot maintains, at each time step k, the current state s_k of the event model and the current state q_k of the DFA. Then, at step k, it computes for all events e for which $\delta(q_k, e) \neq q_k$ and from $\delta(q_k, e)$ an accepting state is reachable, the following probability

$$Pr(e \text{ happens in the next time step} \mid s_k) = \sum_{s'\in S} \mathbf{P}(s_k, s')g(s', e), \tag{25}$$

which is essentially the probability that e happens in the next time step. Then, from among all events that obtain the highest value in this equation, the robot chooses one to attempt to record in the next time step.

We compare this greedy algorithm with our general algorithm in Section 9 to assess their relative solution quality.

8 Construction of Specification Languages

This section describes how one might construct, in a partially automated way, specifications for a variety of interesting scenarios. The idea is to use a variety of mutators to construct specification DFAs.

8.1 Multiple recipients

Suppose we would like to capture several videos, one for each of several recipients, within a single execution. Given language specifications $\mathcal{D}_1, \ldots, \mathcal{D}_n \in \mathscr{D}$, where $\mathscr{D}$ denotes the set of all DFAs over a fixed event set E, how can we form a single specification that directs the robot to capture events that can be post-processed into the individual output sequences? One way is via two relatively simple operations on DFAs:

($\mathbf{M_S}$) A *supersequence* operation $\mathbf{M_S} : \mathscr{D} \to \mathscr{D}$, where

$$\mathcal{L}(\mathbf{M_S}(\mathcal{D})) = \{w' \in E^* \mid w' \text{ is supersequence of a } w \in \mathcal{L}(\mathcal{D})\}.$$

This operation is produced by first treating $\mathcal{D}$ as a nondeterministic finite automaton (NFA) and then, for each event and state, adding a transition labeled by that event from that state to itself, and converting result back into a DFA (Rabin and Scott, 1959).

($\mathbf{M_I}$) An *intersection* operation $\mathbf{M_I} : \mathscr{D} \times \mathscr{D} \to \mathscr{D}$, under which

$$\mathcal{L}(\mathbf{M_I}(\mathcal{D}_1, \mathcal{D}_2)) = \mathcal{L}(\mathcal{D}_1) \cap \mathcal{L}(\mathcal{D}_2).$$

Based on these two operations, we can form a specification that asks the robot to capture an event sequence that satisfies all n recipients as follows:

$$\mathcal{D} = \mathbf{M_I}(\mathbf{M_I}(\mathbf{M_S}(\mathcal{D}_1), \mathbf{M_S}(\mathcal{D}_2)) \ldots, \mathbf{M_S}(\mathcal{D}_n))$$

Then from any $\xi \in \mathcal{L}(\mathcal{D})$, we can produce a $\xi_i \in \mathcal{L}(\mathcal{D}_i)$ by discarding (as a post-production step) some events from ξ.

8.2 Mistakes were made

What should the robot do if it simply cannot capture an event sequence that fits its specification $\mathcal{D}$, either because some necessary events did not occur, or because the robot failed to capture them when they did occur? One possibility is to accept some limited deviation between the desired specification and what the robot actually captures.

Let $d : E^* \times E^* \to \mathbb{Z}^+$ denote the Levenshtein distance (Levenshtein, 1966), that is, a distance metric that measures the minimum number of insert, delete, and substitute operations needed to transform one string into another. A mutator that allows a bounded amount of such distance might be:

($\mathbf{M_L}$) A *Levenshtein mutator* $\mathbf{M_L} : \mathscr{D} \times \mathbb{Z}^+ \to \mathscr{D}$ that transforms a DFA $\mathcal{D}$ into one that accepts strings within a given distance from some string in $\mathcal{L}(\mathcal{D})$.

$$\mathcal{L}(\mathbf{M_L}(\mathcal{D}, k)) = \{\xi \mid \exists \xi' \in \mathcal{L}(\mathcal{D}), d(\xi, \xi') \leq k\}.$$

This mutation can be achieved using a *Levenshtein automaton* construction (Schulz and Mihov, 2002; Konstantinidis, 2007). Then, if the robot captures a sequence in $\mathcal{L}(\mathbf{M_L}(\mathcal{D}, k))$, it can be converted to a sequence in $\mathcal{L}(\mathcal{D})$ by at most k edits. For example, an insertion edit would perhaps require the undesirable use of alternative 'stock footage', rendering of appropriate footage synthetically, or simply a leap of faith on the part of the viewer. By assigning the costs associated with each edit appropriately in the construction, we can model the relative costs of these kinds of repairs.

8.3 At least one good shot

In some scenarios, there are multiple distinct views available of the same basic event. We may consider, therefore, scenarios in which this kind of good/better correspondence is known between two events, and in which the robot should endeavor to capture, say, at least one better shot from that class. We define a mutator that produces such a DFA:

($\mathbf{M_G}$) An *at-least-k-good-shots* mutator $\mathbf{M_G} : \mathscr{D} \times E \times E \times \mathbb{Z}^+ \to \mathscr{D}$, in which $\mathbf{M_G}(\mathcal{D}, e, e', k)$ produces a DFA in which e' is considered to be a superior version of event e, and the resulting DFA accepts strings similar to those in $\mathcal{L}(\mathcal{D})$, but with at least k occurrences of e replaced with e'.

The construction makes a DFA in which $\mathcal{D}$ has been copied $k+1$ times, each called a *level*, with the initial state at level 1 and the accepting states at level $k+1$. Most edges remain unchanged, but each edge labeled e, at all levels less than $k+1$, is augmented by a corresponding edge labeled e' that moves to the next level. This guarantees that e' has replaced e at least k times, before any accepting state can be reached.

9 Case studies

In this section, we present several examples, solved via our Python implementation of the general algorithm proposed in Section 4. (We will refer to it as "the general algorithm" from now on.) For RTM/FOM we form the Goal MDP, while for RTM/FHM and RTM we form a Goal POMDP. To solve the POMDP, we use APPL online (Approximate POMDP Planning Online) toolkit, which implements the DESPOT algorithm (Somani et al., 2013)—one of the fastest known online solvers. We compare the results for different observability conditions based upon the number of steps that the system will run. It will run until the robot records a desired story under an optimal policy, so we must consider the expected number of steps.

We also compare our general algorithm with a the greedy algorithm of Section 7, which, at each time step, attempts to capture an event that has the highest probability of occurrence at the next time step.

9.1 Turisti Oulussa

In pre-COVID 2019, William is a tourist visiting Oulu as shown in Figure 6a. William's family has privately contracted a robotic videography company to record him seeing the sights, specifically the Kauppahalli (k), the Hupisaaret park (h), and either Tietomaa museum (t) or the Oulu Cathedral (c). The robot does not know William's specific plans, but it does know, through some statistics, that a typical tourist moves among those districts according to the event model in Figure 6b.

The desired video is specified using the DFA in Figure 6c. The robot is given other tasks to do aside from recording William, and thus, cannot merely follow William; it must form a strategy that predicts which events to try to capture.

We considered three settings: (1) RTM/FOM: the robot always knows the current district in which William is located, perhaps by the help of some static sensors; (2) RTM: the robot does not know at which district William is currently located but there is a single useful observation, a message sent from a security guard in district s_1, that informs the robot that William is in district s_1 whenever he is there; (3) RTM/FHM: the robot receives no direct knowledge about William's location. We computed the optimal policy for RTM/FOM, case (1), using the Goal MDP approach in Section 4.3. According to this policy, the expected number of steps to record under an optimal policy with full observability, a story satisfying the specification, is approximately 35.39. The computed optimal policy for this case is shown in Figure 6d. Each oval in this figure is a state of the DFA and inside each of those ovals, all the states of the event model are drawn as boxes. The event labeled inside a box is the event chosen by the optimal policy when the DFA is in the state represented by that oval and the event model is in the state represented by that box. Each empty box is assigned an arbitrary event by the policy and those events assigned to those empty boxes are irrelevant to recording a desired story.

To verify the correctness of the algorithm, we simulated the execution of this policy 5,000 times. In each simulation, William followed a random path through the city according to the event model in Figure 6b, and the robot executed

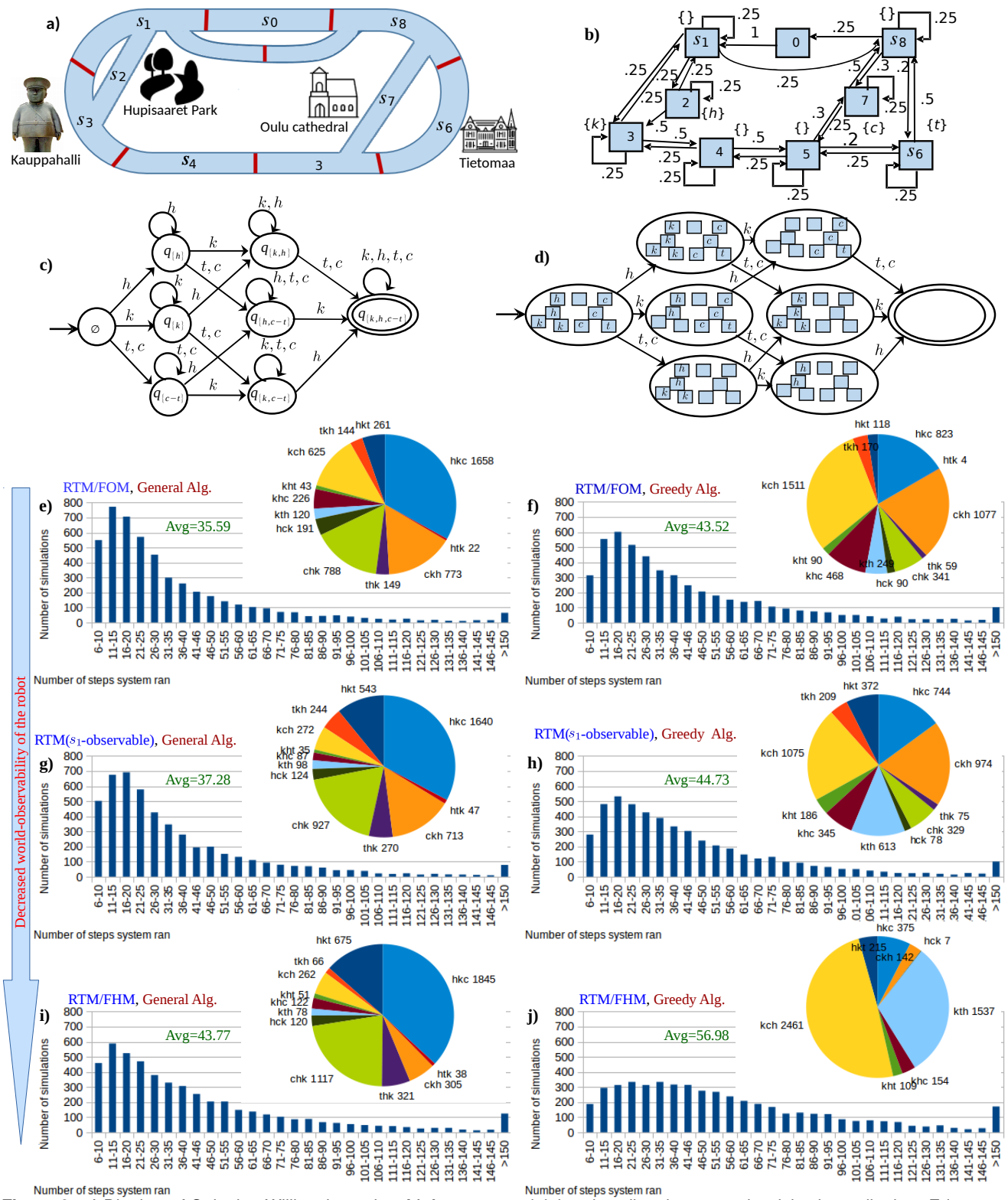

Figure 6. **a)** Districts of Oulu that William is touring. **b)** An event model that describes how a tourist visits those districts. Edges are labeled with transition probabilities. **c)** A DFA specifying that the captured story must contain events k and h and at least one of c or t. **d)** The optimal policy for the RTM/FOM problem for Oulu. Subfigures **e–j** consist of a histogram showing, for 5,000 simulations, the distribution of the number of hours (steps) William (system) circulated (ran) until the robot recorded a story specified by the DFA, and a pie chart showing the distribution of recorded sequences in these simulations. Pairs **e)** and **f)** are for the RTM/FOM problem, **g)** and **h)** for the RTM problem, and **i)** and **j)** are for the RTM/FHM problem. The left column has the robot using the general algorithm, while the right column uses the greedy approach. Moving down the column, the average number of steps to record a story increases as the robot's perception of the world's state diminishes. While the distribution of the recorded sequences by the general algorithm differs from the distribution of the recorded sequences by the greedy algorithm, for each of the greedy algorithm and the general algorithm, the distributions of the recorded sequences under different levels observability were similar.

the computed policy to capture an event sequence satisfying the specification. The average number of steps to record a

satisfactory sequence for those 5,000 simulations using our general algorithm was 35.59, quite close to the expected number of steps. Figure 6e shows results of those simulations in form of a histogram and a pie chart. We also made 5,000 simulations of the same RTM/FOM problem and in each simulation we let the robot to use the greedy algorithm to record a desired story. The average number of steps for this experiment was 43.52, which is substantially longer than the expected number of steps to record a desired event sequence with the optimal policy for RTM/FOM, 35.59. This is justified, in particular, by the fact that when the robot is in state s_0 and it has captured neither h nor k, the greedy algorithm does not consider the fact that the best event to predict at that time to decrease the average number of steps is h because, if William enters s_2 in the next time step, then it is possible than he enters s_3 from s_2 and, thus, the robot could capture both events h and k during a single circuit of the environment. See Figure 6f for additional details regarding this experiment.

For cases (2) and (3), our algorithm constructed a Goal POMDP, as described by Definition 6, which is then supplied to APPL Online to perform 5,000 simulations. Also, for each of the two cases, we generated 5,000 simulations in our program and let the greedy algorithm decide which event to try to capture. In case (2), RTM with a useful observation, the average number of steps to record a desired story using our general algorithm and the greedy algorithm were 37.28 and 44.73, respectively. In case (3), RTM/FHM, the general algorithm and the greedy algorithm had the robot record a desired story in 43.77 and 56.98 steps, respectively, on average.

The histograms and the pie charts for these four experiments are shown in Figures 6g–6j. In these figures, notice how a single observation of whether William is in s_1 helps the robot to record a story considerably faster than when it hasn't got access to that state information. To such a robot, even a stream of quite limited information, if aptly chosen, can be very useful.

A further qualitative remark: note how the histogram changes as the level of observability increases, from RTM/FOM to RTM/FHM: the robot is able to utilize the additional information to capture stories more rapidly. Also, across each of the three settings RTM/FOM, RTM, and RTM/FHM, the average number of steps needed via the greedy algorithm is considerably greater than via the general algorithm.

9.2 Wedding reception

Suppose a videographer robot is asked to produce videos that convey different stories, assembled from unpredictable events at a wedding reception. The wedding guests include Alice, Bob, and Chris, and the events of interest for any of those guests are: arriving at the reception, (i); dancing, (d); drinking coffee, (c); drinking other beverages, (b); smoking, (s); and being entertained, (e). Each guest has their own sense of the events they would like to see captured: Alice is mainly interested in seeing Chris drinking or smoking, but also has plans to share the last dance with Bob; Bob cares for nothing but seeing his own dancing through the evening, but hopes to share the last dance with Alice; Chris does not care to see any events at all, but Chris's children are concerned about his unhealthy habits, and so if Chris is drinking too much coffee or smoking too much, they would like to know. The robot in that scenario is given three parallel objectives. We can formalize those as languages, shown here for compactness as regular expressions: for Alice, $r_1 = (s_3 + c_3)^+ d_{12}$; for Bob, $r_2 = (d_2 + d_{12} + d_{23})^+ d_{12}$; and for Chris, $r_3 = (s_3 + c_3)(s_3 + c_3)(s_3 + c_3)^+$. These three requests—where subscript labels 1, 2, and 3, respectively represent Alice, Bob, and Chris—are encoded using DFAs $\mathcal{D}_1$, $\mathcal{D}_2$, and $\mathcal{D}_3$, respectively.

The behavior of each guest is modeled by the event model in Figure 7a, in which $\mathbf{P}$ is the transition probability function of the model. The joint behavior of the three guests is modeled by an event model $\mathcal{M}$ obtained as the Cartesian product of the models for the individuals, which has 6^3 states in this example. The joint event model is further enhanced with joint events created from single events. To form a DFA $\mathcal{D}$ from the given specification DFAs, the robot uses $\mathcal{D} = \mathbf{M_I}(\mathbf{M_I}(\mathbf{M_S}(\mathcal{D}_1), \mathbf{M_S}(\mathcal{D}_2)), \mathbf{M_S}(\mathcal{D}_3))$.

Our implementation for this case study considers five settings: (1) RTM/FOM: the current state of the event model is always observable to the robot, that is, the robot always knows what each of the guests are doing, (2) RTM with a smoke detector device and a microphone: the robot is not aware what each of the guests are doing, but there is a smoke detector that if at each time step tells if somebody is smoking or not but without telling who is exactly smoking, and there is a microphone whose being turned on means that someone is dancing or being entertained; (3) RTM with a smoke detector device: the robot does not know what each of the guests are doing, but using the smoke detector can detect if right now somebody is smoking or not; (4) RTM with a microphone: the robot is not aware about the current status of the guests but if the microphone is turned on, then it means that someone is dancing or being entertained; (5) RTM/FHM: the robot receives no direct information about the current behavior of the guests.

For each of these five settings, we conducted two experiments, each consisting of 5,000 simulations. In one experiment we let the robot use the general algorithm to record a desired story, while in the other one we let the robot use the greedy algorithm.

The expected number of steps for an optimal policy for RTM/FOM is 35.38, and over the 5,000, simulations, the average number of steps to record a story using the general algorithm was 35.58; both numbers are very close. The average number of steps over 5,000 simulation using the greedy algorithm was 37.54, which shows that the greedy algorithm was also outperformed by the general algorithm in minimizing the number of steps to record for this case study.

The average number for RTM with a smoke detector and microphone using the general algorithm and the the greedy algorithm were 37.64 and 38.25, respectively. For RTM with a smoke detector, the average number was 37.95 when the general algorithm was used, and it was 38.06 when the greedy algorithm was used. The general algorithm and the greedy algorithm for RTM with a microphone respectively yielded 38.53 and 38.75. Finally, the average number of steps using the general algorithm for RTM/FHM was 39.20, while the average number using the greedy algorithm was 38.99.

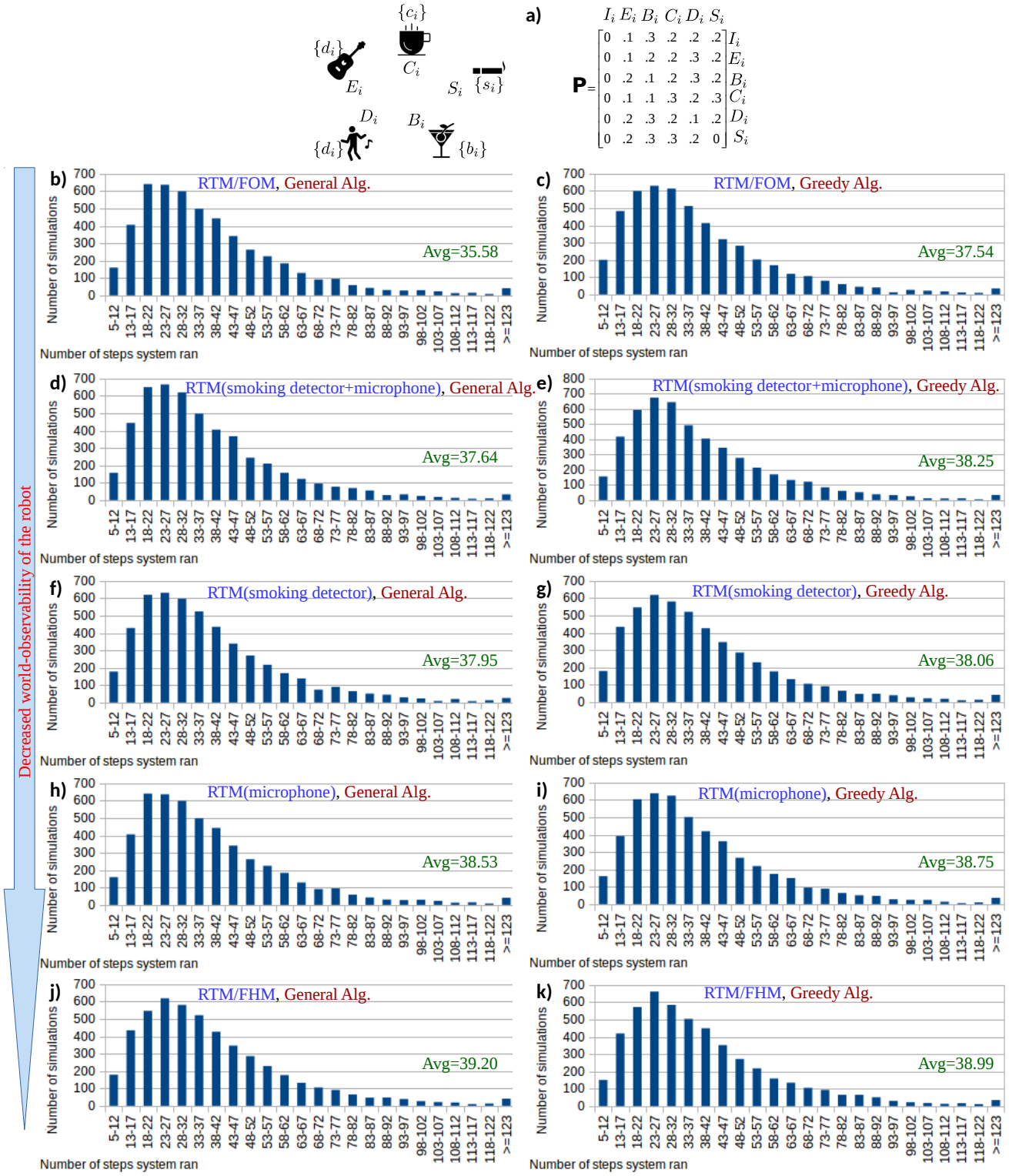

Figure 7. **a)** The event model for the behavior of a person attending a wedding reception, which has six states: I_i, the state of arriving; E_i, the state of being entertaining; C_i, for consuming coffee; B_i, for drinking other beverages; D_i, for dancing; and S_i, for smoking. Each histogram shows the average number of steps to record a desired story for 5,000 simulations of the wedding reception scenario. **b)** The general algorithm for RTM/FOM. **c)** The greedy algorithm for RTM/FOM. **d)** The general algorithm for RTM with a smoke detector, which provides the observation of whether someone is smoking, and with a microphone, capable of detecting that someone is dancing or being entertained. **e)** The greedy algorithm for RTM with a smoke detector and a microphone. **f)** The general algorithm for RTM with a smoke detector. **g)** The greedy algorithm for RTM with a smoke detector. **h)** The general algorithm for RTM with a microphone. **i)** The greedy algorithm for RTM with a microphone. **j)** The general algorithm for RTM/FHM. **k)** The greedy for algorithm RTM. It seems, at first, that the RTM problem with a smoke detector might be incomparable with RTM using a microphone, but the single useful observation in the former guarantees that at least one guest is in the state of smoking, while the single useful observation in the later case guarantees that at least one guest is either in state of dancing or in state of being entertained, which is less informative. This experiment also shows that increasing observability will decrease the time to capture a desired story. Furthermore, it shows that although the general algorithm often outperformed the greedy algorithm in terms of average number of steps to record a desirable event sequence, here the greedy algorithm gives a reasonable approximation to the optimal solutions.

Again we observed that increasing the robot's ability to perceive the world will help reduce the average number of steps to record a desirable event sequence. Also, for four out of the five considered settings, the general algorithm yielded a fewer average number of steps compared to the greedy algorithm, but for RTM/FHM, the greedy algorithm produced a slightly superior average number of steps. In this experiment, except for RTM/FOM, which we solve using an MDP rather than a POMDP, the expected number of steps for the greedy algorithm and for the general algorithm were close. This is perhaps because APPL Online, the tool we used for solving the POMDP, is an online POMDP solver and the solution is provide is an approximate solution rather than an exact solution, which is in general intractable to provide for POMDPs. This suggests this particular case study is an example where the greedy algorithm is able to closely approximate the optimal solutions.

9.3 Running a race

John and James are two runners that running a race against one another. A videographer robot is asked to record a video whose events involve John and James. The events of interest are: r_1, John is running; h_1, John is crossing the flag located the middle of the race field; f_1, John is crossing the finish line; p_{12}, John is passing James; r_2, James is running; h_2, James is crossing the flag located the middle of the race course; f_2, James is crossing the finish line; p_{21}, James is passing John.

To make an event model for this problem, we divide the racetrack into several sections of the identical length. Figure 8a shows an example in which the race is divided into 8 sections, s_0 through s_7. To make an event model for a single runner, we represent each of those sections using a single state of the event model. The transition probability function is based on the distance a runner can travel in a single time step, and as the sections form a sequence, their neighbor-to-neighbor connections.

Figure 8b shows the event model for a runner i where the track is in sections s_0–s_7. In this example, when the runner is in state s_j then, at the next time step, based on his speed, he could be in any of states s_j, s_{j+1}, s_{j+2}, and s_{j+3}. The event model for the joint behavior of John and James is formed from the product of their individual event models. Each state of this event model represents a tuple of sections of the field in which John and James could be. Events p_{12} and p_{21} both happen with probability 0.5 at each state representing a situation where both John and James are in one section of the track. The current state of the event model is observable by the robot: perhaps, at specific locations along the course, there are stationary cameras that tell the robot the sections the runners currently occupy. The robot does not, however, know the sections which John and James will be in in the next time step because it does not know how their speed will change in the future. Thus, to find an optimal policy for capturing events, we need to solve the RTM/FOM problem.

The desired story is specified by the DFA in Figure 8c. To solve this problem, we form the Goal MDP and we can use the (classical) value iteration to compute an optimal policy for the MDP. However, because the given DFA is loop-omitted acyclic, a better option would be to use the topological value iteration algorithm, introduced in Section 6.1, to compute an optimal policy for the MDP. Closer observation suggests that we can even use the algorithm introduced in Section 6.2 for loop-omitted DAG MDPs. This is because not only the DFA is loop-omitted acyclic, but the graph underlying the event model also does not have any cycles other than self-loops; these facts together make the Goal MDP a loop-omitted DAG MDP. We designate the algorithm that we introduced for solving loop-omitted MDPs, *single value iteration*, owing to the fact that only one iteration for each state is required to solve the Bellman equation for this kind of MDP, and the optimal action for each state is computed by solving several single variable equations.

In this experiment, we compare the running times of classic value iteration, topological value iteration, and the single value iteration algorithms to compute an optimal policy for the MDP. To do so, we consider 10 racetracks, varying the number of sections from 30 to 120. For each of these 10 problem sizes $\{30, 40, \ldots, 120\}$, we construct a Goal MDP from the DFA and the event model, and then compute an optimal policy for that MDP using each of the three algorithms. Figures 8d and 8e give a sense of the sizes of those MDPs; the former shows for each of those MDPs, the number of states of the MDP, while the latter shows the number of edges of the graph underlying the MDP.

For each of those 10 problem sizes, we conducted 10 trials for each of the three algorithms. Figure 8f shows, for each of those 10 problem sizes, the average computation time of each of the algorithms. The computation time for the topological value iteration in this diagram includes the computation for forming the SCCs of the graph and the time needed to find a topological ordering. Figure 8g shows how much time these steps contribute to the overall computation time of the topological value iteration algorithm. Likewise, Figure8h shows the contributions for the two phases that make up the single value iteration, namely, finding a topological ordering of the MDP states, and solving the single value equations to find optimal actions for states.

In this experiment, we observe that the topological value iteration outperforms the classical value iteration and the single value iteration outperforms both of them. For the last MDP, which has approximately 57,000 states and approximately 7,000,000 nonzero entries within its transition function, it took 174 seconds on average for the classical value iteration to compute an optimal policy. In contrast, the single value iteration on average took less than 3 seconds to compute an optimal policy. This experiment justifies the use of the algorithms we introduced in Section 6 for special inputs of our problem.

10 Conclusions and future work

We considered the problem of minimizing the expected time to record an event sequence satisfying a set of specifications. This was posed as the problem of computing an optimal policy in an associated Markov decision problem. Our implementation confirmed the fact that, by considering global implications, careful planning of observations can improve performance significantly. Also, having studied differing of levels of observability, the results support the intuition that as the robot's ability to perceive the world

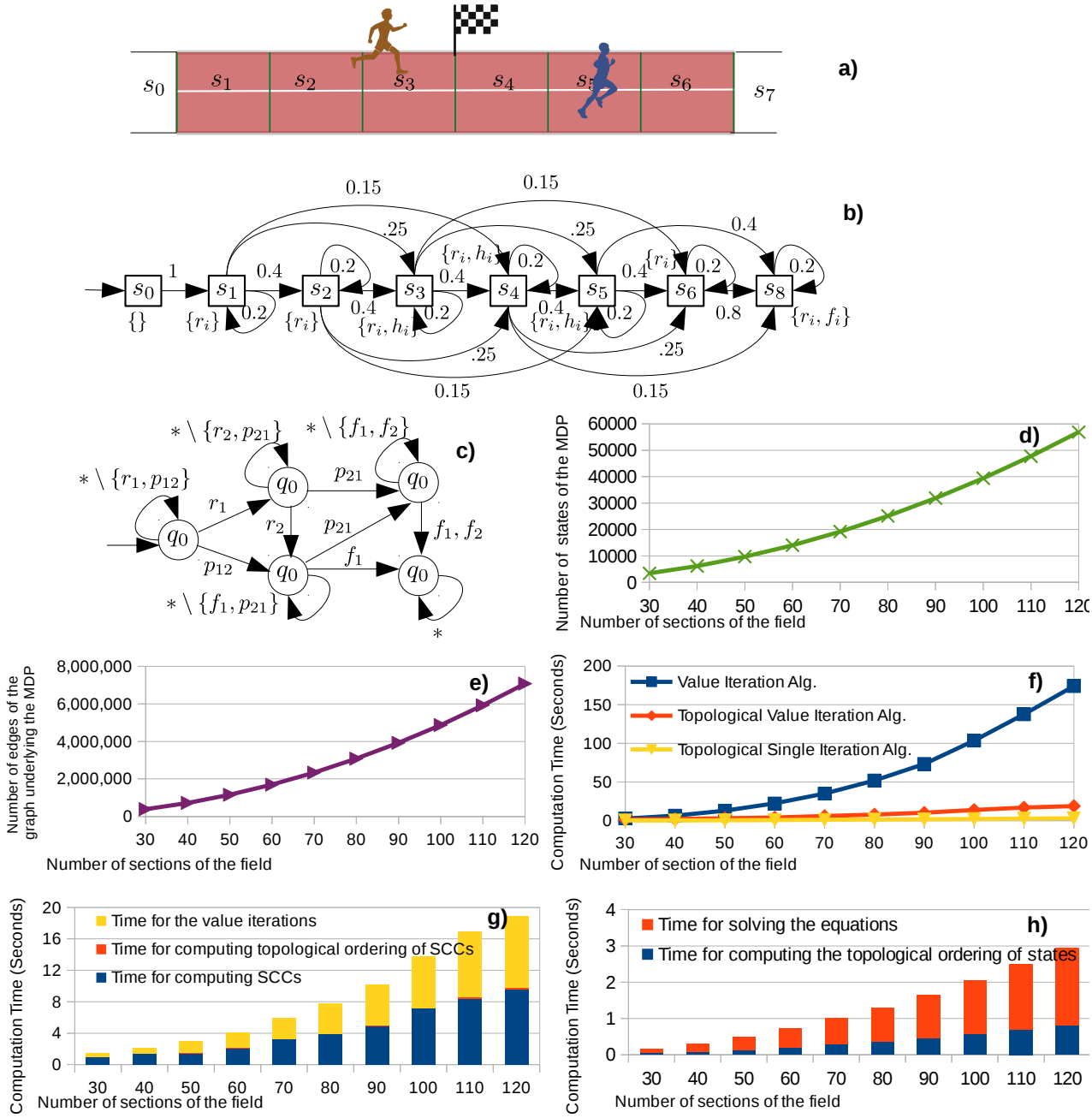

Figure 8. **a)** An example of a race source, which we have divided into 8 sections s_0 through s_7 form an event model for the race. **b)** The event model for a single runner $i \in \{1, 2\}$, including events r_i, runner i is running; h_i, runner i is crossing the race flag at the middle of the race field; f_i, runner i is crossing the finish line. The event model for the two runners John and James is made by the product of the event models for each of them. Two more events, p_{12} and p_{21}, are introduced to the joint event model. Event p_{12} denotes that John is overtaking James, while p_{21} means that James is overtaking John. Each of these two events happens with probability 0.5 at each state that represents the runners are in the same section. **c)** The DFA specifying the set of all desirable videos for the race example. **d)** A diagram showing the number of states of the MDP created by the product of the event model and the DFA in this race example. **e)** A diagram showing the number of edges of the graph underlying the MDP. Parts **d** and **e** together show how the size of the MDP increases as the size of the problem increases. They show as the problem's size doubles, the MDP's size approximately quadruplicates. **f)** A diagram showing the computation time for classical value iteration, which we use in our general algorithm, the computation time of the topological value iteration algorithm, which use to solve our problem where the DFA is loop-omitted acyclic, and the computation time of the topological value iteration, which we use to solve our problem when Goal MDP is a loop-omitted DAG. For this experiment, the topological single iteration was on average 6.5 faster than the topological value iteration and also the latter on average was 6 times faster than the general, classical value iteration algorithm. **g)** A diagram showing, for topological single value iteration, the breakdown of its computation time into the three phases of the algorithm: decomposing the MDP into its SCCs, finding a topological ordering of the SCCs, and performing the value iteration. **h)** A diagram showing, for the topological single value iteration algorithm, how much each of the two phases of the algorithm, namely, finding a topological ordering of the states of the MDP, and computing an optimal action for each state via solving the single variable equations, contribute to the computation time of the algorithm. The results for each of the section sizes 30-120 in the diagrams in parts **(f)**, **(g)**, and **(h)** are the average computation times across 10 trials.

improves, the expected number of steps to record a desired story decreases.

Gaps remain between the results presented in this article and the eventual use of real sensors aboard real robots to chronicle sequences of events. In particular, our approach relies heavily on the event model, which must abstract sufficient detail about the physical environment to model the occurrences of events. In addition, our approach abstracts the details of planning of the robot's physical movements in its attempt to capture events, but of course those details of motions may have a major impact on the liklihood of successfully capturing events. The applicability of our results to real systems depends directly on the practicability of resolving these abstract elements into fully-realized implementations within a complete system stack. Such a process seems likely to involve a number of challenges, including for example, the tradeoff between granularity of state (which, at some level, must model locations) and computational efficiency.

Future work could consider several extensions. For instance, considering in the means needed to navigate to record an event, so that the objective might minimize some expected cost rather than the expected number of steps. Also, one might consider the case where a set of events (rather than a single event), each assigned to a single robot, may be predicted. A more dynamic case might examine problems where the robot is given new specification DFAs to satisfy while recording stories for previous requests, especially where those specification DFAs are prioritized and the prioritization is subject to changes (Rahmani and O'Kane, 2019), or perhaps the case where the robot needs to learn a new event model to describe the environment owing to failure in predicting events. Thus, since the Goal MDPs/POMDPs we construct in this paper have particular structures, future work should consider designing more efficient algorithms to solve the specific forms of Goal MDP/POMDP that arise in this context, with an eye toward computing optimal polities more efficiently than the general-purpose algorithms utilized in this paper. Using the ideas of factored-MDPs (Boutilier et al., 2000; Guestrin et al., 2003), factored-POMDPs (Boutilier and Poole, 1996), and POMDP-lite (Chen et al., 2016) might be good starting points for doing so.

Acknowledgements

This work was graciously supported, in part, by the National Science Foundation through awards IIS-1453652, IIS-1849249, and IIS-1849291.

References

Alcántara A, Capitán J, Cunha R and Ollero A (2021) Optimal trajectory planning for cinematography with multiple unmanned aerial vehicles. *Robotics and Autonomous Systems* 140: 103778.

Astrom KJ (1965) Optimal control of markov decision processes with incomplete state estimation. *Journal of Mathematical Analysis and Applications* 10: 174–205.

Bajcsy R (1988) Active perception. *Proceedings of the IEEE* 76(8): 966–1005.

Bajcsy R, Aloimonos Y and Tsotsos JK (2018) Revisiting active perception. *Autonomous Robots* 42(2): 177–196.

Bonet B and Geffner H (2003) Faster heuristic search algorithms for planning with uncertainty and full feedback. In: *International Joint Conference on Artificial Intelligence*. pp. 1233–1238.

Bonet B and Geffner H (2009) Solving POMDPs: RTDP-Bel versus point-based algorithms. In: *International Joint Conference on Artificial Intelligence*. pp. 1641–1646.

Boutilier C, Dearden R and Goldszmidt M (2000) Stochastic dynamic programming with factored representations. *Artificial Intelligence* 121(1-2): 49–107.

Boutilier C and Poole D (1996) Computing optimal policies for partially observable decision processes using compact representations. In: *National Conference on Artificial Intelligence*. Citeseer, pp. 1168–1175.

Chaudhuri D, Ike R, Rahmani H, T Becker A, A Shell D and M O'Kane J (2021a) Conditioning style on substance: Plans for narrative observation. In: *IEEE International Conference on Robotics and Automation (ICRA)*.

Chaudhuri D, Rahmani H, A Shell D and M O'Kane J (2021b) Tractable planning for coordinated story capture: Sequential stochastic decoupling. In: *International Symposium on Distributed Autonomous Robotic Systems*.

Chen M, Frazzoli E, Hsu D and Lee WS (2016) POMDP-lite for robust robot planning under uncertainty. In: *IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, pp. 5427–5433.

Dai P and Goldsmith J (2007) Topological value iteration algorithm for Markov decision processes. In: *International Joint Conference on Artificial Intelligence*. pp. 1860–1865.

Dibangoye JS, Shani G, Chaib-Draa B and Mouaddib AI (2009) Topological order planner for POMDPs. In: *Twenty-First International Joint Conference on Artificial Intelligence*.

Girdhar Y and Dudek G (2012) Efficient on-line data summarization using extremum summaries. In: *IEEE International Conference on Robotics and Automation (ICRA)*. pp. 3490–3496.

Gong B, Chao WL, Grauman K and Sha F (2014) Diverse sequential subset selection for supervised video summarization. In: *Advances in Neural Information Processing Systems*. pp. 2069–2077.

Guestrin C, Koller D, Parr R and Venkataraman S (2003) Efficient solution algorithms for factored MDPs. *Journal of Artificial Intelligence Research* 19: 399–468.

Gygli M, Grabner H, Riemenschneider H and Van Gool L (2014) Creating summaries from user videos. In: *European Conference on Computer Vision*. pp. 505–520.

Ji Z, Xiong K, Pang Y and Li X (2019) Video summarization with attention-based encoder–decoder networks. *IEEE Transactions on Circuits and Systems for Video Technology* 30(6): 1709–1717.

Junges S, Jansen N and Seshia SA (2021) Enforcing almost-sure reachability in pomdps. In: *International Conference on Computer Aided Verification*. Springer, pp. 602–625.

Keyder E and Geffner H (2008) The HMDPP planner for planning with probabilities. *Sixth International Planning Competition at ICAPS* 8.

Kolobov A, M and Weld DS (2010) Sixthsense: Fast and reliable recognition of dead ends in MDPs. In: *Proceedings of the*

Twenty-Fourth AAAI Conference on Artificial Intelligence.

Kolobov A, Mausam and Weld DS (2012) A theory of goal-oriented MDPs with dead ends. In: *Twenty-Eighth Conference on Uncertainty in Artificial Intelligence*. AUAI Press, p. 438–447.

Konstantinidis S (2007) Computing the edit distance of a regular language. *Information and Computation* 205(9): 1307–1316.

LaValle SM (2006) *Planning Algorithms*. Cambridge, U.K.: Cambridge University Press. Available at http://planning.cs.uiuc.edu/.

Lee YJ, Ghosh J and Grauman K (2012) Discovering important people and objects for egocentric video summarization. In: *IEEE Conference on Computer Vision and Pattern Recognition*. pp. 1346–1353.

Levenshtein VI (1966) Binary codes capable of correcting deletions, insertions, and reversals. *Soviet Physics Doklady* 10(8): 707–710.

Little I and Thiébaux S (2007) Probabilistic planning vs. replanning. In: *ICAPS Workshop on International Planning Competition: Past, Present and Future*.

Lu Z and Grauman K (2013) Story-driven summarization for egocentric video. In: *IEEE Conference on Computer Vision and Pattern Recognition*. pp. 2714–2721.

Mademlis I, Mygdalis V, Nikolaidis N, Montagnuolo M, Negro F, Messina A and Pitas I (2019a) High-level multiple-uav cinematography tools for covering outdoor events. *IEEE Transactions on Broadcasting* 65(3): 627–635.

Mademlis I, Nikolaidis N, Tefas A, Pitas I, Wagner T and Messina A (2019b) Autonomous uav cinematography: A tutorial and a formalized shot-type taxonomy. *ACM Computing Surveys (CSUR)* 52(5): 1–33.

Mahasseni B, Lam M and Todorovic S (2017) Unsupervised video summarization with adversarial lstm networks. In: *IEEE Conference on Computer Vision and Pattern Recognition*. pp. 202–211.

Narasimhan M, Rohrbach A and Darrell T (2021) Clip-it! language-guided video summarization. *arXiv preprint arXiv:2107.00650* .

Plummer BA, Brown M and Lazebnik S (2017) Enhancing video summarization via vision-language embedding. In: *IEEE Conference on Computer Vision and Pattern Recognition*. pp. 5781–5789.

Rabin MO and Scott D (1959) Finite automata and their decision problems. *IBM Journal of Research and Development* 3(2): 114–125.

Rahmani H and O'Kane JM (2019) Optimal temporal logic planning with cascading soft constraints. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, pp. 2524–2531.

Rahmani H, Shell DA and O'Kane JM (2020) Planning to chronicle. In: *Algorithmic Foundations of Robotics (WAFR XIV)*.

Riedl MO and Young RM (2010) Narrative planning: Balancing plot and character. *Journal of Artificial Intelligence Research* 39: 217–268.

Robertson J and Young RM (2017) Narrative mediation as probabilistic planning. In: *Thirteenth Artificial Intelligence and Interactive Digital Entertainment Conference*.

Ross S, Pineau J, Paquet S and Chaib-Draa B (2008) Online planning algorithms for POMDPs. *Journal of Artificial Intelligence Research* 32: 663–704.

Rot J, Bonsangue M and Rutten J (2016) Proving language inclusion and equivalence by coinduction. *Information and Computation* 246: 62–76.

Sabetghadam B, Alcántara A, Capitán J, Cunha R, Ollero A and Pascoal A (2019) Optimal trajectory planning for autonomous drone cinematography. In: *European Conference on Mobile Robots (ECMR)*. IEEE, pp. 1–7.

Schulz KU and Mihov S (2002) Fast string correction with Levenshtein automata. *International Journal on Document Analysis and Recognition* 5(1): 67–85.

Shani G, Pineau J and Kaplow R (2013) A survey of point-based POMDP solvers. *Autonomous Agents and Multi-Agent Systems* 27(1): 1–51.

Shell DA, Huang L, Becker AT and O'Kane JM (2019) Planning coordinated event observation for structured narratives. In: *IEEE International Conference on Robotics and Automation (ICRA)*. pp. 7632–7638.

Somani A, Ye N, Hsu D and Lee WS (2013) Despot: Online POMDP planning with regularization. *Advances in Neural Information Processing Systems* 26: 1772–1780.

Sondik EJ (1978) The optimal control of partially observable markov processes over the infinite horizon: Discounted costs. *Operations Research* 26(2): 282–304.

Truong BT and Venkatesh S (2007) Video abstraction: A systematic review and classification. *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)* 3(1): 3–es.

Yu J and LaValle SM (2010) Cyber detectives: Determining when robots or people misbehave. In: *Algorithmic Foundations of Robotics (WAFR IX)*. Springer, pp. 391–407.

Yu J and LaValle SM (2011) Story validation and approximate path inference with a sparse network of heterogeneous sensors. In: *IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, pp. 4980–4985.

Zhang K, Grauman K and Sha F (2018) Retrospective encoders for video summarization. In: *European Conference on Computer Vision (ECCV)*. pp. 383–399.