Multiscale Time Abstractions for Long-Range Planning under Uncertainty

Roykrong Sukkerd
Institute for Software Research
Carnegie Mellon University
Pittsburgh, PA, USA
rsukkerd@cs.cmu.edu

Javier Cámara
Institute for Software Research
Carnegie Mellon University
Pittsburgh, PA, USA
jcmoreno@cs.cmu.edu

David Garlan
Institute for Software Research
Carnegie Mellon University
Pittsburgh, PA, USA
garlan@cs.cmu.edu

Reid Simmons
Robotics Institute
Carnegie Mellon University
Pittsburgh, PA, USA
reids@cs.cmu.edu

ABSTRACT

Planning in CPSs requires temporal reasoning to handle the dynamics of the environment, including human behavior, as well as temporal constraints on system goals and durations of actions that systems and human actors may take. The discrete abstraction of time in a state space planning should have a time sampling parameter value that satisfies some relation to achieve a certain precision. In particular, the sampling period should be small enough to allow the dynamics of the problem domain to be modeled with sufficient precision. Meanwhile, in many cases, events in the far future (relative to the sampling period) may be relevant to the decision making earlier in the planning timeline; therefore, a longer planning look-ahead horizon can yield a closer-to-optimal plan. Unfortunately, planning with a uniform fine-grained discrete abstraction of time and a long look-ahead horizon is typically computationally infeasible. In this paper, we propose a multiscale temporal planning approach – formulated as MDP planning – to preserve the required time fidelity of the problem domain and at the same time approximate a globally optimal plan. We illustrate our approach in a middleware used to monitor large sensor networks.

Keywords

Temporal planning; probabilistic planning; multiscale time abstractions

1. INTRODUCTION

Planning in smart cyber-physical systems requires reasoning about the dynamics of the environment (i.e., exogenous events expected to occur at some future time), including the behavior of human actors, as well as temporal constraints on system goals and durations of the actions or tactics that the system may use. Therefore, an explicit representation of time is required in a planning formalism. Probabilistic planning is appropriate for cyber-physical systems, especially because of the uncertainty in the behavior of the environment and human actors with whom CPSs interact. Markov decision process (MDP) is one of the most widely used formalisms to formulate probabilistic planning problems. For instance, the work in [8] presents a time modeling approach for MDP planning using uniform time discretization (i.e., all time steps have the same duration). This approach is particularly useful for handling (probabilistic) predictable exogenous events.

Planning may need to consider the environment dynamics that involve different time scales. Some aspects of the environment may change quickly, and planning requires a certain time fidelity to represent those dynamics sufficiently precisely. Meanwhile, some events occurring in the distant future (relative to the time scale of the fast dynamics) may be relevant to the decision making earlier in the planning timeline, since they reveal the opportunity, or the lack thereof, for tasks to be accomplished. Without considering such distant but relevant events, planning may yield a suboptimal solution. Such characteristics of a planning problem appear in CPSs – an example is a class of CPSs that cooperate with humans to achieve some goals. The dynamics of human behavior often involve multiple time scales, for instance, humans' attention may change quickly relative to their locations. Consider a system that must initiate interactions with humans at an appropriate time (e.g., when they are not attending to other important things) and when the humans are at certain locations. In order to have an optimal cooperation between the system and the humans, both the attention and location dynamics of the humans must be taken into account in planning.

To yield an optimal plan given an environment whose dynamics involve different time scales, state-space planning, such as MDP planning, requires a fine-grained time discretization and a long look-ahead horizon to model changes in the environment at a sufficient precision and to model

ACM ISBN 978-1-4503-2138-9.
DOI: 10.1145/1235

potentially relevant events in the distant future. Unfortunately, uniform-resolution time discretization makes such multiscale temporal planning computationally infeasible, due to the fast growth in the number of states as the length of a look-ahead horizon increases. The problem becomes worse as the number of actions increases.

To solve this problem, we can take advantage of the fact that prediction uncertainty of the environment and human behavior increases with time. Therefore, potentially relevant events in the distant future can be modeled in a coarser time resolution, capturing vagueness in the predicted time of occurrences of events and simplified dynamics. Moreover, it is not necessary to provide plans with a uniform high resolution, since the realization of the environment in the distant future may diverge from the original predictions, and replanning will be required. We propose a multi-resolution temporal planning approach, which uses a finer time resolution early in the planning timeline, and a coarser time resolution later. The coarser-resolution part of the plan will gradually be refined via replanning. This approach preserves the required time fidelity of the problem domain, and at the same time approximates a globally optimal plan by enabling a long look-ahead horizon for planning. We implement our planning approach for a human-in-the-loop adaptation scenario of a middleware used to monitor large sensor networks. Our preliminary results based on this case study show that (1) with the same planning horizon, multi-resolution temporal planning can generate plans whose utility is close to those generated by uniform-resolution temporal planning, but with much lower planning complexity, and (2) with the same amount of planning time, multi-resolution temporal planning can generate higher-utility plans than uniform-resolution planning by looking further ahead into the future.

The rest of the paper is organized as follows. Section 2 presents the motivating example. Section 3 describes the formal details of our approach. Section 4 presents the preliminary results and discussion. Section 5 discusses the related work in the areas of multi-resolution planning and scheduling. Section 6 concludes the paper.

2. MOTIVATING EXAMPLE

The Data Acquisition and Control Service (DCAS) [3] is a middleware from Critical Software that provides a reusable infrastructure to manage the monitoring of highly populated networks of devices equipped with sensors. In particular, the middleware is designed to be seamlessly integrated with Critical's Energy Management System (csEMS)[1], which is a platform that provides asset management support for power producing companies based on renewable energy sources. The overall csEMS architecture aims at high scalability, flexibility and customization with management capabilities that enable the operation of control centers independently of the underlying application (*e.g.*, wind, solar, etc.).

The basic building blocks in a DCAS-based system (Figure 1) are: [2]

- *Devices* are equipped with one or more sensors to obtain data from the application domain (e.g., from wind towers, solar panels, etc.). Each sensor has an associated *data stream* from which data can be read. Each type of device has its particular characteristics (e.g., data polling rate, or expected value ranges) specified in a *device profile*.
- *Processor nodes* pull data from the devices at a rate configured in the device profile, and dispatch this data to the database server. Each processor node includes a set of processes called *Data Requester Processor Pollers* (DRPPs or *pollers*, for short) responsible for retrieving data from the devices.
- *Database server* stores the data collected from devices by processor nodes.
- *Application server* is connected to the database server to obtain data, which can be presented to the operators of the system or processed automatically via software.

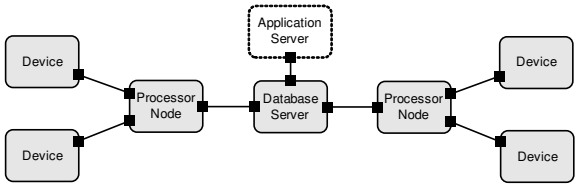

Figure 1: Architecture of a DCAS-based system

The main objective of DCAS is to collect data from the connected devices at a rate as close as possible to the one configured in their device profiles, while making an efficient use of the computational resources in the processor nodes. Specifically, the primary concern in DCAS is providing service while maintaining acceptable levels of performance, measured in terms of processed data requests per second (rps) inserted in the database.

DCAS has several self-adaptation mechanisms, one of which is called *scale-out*, which aims at maintaining an acceptable performance level when new devices are connected to the network at runtime and all available resources in the set of active processor nodes are already being used. One way to achieve this is by manually deploying a new processor node, and re-attaching some devices across the different processor nodes, according to the particular situation. Specifically, the scale-out process requires human operators to perform the following steps: (i) deploy new processor node(s), (ii) determine which of the new devices can be attached to a currently active processor node, and which must be attached to a new one, and attach the new devices to the corresponding processor node(s), (iii) re-attach devices that are already attached to other processor nodes to the new processor node(s), if necessary, (iv) activate the new processor node(s), and (v) restart active processor nodes that have been assigned new devices, if any.

Scale-out requires at least two human operators, who have different responsibilities in performing the tactic. Human operators have (potentially different) regular working schedules which do not include scaling out DCAS. Since the human operators are not always available to perform the tactic, the decision of when to have which human operators scale out DCAS must aim at not only the performance improvement of DCAS but also at minimizing the interruption to the operators. Formally, we define the utility of performing each (sub)task of the scale-out tactic, $task$, at a particular time, t, as the following function:

$$U(op, task, t) = w_I \cdot U_I(op, t) + w_T \cdot U_T(task, t) \quad (1)$$

where: (i) $U_I(op, t)$ is the utility of interrupting operator op at time t, (ii) $U_T(task, t)$ is the utility of the timeliness

[1]http://solutions.criticalsoftware.com/products_services/csems/
[2]Further details about DCAS can be found in [3].

of completing the scale-out tactic, and (iii) w_I and w_T are the constants representing the relative importance of operator interruption and timeliness of completing the scale-out tactic, respectively. Planning for scaling out DCAS aims at maximizing the utility function in Equation 1.

The interruption utility $U_I(op, t)$ in Equation 1 depends on the level interruptibility of operator *op* at time t, which can be predicted based on the regular work schedule of the operator. We employ five levels of interruptibility: *best, good, ok, bad, and worst* – each has a corresponding constant utility value. The interruptibility of an operator can change every few minutes depending on what the operator is doing; therefore, in order to model the operators' interruptibility with sufficient precision, the granularity of time discretization must be at a minute-level (e.g., the sampling period is 1 minute). To account for uncertainty in the prediction of the operators' interruptibility, we use a discrete-time Markov chain to model the dynamics of the interruptibility of each operator. Meanwhile, the timeliness utility $U_T(task, t)$ in Equation 1 is defined as:

$$U_T(task, t) = \begin{cases} 0 & \text{if } task \text{ is not the} \\ & \text{last step of scale-out} \\ 1 - \frac{t + \text{duration}(task)}{t_{end}} & \text{if } task \text{ is the} \\ & \text{last step of scale-out} \end{cases}$$

where t_{end} is the maximum time in the planning timeline. In this example, we employ the timeliness utility of completing the scale-out tactic as a proxy to the performance improvement of DCAS.

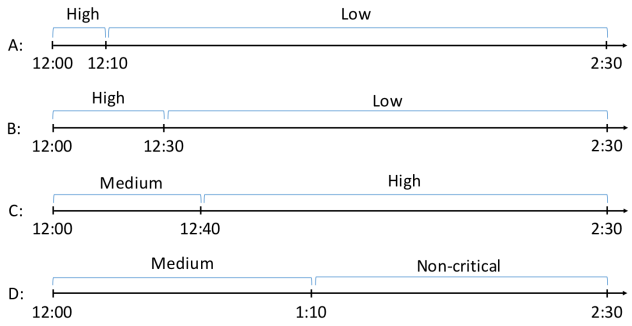

Figure 2: Criticality of tasks on operator work schedules

Consider the following scenario. Operators A and B are individually responsible for step 1 of scale-out (i.e., deploying processor nodes), and operators C and D are individually responsible for steps 2-5 of scale-out (i.e., (re)attaching devices to processor nodes and (re)starting processor nodes). Operators A and B can complete step 1 of scale-out in 30 minutes and 45 minutes, respectively, and both operators C and D can complete steps 2-5 of scale-out in 60 minutes. The regular work schedules of the operators are shown in Figure 2. Starting from 12:00pm, operator A is scheduled to work on a high-criticality task for the first 10 minutes and on low-criticality tasks for the next 140 minutes; operator B is scheduled to work on high-criticality tasks for the first 30 minutes and on low-criticality tasks for the next 120 minutes; operator C is scheduled to work on medium-criticality tasks for the first 40 minutes and on high-criticality tasks for the next 110 minutes; and operator D is scheduled to work on medium-criticality tasks for the first 70 minutes and on non-critical tasks for the next 80 minutes. Operators generally have lower interruptibility during tasks of higher criticality, and have higher interruptibility during tasks of lower criticality (although, the specific level of interruptibility varies within a task). If we choose the planning timeline to be from 12:00pm to 2:30pm, then the optimal plan for scaling out DCAS is to have operator B perform step 1 of scale-out at around 12:30pm and operator D perform steps 2-5 of scale-out at around 1:10pm. This plan completes the scale-out tactic at an acceptable time (around 2:10pm), and interrupts operators B and D when they both have a relatively high interruptibility.

However, planning at the required time granularity of 1 minute is computationally infeasible for a look-ahead horizon of 150 minutes, due to a large state space. If we choose a shorter look-ahead horizon of 120 minutes (i.e., the planning timeline is from 12:00pm to 2:00pm) which is more feasible, then the optimal plan for scaling out DCAS with respect to that timeline is to have operator A perform step 1 of scale-out at around 12:10pm and operator C perform steps 2-5 of scale-out at around 12:40pm. Unfortunately, this plan is less desirable because it interrupts operator C when she has a very low interruptibility.

This example illustrates a problem class for which multi-resolution temporal planning is suitable. The dynamics of the DCAS operators' interruptibility require a time granularity of a minute to be represented precisely. At the same time, the predicted interruptibility of operator D in the relative distant future (from 1:10pm to 2:30pm) is relevant to the decision making earlier in the planning timeline, since it reveals an opportunity to complete the scale-out tactic with a significantly less interruption to the operators. Planning at a uniform time resolution with a long look-ahead horizon is computationally infeasible due to a large state space, and the problem becomes even worse as the number of actions increases (e.g., more DCAS operators).

In the next section, we describe our approach of probabilistic, multi-resolution temporal planning.

3. MULTI-RESOLUTION TEMPORAL PLANNING

The scope of temporal planning that we consider in this paper involves: (i) durative actions, (ii) exogenous events predicted to occur at some future time, (iii) time-dependent objectives (e.g., problems in which time is a relevant dimension of a potentially multi-dimensional utility function), and (iv) probabilistic behavior. In this section, we formalize temporal planning with probabilistic exogenous events as a Markov decision process. We start by describing the approach to formalize temporal planning as MDP using uniform time discretization. Then, we extend the approach to support multi-resolution time discretization, and discuss a replanning strategy for multi-resolution temporal planning.

3.1 Temporal Planning as MDP

In this section, we formalize a temporal planning problem with probabilistic exogenous events as a Markov decision process.

Definition 1 (Markov decision process). *A Markov decision process (MDP) is a tuple $\mathcal{M} = \langle S, s_I, A, \Delta, r \rangle$, where $S \neq \emptyset$ is a finite set of states; $s_I \in S$ is an initial state; $A \neq \emptyset$ is a finite set of actions; $\Delta : S \times A \to \mathcal{D}(S)$ is a (partial) probabilistic transition function; and $r : S \times A \to \mathbb{Q}_{\geq 0}$ is*

a reward structure mapping each state and action to a non-negative rational reward. $\mathcal{D}(X)$ denotes the set of discrete probability distributions over finite set X.

An MDP models how the state of a system can evolve in discrete time steps. In each state $s \in S$, the set of enabled actions is denoted by $A(s)$ (we assume that $A(s) \neq \emptyset$ for all states). Moreover, the choice of which action to take in every state s is assumed to be nondeterministic. Once an action a is selected, the successor state is chosen according to probability distribution $\Delta(s, a)$.

We can reason about the behavior of MDPs using policies. A policy resolves the nondeterministic choices of an MDP, selecting which action to take in every state.

DEFINITION 2 (POLICY). *A policy of an MDP is a function $\sigma : (SA)^*S \to \mathcal{D}(A)$ s.t., for each path $\pi \cdot s$, it selects a probability distribution $\sigma(\pi \cdot s)$ over $A(s)$.*

In this paper, we use policies that are *memoryless* (i.e., based solely on information about the current state) and *deterministic* ($\sigma(s)$ is a Kronecker function such that $\sigma(s)(a) = 1$ if action a is selected, and 0 otherwise).

3.1.1 State Structure

The set of states S is determined by a number of relevant properties of the system and the environment, characterized by a collection of random variables:

$$\{e_1, \ldots, e_m, c_1, \ldots, c_n, p_1, \ldots, p_k, t\}$$

that can be monitored at run time, where: (i) $e_1 \ldots e_m$ is a set of environment variables, (ii) $c_1 \ldots c_n$ is a set of system configuration variables, (iii) $p_1 \ldots p_k$ is a set of progress variables for all durative actions in A (see Section 3.1.2), and (iv) t is a variable representing the time of the state. First, we consider a uniform time discretization whose sampling period is τ. The values of the time variable t are then $t_0, t_0 + \tau, t_0 + 2\tau, ..., t_{end}$, where t_0 is the start time and t_{end} is the planning horizon.

3.1.2 Actions and Transition Function

The set A contains the actions available to the system, including a no-op action to allow the possibility of simply waiting for the next decision-epoch. Each action has the associated duration. However, to allow modeling of exogenous events that may occur during the executions of actions, all transitions in the MDP occur between states of two consecutive time values. For every action $a \in A$ whose duration – denoted as $d(a)$– is larger than the sampling period τ, there is a corresponding progress variable $p \in [0, \lceil d(a)/\tau \rceil)$ to keep track of the progress of the action at every decision-epoch. The no-op action has the duration equal to the sampling period τ, and it is applicable in all states in which no other actions are in progress. The probabilistic transition function $\Delta : S \times A \to \mathcal{D}(S)$ describes the effects of the actions on the system configuration and the evolution of the environment state, both as a result of the actions and independently of the actions. The prediction of the environment evolution can be made using an environment predictor, e.g., an autoregressive (AR) time series predictor, to predict the environment state at every time period τ up to the planning horizon t_{end}.

3.1.3 Reward Structure

The reward structure $r : S \times A \to \mathbb{Q}_{\geq 0}$ defines the reward obtained by executing (or continuing the execution of) a particular action, including no-op, in a particular state. In this paper, we assume that the objectives of the planning problem are captured as a reward, defined as a multi-dimensional utility-based function, where each dimension represents system qualities that correspond to different business concerns (e.g., performance, cost). For all states s and all actions a except no-op, we define the reward structure $r(s, a) = \sum_i w_i r_i(s, a)$, where $r_i : S \times A \to \mathbb{Q}_{\geq 0}$ corresponds to the reward for the i^{th} dimension of the planning problem, and w_i is the weight for the i^{th} dimension, which represents the importance of that dimension in the system's objectives (e.g., if performance is given more importance than cost, it will be given a higher weight). We define the reward $r(s, \text{no-op}) = 0$ for all states s. We seek to find an optimal policy, σ^*, that maximizes the expected accumulative reward until a state whose time value is t_{end} is reached.

3.2 Multi-Resolution Temporal Planning

Far future states may be relevant to the decision making regarding what actions to perform in the current moment or in the near future. A long planning horizon may reveal opportunities, or the lack thereof, for tasks to be accomplished in the future, which allows for planning to analyze the tradeoffs among executing different actions at different points in time. In some problem domains, it may be desirable to have a planning horizon be larger in order of magnitude than the appropriate sampling period of time discretization of the domains – as determined by the dynamics of the environment, the durations of actions, etc. For instance, the appropriate sampling period may be minutes, while the desirable planning horizon may be several hours into the future. However, planning using a uniform time discretization and a relatively long planning horizon is likely computationally infeasible, especially when the action space is large.

An important observation is that the prediction uncertainty increases with time. An environment predictor may not be able to determine precisely when certain events will occur, but it may predict (possibly with relatively high accuracy) that the events will occur at some point during certain intervals of time. We can utilize this fact by using a larger sampling period of time discretization in the far future, in order to reduce the growth in the number of states as we use longer planning horizons. In this section, we describe our proposed multi-resolution temporal planning approach formulated in the MDP framework, where the lengths of the sampling periods of time discretization increase with time. In this paper, we demonstrate a 2-resolution temporal planning scheme; however, our approach can be generalized to arbitrary n-resolution temporal planning.

3.2.1 State Structure

Our multi-resolution time discretization strategy employs two different sampling parameters: τ_f and τ_c. For simplicity, we choose τ_c to be a multiple of τ_f. τ_f is an appropriate time granularity for the problem domain – determined by the dynamics of the environment, durations of actions, etc., and τ_c can be much larger than τ_f to allow a long planning horizon. τ_f-discretization is used from the start of the planning timeline t_0 to a near-future time t_{exec}, which we refer to as the execution horizon, and τ_c-discretization is used from t_{exec} to the planning horizon t_{end} (see Figure 3).

Since τ_f is a relatively small period for the problem domain, we assume that the environment may only evolve

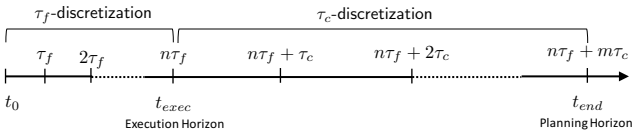

Figure 3: 2-resolution planning timeline

from one state to another during τ_f. Therefore, we construct the states to represent the system configurations and the (predicted) environment states at time $t_0, t_0 + \tau_f, t_0 + 2\tau_f, ..., t_{exec}$. That is, the structure of these states is as described in Section 3.1.1. However, from time $t_{exec} + \tau_c$ onwards, we construct the states to capture the intervals of time $[t_{exec}+\tau_c, t_{exec}+2\tau_c), [t_{exec}+2\tau_c, t_{exec}+3\tau_c), ..., [t_{end}-\tau_c, t_{end})$. This is because during a period τ_c, the environment is likely to evolve through multiple states. We do not construct the states to represent the environment states at time $t_{exec}+\tau_c, t_{exec}+2\tau_c, ..., t_{end}$, for 2 reasons: (1) there is high uncertainty in the prediction of the environment states in the far future (e.g., after time $t_{exec}+\tau_c$), and (2) completely disregarding the intermediate environment states during τ_c in planning likely results in highly suboptimal policies.

Since the environment can be in multiple states during an interval $[t, t + \tau_c)$, the structure of the states after time t_{exec} differs from that of the preceding states. In particular, the subset of states $S' \subset S$ after time t_{exec} is determined by a collection of random variables $\{E, c_1, \ldots, c_n, p_1, \ldots, p_k, t\}$, where: (i) E is a sequence of sets of the environment variables $e_1 \ldots e_m$, (ii) $c_1 \ldots c_n$ is a set of system configuration variables, (ii) $p_1 \ldots p_k$ is a set of progress variables for all durative actions (see Section 3.2.2), and (iv) t is a variable representing the time interval $[t, t+\tau_c)$ of the state. The sequence E of sets $e_1 \ldots e_m$ represents a total ordering (based on time) of the predicted environment states over the time interval $[t, t+\tau_c)$. Note that although we represent multiple predicted environment states over the interval τ_c in the state structure, we only represent a single system configuration, since actions can only execute once in a state.

3.2.2 Actions and Transition Function

The set A of an MDP with multi-resolution time discretization is the same as that of an MDP with uniform time discretization, except that the no-op action has duration τ_f and τ_c if executed before time t_{exec} and at time t_{exec} onwards, respectively. For every action a whose duration is $d(a) > \tau_f$ (except the no-op), there is a corresponding progress variable $p \in [0, \lceil d(a)/\tau_f \rceil)$ to keep track of the progress of the action at every decision-epoch. However, each transition of an action updates the corresponding progress variable differently depending on the time of the state in which the action is executed. The progress variable p of an action a advances by 1 if a is executed before time t_{exec}, and p advances by $\min(\lceil d(a)/\tau_f \rceil - p, \tau_c/\tau_f)$ if a is executed at time t_{exec} and after. Similar to the case of a uniform time discretization, the probabilistic transition function $\Delta : S \times A \to \mathcal{D}(S)$ describes the effects of the actions on the system configuration and the environment state, and the predicted evolution of the environment state independently of the actions. Since each state at time $t_{exec} + \tau_c$ onwards represents an interval of time, a transition from such state s to s' describes the evolution of the environment, regardless of the system's actions, from the end of the time interval of s to the beginning of the time interval of s'.

3.2.3 Reward Structure

We define the reward structure $r^f(s, a)$ for all states s at time $t \leq t_{exec}$ (i.e., in the finer time resolution) and for all actions a, as described in Section 3.1.3. We define the reward structure $r^c(s, a)$ for all states s at time $t \geq t_{exec}+\tau_c$ (i.e., in the coarser time resolution) and for all actions a. Since we seek to find an optimal policy σ^* that maximizes the expected accumulative reward until t_{end} is reached, we want the reward structure r^c to represent the fact that, if we were to schedule an action a – with the exception of no-op – to be executed in some time interval $[t, t + \tau_c)$, a can be executed at the point in the interval such that it yields the maximum utility of the system. Additionally, r^c should capture the minimum utility of the system if we were to not execute any action during the interval. To achieve that, we define the reward structure $r^c(s, a)$ as follows. Suppose s has the values $E, c_1, \ldots, c_n, p_1, \ldots, p_k, t$.

- If a is a no-op action, then $r^c(s, \text{no-op}) = 0$.
- If a has already started in s, i.e., the corresponding progress variable of a in s is $p > 0$, then $r^c(s, a)$ is the utility of continuing the execution of a in the beginning of the interval $[t, t + \tau_c)$ of s. Formally,
$$r^c(s, a) = r(\hat{s}_0, a) = \sum_i w_i r_i(\hat{s}_0, a),$$
where r_i and w_i are the reward function and the weight of the i^{th} quality dimension, respectively, and $\hat{s}_0$ has the values $e^0_1, \ldots, e^0_m, c_1, \ldots, c_n, p_1, \ldots, p_k, t$, where the set $e^0_1, \ldots, e^0_m$ is the first element in the sequence E.
- If a has not started in s, then $r^c(s, a)$ is the maximum utility of executing a at some point in the interval $[t, t+\tau_c)$ of s. Formally,
$$r^c(s, a) = \max_{\hat{s}} r(\hat{s}, a) \ = \max_{\hat{s}} \sum_i w_i r_i(\hat{s}, a),$$
where each $\hat{s}$ has the values $e_1, \ldots, e_m, c_1, \ldots, c_n, p_1, \ldots, p_k, t$, where the set $e_1, \ldots, e_m$ is one of the elements in the sequence E.

However, if the utility of executing an action a – with the exception of no-op – is associated with the duration of a, then the reward $r^c(s, a)$ must be adjusted accordingly. In particular, if a has already started in s with progress p, then the reward is
$$r^c(s, a) = l \cdot r(\hat{s}_0, a) = l \cdot \textstyle\sum_i w_i r_i(\hat{s}_0, a)$$
where $l = \min(\lceil d(a)/\tau_f \rceil - p, \tau_c)$. Otherwise, if a has not started in s, then the reward is
$$r^c(s, a) = l' \cdot \max_{\hat{s}} r(\hat{s}, a) \ = l' \cdot \max_{\hat{s}} \sum_i w_i r_i(\hat{s}, a)$$
where $l' = \min(\lceil d(a)/\tau_f \rceil, \tau_c)$.

3.3 Replanning

A long planning horizon allows planning to yield a more optimal plan compared to a short planning horizon, because it considers opportunities, or the lack thereof, for tasks to be accomplished in the far future. However, since the prediction uncertainty increases as we look further into the future, the scheduled executions of actions in the far future may in fact be suboptimal or inapplicable as the realization of the environment diverges from the prediction used in the planning. Hence, replanning is required as we move forward in

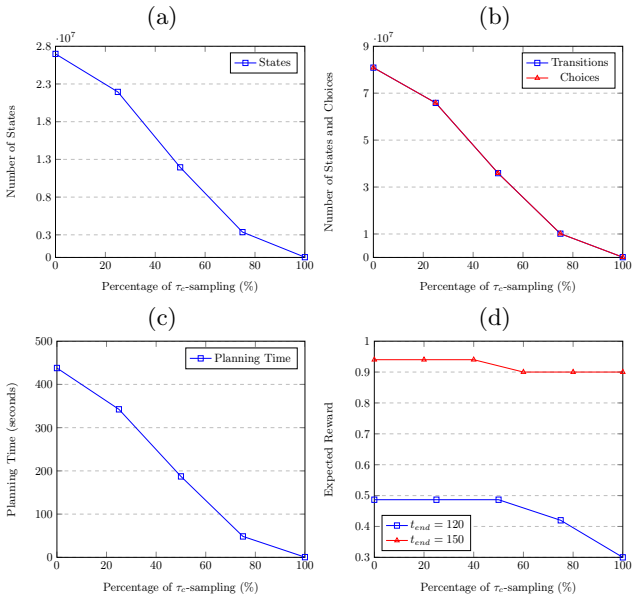

Figure 4: Percentage of τ_c-sampling vs: (a) number of states, (b) transitions and choices, (c) planning time, and (d) expected reward

time – to update the plan as we obtain new observations. For our multi-resolution temporal planning scheme, replanning serves the purpose of refining the coarser part of the plan in addition to updating the plan. We execute an optimal policy σ^*, generated from an MDP with multi-resolution time discretization, from the start time up to a maximum of the execution horizon. As we move closer to the execution horizon, we replan using a new prediction obtained from new observations.

4. PRELIMINARY RESULTS

We implement our multi-resolution temporal MDP in the probabilistic model checker PRISM [6]. The overall structure of our PRISM model is adopted from the work by Moreno et al. on proactive latency-aware self-adaptation under uncertainty [8] – with the distinction in the use of multiscale time abstraction. We conduct a set of preliminary experiments to evaluate our multi-resolution temporal planning approach and compare it to uniform-resolution temporal planning, using the example described in Section 2. The scenario we use in the experiments involves 6 DCAS operators – 4 of whom are described in Section 2. For uniform-resolution temporal planning, we use a sampling parameter $\tau = 1$ minute, and for multi-resolution temporal planning, we employ 2-resolution time discretization with sampling parameters $\tau_f = 1$ minute and $\tau_c = 30$ minutes. In this section, we discuss the preliminary results.

4.1 Planning Complexity

Figures 4(a) - 4(c) show the reduction of planning complexity achieved by our approach. Figure 4(a) shows the number of states in MDP as we vary the percentage of the use of coarse-grained time sampling (sampling period = τ_c) in the 2-resolution time discretization, with the planning horizon $t_{end} = 120$. For instance, 25% τ_c-sampling means that τ_f-sampling is used from $t = 0$ to $t = 90$, and τ_c-sampling is used from $t = 90$ to $t = 120$. Similarly, Figure 4(b) and Figure 4(c) show the number of transitions and choices in MDP, and the amount of planning time took to solve the MDP, as we vary the percentage of the use of coarse-grained time sampling.

	Uniform-resolution	Multi-resolution
States	35267862	35267862
Planning Horizon	121	150
Sampling Parameters	$\tau = 1$	$\tau_f = 1$ $\tau_c = 30$ $t^{max}(\tau_f) = 120$
Expected Reward	0.486666667	0.94

Table 1: Expected rewards of optimal policies generated from uniform- and multi-resolution temporal planning, provided both approaches use the same number of states

4.2 Coarse-grained Approximation

The use of a coarse-resolution time sampling in an MDP can result in suboptimal policies compared to a fine-resolution time sampling. For instance, any two actions that have an ordering dependency would be scheduled far apart in time – the two actions are scheduled in two different, relatively large time periods – while if the time sampling were finer-grained, the two actions may be scheduled much closer in time. This reduces the number of possible solutions, resulting in suboptimal policies. However, the sub-optimality may not be severe, given that we choose the right parameter of the coarse-resolution sampling. How to systematically determine an appropriate granularity of the coarse-resolution sampling remains a future work.

In our preliminary experiments, we observe how the expected reward of a policy changes with varying percentage of the use of coarse-grained time sampling (sampling period = τ_c) in the 2-resolution time discretization, as shown in Figure 4(c). We observe two sets of policies generated from the 2-resolution temporal planning using the planning horizons $t_{end} = 120$ and $t_{end} = 150$. For both sets of policies, there is a relatively small decrease in the expected reward as the percentage of use of coarse-grained timeline increases.

4.3 Long-horizon Planning

Given that both planning approaches generate the same number of states, the multi-resolution temporal planning can potentially find better plans in the long run compared to the uniform-resolution approach, by looking further ahead into the future. Table 1 shows the results of the particular scenario of DCAS system described in Section 2. Given that both approaches generate the same number of states, the 2-resolution temporal planning, which uses 20% τ_c-sampling and the planning horizon $t_{end} = 150$, generates a policy with a significantly higher expected reward (+93%) than a policy generated from the uniform-resolution planning, which has the planning horizon $t_{end} = 121$. The policies generated from both approaches are described in Section 2.

5. RELATED WORK

Several work in mobile robot path planning and motion planning employ multi-resolution state lattices to reduce the complexity of the global search while still providing high-quality solutions. Likhachev et el. [7] proposed an algorithm for generating dynamically-feasible maneuvers for au-

tonomous vehicles traveling over large distances. Their work considers multi-resolution discretization of the 2D position, orientation, and speed of a vehicle. Steffens et al. [11] proposed an approach to motion planning for service robots with multi-resolution in time, and Petereit et al. [10] proposed a hybrid-dimensional multi-resolution state and time lattice for mobile robot motion planning. Although some of these approaches address multi-resolution discretization of time, they differ from our work in that they do not consider exogenous events in multiscale abstraction of time.

Variable resolution discretization is also used in optimal control and safe control problems. Munos et el. [9] proposed several local splitting criteria for a variable resolution discretization. Cámara et al. [4, 5] proposed an approach for multiscale discrete abstractions for controller synthesis of switched systems. The focus of our work differs from theirs in that we use multiscale time abstraction to leverage the far future prediction to make optimal decisions for the current moment and for the near future.

Multi-resolution scheduling is closely related to our work. For instance, Bakirtzis et al. [1, 2] proposed an approach for operations scheduling using finer time resolution and coarser time resolution in the first and latter hours of the scheduling horizon. While their multiscale time sampling scheme is similar to ours, the novelty of our work is in the semantics of the utility model for multiscale temporal planning.

6. CONCLUSION

We propose a probabilistic, multi-resolution temporal planning approach for a class of planning problems which must reason about an environment whose dynamics involve different time scales. In particular, an environment may exhibit both fast and slow dynamics – the faster dynamics require a high time resolution of the planning model to represent the changes at a sufficient precision; meanwhile, the slower dynamics require a long look-ahead horizon for planning to consider potentially relevant events in the distant future. An example problem class of CPSs that have the multi-time scale characteristics is a class of CPSs that cooperate with humans to achieve some goals, since the dynamics of human behavior often involve different time scales. In this paper, we demonstrate our planing approach on a human-in-the-loop adaptation scenario. The preliminary results show that our multi-resolution temporal planning can yield higher-utility plans compared to the uniform-resolution approach.

One limitation of the proposed multi-resolution temporal planning approach is that rewards (or utilities) can only be associated with executions of actions, and not with states. In some applications, it is important to consider the rewards associated with the states of the system and the environment, which we plan to include in future work.

7. ACKNOWLEDGMENTS

This work is supported in part by Bosch Research and Technology Center North America, the National Science Foundation (while Dr. Simmons was serving there), by AFRL and DARPA under agreement number FA8750-16-2-0042, and the U.S. Department of Defense under Contract No. FA8721-05-C-0003 with Carnegie Mellon University for the operation of the Software Engineering Institute. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the Software Engineering Institute, Carnegie Mellon, the AFRL, DARPA, the United States Department of Defense, ASD(R&E), or the U.S. Government.

8. REFERENCES

[1] E. A. Bakirtzis, P. N. Biskas, and A. G. Bakirtzis. Dynamic reserves quantification for variable time resolution scheduling. In *PowerTech, 2015 IEEE Eindhoven*, pages 1–6. IEEE, 2015.

[2] E. A. Bakirtzis, P. N. Biskas, D. P. Labridis, and A. G. Bakirtzis. Multiple time resolution unit commitment for short-term operations scheduling under high renewable penetration. *Power Systems, IEEE Transactions on*, 29(1):149–159, 2014.

[3] J. Cámara, P. Correia, R. de Lemos, D. Garlan, P. Gomes, B. R. Schmerl, and R. Ventura. Evolving an adaptive industrial software system to use architecture-based self-adaptation. In M. Litoiu and J. Mylopoulos, editors, *Proceedings of the 8th International Symposium on Software Engineering for Adaptive and Self-Managing Systems, SEAMS 2013, San Francisco, CA, USA, May 20-21, 2013*, pages 13–22. IEEE Computer Society, 2013.

[4] J. Cámara, A. Girard, and G. Gößler. Safety controller synthesis for switched systems using multi-scale symbolic models. In *Proceedings of the 50th IEEE Conference on Decision and Control and European Control Conference, CDC-ECC 2011, Orlando, FL, USA, December 12-15, 2011*, pages 520–525, 2011.

[5] J. Cámara, A. Girard, and G. Gößler. Synthesis of switching controllers using approximately bisimilar multiscale abstractions. In *Proceedings of the 14th ACM International Conference on Hybrid Systems: Computation and Control, HSCC 2011, Chicago, IL, USA, April 12-14, 2011*, pages 191–200, 2011.

[6] M. Z. Kwiatkowska, G. Norman, and D. Parker. PRISM 4.0: Verification of probabilistic real-time systems. In G. Gopalakrishnan and S. Qadeer, editors, *Computer Aided Verification - 23rd International Conference, CAV 2011, Snowbird, UT, USA, July 14-20, 2011. Proceedings*, volume 6806 of *Lecture Notes in Computer Science*, pages 585–591. Springer, 2011.

[7] M. Likhachev and D. Ferguson. Planning long dynamically feasible maneuvers for autonomous vehicles. *I. J. Robotic Res.*, 28(8):933–945, 2009.

[8] G. A. Moreno, J. Cámara, D. Garlan, and B. R. Schmerl. Proactive self-adaptation under uncertainty: a probabilistic model checking approach. In E. D. Nitto, M. Harman, and P. Heymans, editors, *Proceedings of the 2015 10th Joint Meeting on Foundations of Software Engineering, ESEC/FSE 2015, Bergamo, Italy, August 30 - September 4, 2015*, pages 1–12. ACM, 2015.

[9] R. Munos and A. W. Moore. Variable resolution discretization for high-accuracy solutions of optimal control problems. In *Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence, IJCAI 99, Stockholm, Sweden, July 31 - August 6, 1999. 2 Volumes, 1450 pages*, pages 1348–1355, 1999.

[10] J. Petereit, T. Emter, and C. W. Frey. Mobile robot motion planning in multi-resolution lattices with hybrid dimensionality. In *Proceedings of the IFAC Intelligent Autonomous Vehicles Symposium*, pages 546–563, 2013.

[11] R. Steffens, M. Nieuwenhuisen, and S. Behnke. Continuous motion planning for service robots with multiresolution in time. In *Intelligent Autonomous Systems 13 - Proceedings of the 13th International Conference IAS-13, Padova, Italy, July 15-18, 2014*, pages 203–215, 2014.