

Mathematical Programming for Deliberation Scheduling in Time-limited Domains

Jianhui Wu*

EECS Department, University of Michigan
Ann Arbor, MI 48109 USA
jianhuiw@umich.edu

Edmund H. Durfee

EECS Department, University of Michigan
Ann Arbor, MI 48109 USA
durfee@umich.edu

ABSTRACT

Deliberation scheduling is the process of scheduling decision procedures to maximize overall performance. In this paper, we present novel mathematical programming algorithms for scheduling deliberations, and illustrate them through several increasingly complex classes of deliberation scheduling problems. In comparison to previous work, our algorithms are able to optimally or near-optimally solve deliberation scheduling problems for decision procedures that have more general performance profiles, and are applicable in complex domains with uncertainty. We also illustrate that, thanks to the mathematical programming formulation, our algorithms can be easily extended to model additional system aspects.

Categories and Subject Descriptors

I.2.8 [ARTIFICIAL INTELLIGENCE]: Problem Solving, Control Methods, and Search

General Terms

Algorithms

Keywords

Deliberation scheduling, control of reasoning, piecewise linear approximation, linear programming, mixed integer linear programming

1. MOTIVATION AND INTRODUCTION

Many complex problems are too difficult for autonomous agents to solve within deadlines. For example, an autonomous aircraft flying a prolonged mission might not have time to prepare a plan that specifies actions and reactions for all possible contingencies over the entire mission before it starts to execute the plan. In this situation, if the problem can be divided into multiple independent sub-problems (such as

takeoff, ingress, rescue, egress, and landing), which we call *phases*, then the agent might do better to focus only on near-term phases. Then, while executing the plan for earlier phases, the agent could use available computational time during execution to reconsider aspects of the problem and improve its solutions for current and future phases. This assertion is not surprising. The challenge, though, is in automating the process of allocating computational time to appropriate phases given the uncertainty and complexity of the problem domain.

Deliberation scheduling is the process of scheduling decision procedures to maximize overall performance. Boddy and Dean proposed an optimal deliberation scheduling algorithm for a subclass of problems with piecewise linear concave value functions [2, 1]. Horvitz considered likelihood of future instances, and explored policies for proactive allocation of idle time for these potential problems in several settings [6]. Goldman, Musliner, and Krebsbach proposed a greedy deliberation scheduling algorithm and compared its performance with the optimal algorithm using Markov Decision Process models [5, 7]. However, these approaches are either only applicable in restricted subclasses of deliberation scheduling problems (e.g., for limited types of performance profiles for decision procedures), or ignore some aspects of systems (e.g., uncertainty, or deliberation cost).

Our work is directed at addressing these shortcomings. Our formulation of the deliberation scheduling problem as a mathematical program—as a linear program (LP) or mixed-integer linear program (MILP)—makes it much easier to work with a wide variety of performance profiles. It also allows us to explicitly model the costs of deliberation both before mission execution (where more deliberation allows an agent to prepare for more contingencies but delays the start of the mission) and during mission execution (where reconsidering plans can improve performance but can also distract when responding to emergent situations). Our mathematical programming approach is unique among proposed solutions to deliberation scheduling in its ability to work with a richer set of performance profiles, and to simultaneously solve the coupled problems of deciding both when to deliberate given its cost, and which decision procedures to execute during deliberation intervals.

Furthermore, our mathematical programming approach provides a domain-independent framework on which we can easily make simplifying transformations or impose additional constraints. For example, as we will show, a deliberation scheduling problem where the mission quality is the multiplication of individual phase qualities (i.e., a nonlinear ob-

*The primary author of the paper is a student.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

AAMAS'06 May 8–12 2006, Hakodate, Hokkaido, Japan.
Copyright 2006 ACM 1-59593-303-4/06/0005 ...\$5.00.

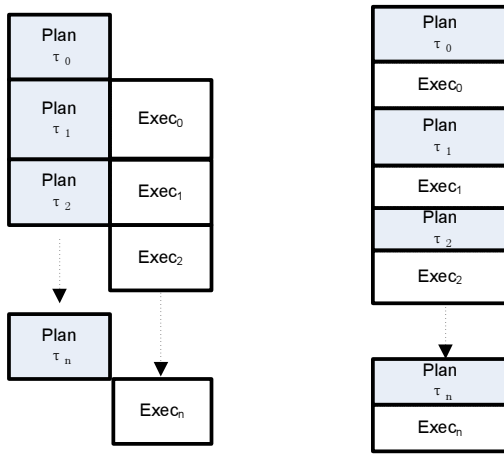


Figure 1: Deterministic phase transitions in two situations: simultaneous planning and execution (left) and interleaved planning and execution (right).

jective function) can be easily transformed into a problem where the mission quality can be represented as the sum of individual phase qualities (i.e., a linear objective function). A mathematical programming formulation also makes it possible to model additional system features. In this paper, we demonstrate this capability by modeling the fact that an autonomous agent should control which phases it will face in the future while scheduling deliberation.

The rest of the paper is organized as follows. Section 2 describes a basic mathematical programming formulation and introduces transformation and approximation techniques. Section 3 extends the formulation to suit domains involving uncertainty, and Section 4 makes further extensions by assuming that phase transitions are controllable by the autonomous agent. We present empirical results in Section 5, where we evaluate the efficiency and optimality of our approach. Finally, Section 6 summarizes the contributions of the work presented and outlines future directions.

2. DETERMINISTIC PHASE TRANSITIONS

We begin our examination of deliberation scheduling techniques by assuming that transitions among phases are deterministic—a mission can be represented as a chain of phases as shown in Figure 1. The left side of the figure depicts the case where deliberation and execution can occur simultaneously, such that deliberation about future phases can be done during execution of the current phase. The right side of the figure shows interleaved planning and execution, where an interval for deliberation is followed by an interval for execution, which in turn is followed by time for deliberation, and so on. In this paper, we describe and illustrate our algorithms on interleaved planning and execution. However, our algorithms, with slight modification, can also be applied for agents that simultaneously plan and execute.

Let τ_0 be the computational time that the autonomous agent initially has when the problem is presented, and τ_i ($i \geq 1$) be additional computational time that the agent can have after finishing the previous phase and before beginning the execution of *phase_i*. Let us temporarily assume that τ_i is specified *a priori*; we will relax this assumption shortly.

The deliberation scheduling problem is then to schedule decision procedures within these available computational time intervals so that the expected quality of mission solution is maximized. A straightforward strategy is to allocate τ_i to *decision_i* (where *decision_i* represents the decision procedure for *phase_i*), but this myopic approach is usually suboptimal since it might be fruitful to use some of the time to get a head start on decision procedures for future phases.

In deliberation scheduling, a common construct is the expected value of computation (EVC) function (also called performance profile) [1, 2, 5, 6, 7]. An EVC (performance profile) of an anytime algorithm is the expected quality of the solution as a function of the algorithm's run time. In this paper, we use this construct as well.

Let t_i denote the total amount of computational time scheduled for decision procedure *decision_i*, and let $V_i(t)$ denote the EVC function of *decision_i*. The deliberation scheduling problem with deterministic phase transitions can then be represented in the following mathematical formulation:

$$\max f(v_0, v_1, \dots, v_n) \quad \left| \begin{array}{l} \sum_{i=0}^k t_i \leq \sum_{i=0}^k \tau_i \\ v_i = V_i(t_i) \\ t_i \geq 0 \end{array} \right. \quad (1)$$

where $\forall k \in \{0, 1, \dots, n\} : \sum_{i=0}^k t_i \leq \sum_{i=0}^k \tau_i$ indicates the fact that the amount of scheduled computational time can never exceed the amount of available computational time. v_i is the expected quality of the solution to *phase_i*, and the objective function $f(v_0, v_1, \dots, v_n)$ specifies how the expected qualities of individual phase solutions determine the overall quality of the mission.

In this paper, we concentrate on the linear objective function $f(v_0, v_1, \dots, v_n) = \sum_i v_i$, which fits many real-world domains where the quality of a mission is the accumulative quality through all its phases. An intuitive example is that of an autonomous delivery robot making several rounds of deliveries: its total quality is the sum of the qualities of individual package deliveries.

Notice that many nonlinear objective functions can be linearized (but not necessarily as $\sum_i v_i$) through mathematical reformulation, and thus the techniques presented in this paper are also applicable in those domains. We now introduce linearization methods for two typical prototypes of nonlinear objective functions.

- **Objective function** $f(v_0, v_1, \dots, v_n) = \min v_i$

A popular objective function in real-world domains is that the quality of a mission is the minimum quality of individual phases. This nonlinear objective function $\min v_i$ can be reformulated as a linear objective function v with additional linear constraints $\forall i : v \leq v_i$. That is, Eq.1 becomes

$$\max v \quad \left| \begin{array}{l} \sum_{i=0}^k t_i \leq \sum_{i=0}^k \tau_i \\ v_i = V_i(t_i) \\ v \leq v_i \\ t_i \geq 0 \end{array} \right.$$

- **Objective function** $f(v_0, v_1, \dots, v_n) = \prod_i v_i$

This objective function is usually used for domains where the probability of successfully completing a mission is the multiplication of the probabilities of successfully completing each phase. By using a logarithmic

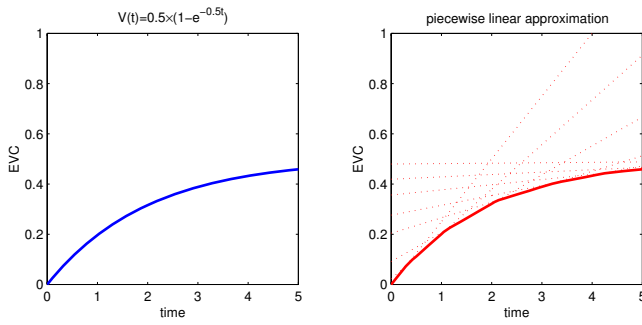


Figure 2: A continuous concave function (left), and its piecewise linear approximation (right).

transformation ($\varepsilon_i = \ln v_i$), Eq.1 can be transformed to:

$$\max e^{\sum_i \varepsilon_i} \quad \left| \begin{array}{l} \sum_{i=0}^k t_i \leq \sum_{i=0}^k \tau_i \\ \varepsilon_i = V'_i(t_i) \\ t_i \geq 0 \end{array} \right.$$

where $V'_i(t) \equiv \ln V_i(t)$.

Notice that e^x is a monotonically increasing function, maximizing $e^{\sum_i \varepsilon_i}$ is equivalent to maximizing $\sum_i \varepsilon_i$, and so the objective function is linearized (while non-linearity is moved to EVC functions).

We have introduced linearization methods for some nonlinear objective functions, but we have not fully addressed the difficulty in solving Eq.1, in which the constraint $v_i = V_i(t_i)$ might be nonlinear. Nonlinear optimization problems are generally intractable; in the following discussion, we present approximation techniques used for linearization in several settings.

• Linear EVC function

If $V_i(t)$ are linear functions of t for any i , then $v_i = V_i(t_i)$ are trivial linear constraints, and deliberation scheduling problems can be formulated as linear programs which are solvable in polynomial time.

• Continuous concave EVC function

For many anytime algorithms, the rate of refinement of the solution slows down with increasing computational activity, i.e., $V_i(t)$ is a continuous concave function. It is well established that a continuous concave function can be approximated as a piecewise linear concave function [8]. Using a sufficiently large number of pieces, such approximation usually performs well. Figure 2 shows an example of approximating function $V(t) = 0.5 \times (1 - e^{-0.5t})$ with a piecewise linear concave function that is composed of eight linear pieces.

Considering that deliberation scheduling often occurs online, we adopt a simple but fast algorithm to construct piecewise linear functions, which, at each time step, myopically adds a linear piece that will reduce approximation error most. Our empirical results show that this myopic algorithm can, in general, approximate a function within several milliseconds, and thus is well suited for online applications.

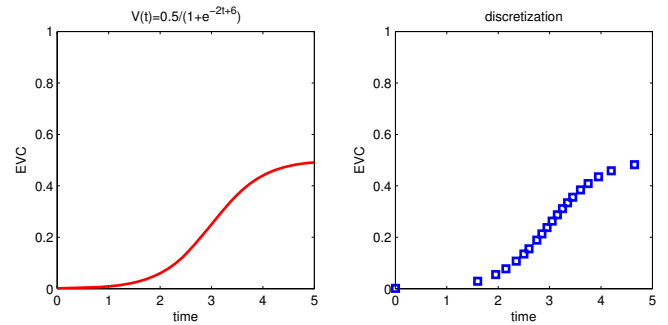


Figure 3: A general nonlinear function (left), and its discrete function (right).

Let $V_{i,j}(t) = a_{i,j}^v \times t + b_{i,j}^v$ be the linear function used to represent the j^{th} segment of the piecewise linear concave curve, then function $V_i(t)$ can be approximated as $V_i(t) = \min_j a_{i,j}^v \times t + b_{i,j}^v$. That is, the constraint $v_i = V_i(t_i)$ in Eq.1 becomes $v_i = \min_j a_{i,j}^v \times t_i + b_{i,j}^v$, and the deliberation scheduling problem can be formulated into the following linear program:

$$\max \sum_i v_i \quad \left| \begin{array}{l} \sum_{i=0}^k t_i \leq \sum_{i=0}^k \tau_i \\ v_i \leq a_{i,j}^v \times t_i + b_{i,j}^v \\ t_i \geq 0 \end{array} \right. \quad (2)$$

A linear program can be solved fast (i.e., in polynomial time), which explains why much previous work focused on continuous concave EVC functions and there exist fast algorithms for such problems [2].

• General nonlinear EVC function

In the previous discussion, we have approximated deliberation scheduling problems with decreasing return rate EVC functions into linear programs. Now, we consider more general nonlinear EVC functions, and use discretization to remove nonlinearity in such functions [8]. Figure 3 shows an example of discretization.

Let $T_{i,j}^v$ and $V_{i,j}$ represent the j^{th} time point and its corresponding value on the discretized function of $V_i(t)$, and let binary variable $\delta_{i,j}^v$ represent whether time point $T_{i,j}^v$ is selected. The problem can then be formulated as:

$$\max \sum_i v_i \quad \left| \begin{array}{l} \sum_{i=0}^k t_i \leq \sum_{i=0}^k \tau_i \\ t_i = \sum_j T_{i,j}^v \times \delta_{i,j}^v \\ v_i = \sum_j V_{i,j} \times \delta_{i,j}^v \\ \sum_j \delta_{i,j}^v = 1 \\ t_i \geq 0 \\ \delta_{i,j}^v \in \{0, 1\} \end{array} \right. \quad (3)$$

$\sum_j \delta_{i,j}^v = 1$ says that a certain amount of computational time is scheduled for decision procedure $decision_i$. The constraints $t_i = \sum_j T_{i,j}^v \times \delta_{i,j}^v$ and $v_i = \sum_j V_{i,j} \times \delta_{i,j}^v$ model the EVC function $V_i(t)$ through binary variable $\delta_{i,j}^v$.

Recall that for the phases with continuous concave EVC functions, we can approximate them into piecewise linear functions. That is, we only need to discretize non-concave EVC functions, which reduces the

number of binary variables used in mixed integer linear programs and thus improves the computational efficiency.

In the discussion so far, it is assumed that τ_i is known *a priori*, but in many real-world domains, τ_i is associated with a cost function $C_i(\tau_i)$ rather than being pre-specified. An autonomous agent thus also needs to determine the intervals of computational time it should use for improving solutions of its decision procedures.

Using our mathematical programming formulation, it is easy to make this extension. That is,

$$\max \sum_i (v_i - c_i) \quad \begin{cases} \sum_{i=0}^k t_i \leq \sum_{i=0}^k \tau_i \\ v_i = V_i(t_i) \\ c_i = C_i(\tau_i) \\ \tau_i \geq 0 \\ t_i \geq 0 \end{cases} \quad (4)$$

Notice that the deliberation cost functions $C_i(\tau)$ are analogous to the EVC functions $V_i(t)$; we can use the approximation techniques described previously to linearize the constraint $c_i = C_i(\tau_i)$ as well. That is, when $C_i(\tau)$ is a continuous convex function (i.e., increasing cost rate), we can approximate it as a piecewise linear convex function. When $C_i(\tau)$ is a general nonlinear function, we can discretize it and build a mixed integer linear program.

We conclude this section by illustrating our algorithm on a simple (randomly generated) example, and more empirical results will be shown in Section 5. In this example, there are four phases $phase_{i \in \{0,1,2,3\}}$. The EVC functions are continuous concave functions:

$$\begin{aligned} V_0(t) &= 3.1319 \times (1 - e^{-0.8233t}) \\ V_1(t) &= 4.0886 \times (1 - e^{-0.3603t}) \\ V_2(t) &= 0.2965 \times (1 - e^{-0.8393t}) \\ V_3(t) &= 2.3293 \times (1 - e^{-0.3057t}) \end{aligned}$$

The deliberation cost functions are continuous convex functions:

$$\begin{aligned} C_0(\tau) &= 0.2037 \times (\tau - 1)^{1.5115} & \text{when } \tau \geq 1 \\ C_1(\tau) &= 0.4808 \times \tau^{1.1843} & \text{when } \tau \geq 0 \\ C_2(\tau) &= 0.4038 \times (\tau - 3)^{1.8348} & \text{when } \tau \geq 3 \\ C_3(\tau) &= 0.2129 \times (\tau - 1)^{1.9415} & \text{when } \tau \geq 1 \\ \text{and } C_i(\tau) &= 0 & \text{otherwise} \end{aligned}$$

Approximating each $V_i(t)$ and $C_i(\tau)$ as a piecewise linear function with 20 pieces, this deliberation scheduling problem can then be formulated as a linear program. With the LP solver *cplex* (www.ilog.com), the mission's expected quality is 5.40 and solving the LP takes 0.021 seconds.

The deliberation schedule is shown in Figure 4. In detail, the agent spends 4.068 time units on its decision procedures

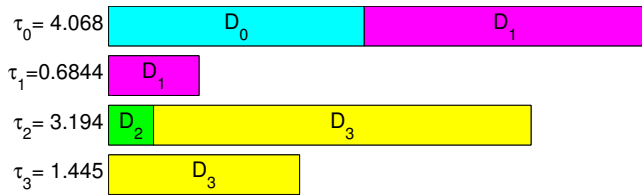


Figure 4: Deliberation schedule where D_i represents the decision procedure for $phase_i$.

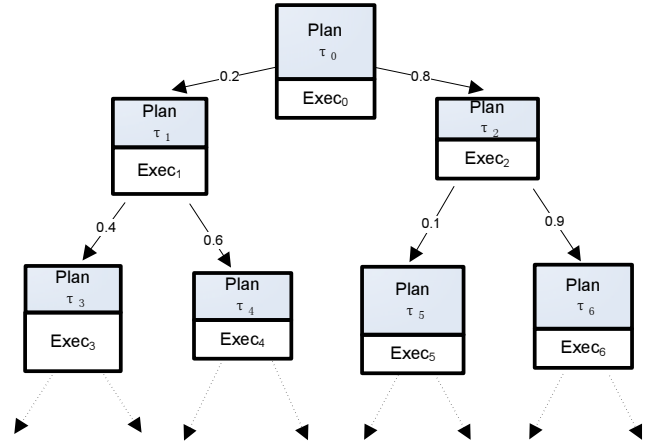


Figure 5: Uncertain phase transitions.

of $phase_0$ and $phase_1$ before it starts to execute the mission. After $phase_0$ is completed, it uses 0.6844 additional time units to improve the solution to $phase_1$. Since $phase_2$ has a much lower expected quality than $phase_3$, most of the available computational time before executing $phase_2$ is used for the decision procedure of $phase_3$. The resulting schedule achieves 20% higher expected quality than using a myopic algorithm that only runs the decision procedure of $phase_i$ right before that phase.

3. UNCERTAIN PHASE TRANSITIONS

We now consider more general deliberation scheduling problems where phase transitions are uncertain. In this paper, we focus on situations where phase transitions can be represented as a tree. Figure 5 shows one such problem. When the agent leaves a phase, it will reach one of the subsequent phases with some probability. The study of uncertain phase transitions is similar to Horvitz's previous work [6], in which future instances are nondeterministic. Our algorithms explore further by explicitly taking into account deliberation costs and more general EVC functions.

Let $p_{i,j}$ represent the transition probability from $phase_i$ to $phase_j$, and then the probability of reaching $phase_j$ in the mission, denoted as P_j , can be computed from $p_{i,j} \times P_i$ where $P_0 = 1$.

Let $\mathcal{A}_i$ denote the set composed of $phase_i$ and its ancestor phases, and let $\mathcal{D}_i$ denote the set composed of $phase_i$ and its descendant phases. Since computational time scheduled for decision procedure $decision_i$ can be from any phase in $\mathcal{A}_i$, $t_i = \sum_{k \in \mathcal{A}_i} \psi_{k,i}$ where $\psi_{k,i}$ is the period of computational time that is located at $phase_k$ and will be used for $phase_i$'s decision procedure. The deliberation scheduling problem with uncertain phase transitions then becomes

$$\max \sum_i P_i \times (v_i - c_i) \quad \begin{cases} t_i = \sum_{k \in \mathcal{A}_i} \psi_{k,i} \\ \tau_k \geq \sum_{i \in \mathcal{D}_k} \psi_{k,i} \\ v_i = V_i(t_i) \\ c_i = C_i(\tau_i) \\ \tau_i \geq 0 \\ t_i \geq 0 \\ \psi_{k,i} \geq 0 \end{cases} \quad (5)$$

where the constraints $\forall k : \tau_k \geq \sum_{i \in \mathcal{D}_k} \psi_{k,i}$ guarantee that

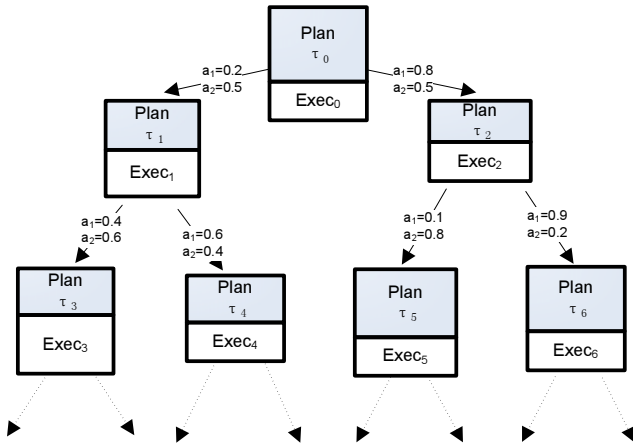


Figure 6: Controllable phase transitions.

the amount of scheduled computational time cannot exceed the amount of available computational time at any point within the mission.

The constraints $t_i = \sum_{k \in \mathcal{A}_i} \psi_{k,i}$ and $\tau_k \geq \sum_{i \in \mathcal{D}_k} \psi_{k,i}$ do not introduce nonlinearities. That is, using the techniques described in Section 2, Eq.5 can be approximated into a linear program whenever, for any $phase_i$, $V_i(t)$ is a continuous concave function and $C_i(\tau)$ is a continuous convex function. Eq.5 can be approximated into a mixed integer linear program whenever some phases have general nonlinear functions $V_i(t)$ or $C_i(\tau)$. Then, it is trivial to derive deliberation schedules from $\psi_{k,i}$ after solving the linear program or mixed integer linear program.

4. CONTROLLABLE PHASE TRANSITIONS

In this section, we further extend our mathematical programming formulation to model another important aspect of deliberation scheduling problems. Recall that in the previous sections it is assumed that transition probabilities (either deterministic or uncertain) among phases are specified *a priori*. However, in many situations, phase transitions might be controllable by agents themselves. For example, an autonomous aircraft needs to determine which route it wants to take when there exist multiple options for reaching a destination. That is, action choices of the agent determine (possibly stochastically) which phases will be reached next. Figure 6 shows one such problem.

With this extension, deliberation scheduling problems become more complex because an agent not only needs to control its reasoning, but also needs to find a policy that maps each phase to its action. A straightforward strategy for solving such problems is to enumerate all possible policies and then, for each policy, use the algorithms presented in the previous sections. However, when the number of policies is large, this simple strategy might become infeasible. In this section, we propose an alternative approach based upon a mathematical programming formulation.

Let $p_{i,a,j}$ represent the probability that the agent reaches $phase_j$ if it executes action a in $phase_i$, let α_i represent the probability that the agent is initially in $phase_i$ (i.e., $\alpha_0 = 1$, $\alpha_{i \geq 1} = 0$), and let $x_{i,a}$ represent the expected number of times that action a is executed at $phase_i$. We can then for-

mulate deliberation scheduling problems with controllable phase transitions into:

$$\begin{aligned} & \max \sum_i x_i \times (v_i - c_i) \\ & \left\{ \begin{array}{l} \sum_a x_{j,a} = \alpha_j + \sum_i \sum_a p_{i,a,j} \times x_{i,a} \\ x_i = \sum_a x_{i,a} \\ t_i = \sum_{k \in \mathcal{A}_i} \psi_{k,i} \\ \tau_k \geq \sum_{i \in \mathcal{D}_k} \psi_{k,i} \\ v_i = V_i(t_i) \\ c_i = C_i(\tau_i) \\ \tau_i \geq 0 \\ t_i \geq 0 \\ \psi_{k,i} \geq 0 \\ x_{i,a} \geq 0 \end{array} \right. \end{aligned} \quad (6)$$

where $x_i = \sum_a x_{i,a}$ is the total expected number of times $phase_i$ is visited. In this paper, it is assumed that the structure of phase transitions can be represented as a tree (and thus a phase cannot be visited more than once during the mission), and so x_i also represents the probability of visiting $phase_i$.

The constraint $\sum_a x_{j,a} = \alpha_j + \sum_i \sum_a p_{i,a,j} \times x_{i,a}$ indicates that the expected number of times $phase_j$ is visited must equal the initial probability distribution at $phase_j$ plus the expected number of times $phase_j$ is entered via all possible transitions [4, 10].

The objective function $\sum_i x_i \times (v_i - c_i)$ in Eq.6, which represents the total expected quality, is a quadratic function (since x_i , v_i and c_i are all variables). In general, quadratic optimization problems are difficult to solve. In the rest of this section, we will show how to reformulate Eq.6 into a mixed integer linear program with the discretization approximation technique.

EVC function-related parameters $T_{i,j}^v$, $V_{i,j}$ and $\delta_{i,j}^v$ were defined in Section 2. We now define deliberation cost function-related parameters $T_{i,j}^c$, $C_{i,j}$ and $\delta_{i,j}^c$ similarly, i.e., $T_{i,j}^c$ and $C_{i,j}$ represent the j^{th} time point and its corresponding value on the discretized function of $C_i(\tau)$ respectively, and binary variable $\delta_{i,j}^c$ represents whether time point $T_{i,j}^c$ is selected, and so Eq.6 can be approximated as the following mixed integer linear program.

$$\begin{aligned} & \max \sum_i \sum_j (\chi_{i,j}^v \times V_{i,j} - \chi_{i,j}^c \times C_{i,j}) \\ & \left\{ \begin{array}{l} \sum_a x_{j,a} = \alpha_j + \sum_i \sum_a p_{i,a,j} \times x_{i,a} \\ x_i = \sum_a x_{i,a} \\ t_i = \sum_{k \in \mathcal{A}_i} \psi_{k,i} \\ \tau_k \geq \sum_{i \in \mathcal{D}_k} \psi_{k,i} \\ t_i = \sum_j T_{i,j}^v \times \delta_{i,j}^v \\ \sum_j \delta_{i,j}^v = 1 \\ \sum_j \chi_{i,j}^v = x_i \\ \chi_{i,j}^v \leq \delta_{i,j}^v \\ \tau_i = \sum_j T_{i,j}^c \times \delta_{i,j}^c \\ \sum_j \delta_{i,j}^c = 1 \\ \sum_j \chi_{i,j}^c = x_i \\ \chi_{i,j}^c \leq \delta_{i,j}^c \\ \psi_{k,i} \geq 0 \\ \tau_i \geq 0, t_i \geq 0 \\ x_{i,a} \geq 0 \\ \chi_{i,j}^v \geq 0, \chi_{i,j}^c \geq 0 \\ \delta_{i,j}^v \in \{0, 1\}, \delta_{i,j}^c \in \{0, 1\} \end{array} \right. \end{aligned} \quad (7)$$

The constraints $\sum_j \delta_{i,j}^v = 1$ (where $\delta_{i,j}^v$ are binary vari-

ables) and $\chi_{i,j}^v \leq \delta_{i,j}^v$ ($\chi_{i,j}^v \geq 0$) indicate that, for each $phase_i$, there exists at most one nonzero variable $\chi_{i,j}^v$, and the constraint $\sum_j \chi_{i,j}^v = x_i$ says that this nonzero variable must equal x_i . All these constraints work together to guarantee $\chi_{i,j}^v = x_i \times \delta_{i,j}^v$. In a similar way, we can have $\chi_{i,j}^c = x_i \times \delta_{i,j}^c$.

Now, we can linearize the quadratic objective function in Eq.6. That is,

$$\begin{aligned} & \sum_i x_i \times (v_i - c_i) \\ \simeq & \sum_i \sum_j (x_i \times V_{i,j} \times \delta_{i,j}^v - x_i \times C_{i,j} \times \delta_{i,j}^c) \\ = & \sum_i \sum_j (\chi_{i,j}^v \times V_{i,j} - \chi_{i,j}^c \times C_{i,j}) \end{aligned}$$

which is the linear objective function (where $V_{i,j}$ and $C_{i,j}$ are constants) used in Eq.7.

$\psi_{k,i}$ in the solution of Eq.7 represents the deliberation schedule, and the policy that maps each phase to its action can be derived from $x_{i,a}$: at $phase_i$, action a is executed with probability $\pi_{i,a} = \frac{x_{i,a}}{\sum_a x_{i,a}}$.

5. EMPIRICAL RESULTS

It is important to remember that the run time of the deliberation scheduling algorithm itself consumes computational time that could otherwise be used for deliberation. In other words, if an agent spends too much time scheduling deliberations, then it might have too little time to actually deliberate. In this section, we give a preliminary empirical evaluation of the efficiency and optimality of our algorithms.

In our simulation, we use continuous concave functions $V_i(t) = M \times (1 - e^{-K \times t})$ to evaluate the LP-based algorithm, and use general nonlinear functions $V_i(t) = \frac{Q}{1 + e^{-J \times (t - D)}}$ to evaluate the MILP-based algorithm. In both cases, continuous convex functions $C_i(\tau) = C \times \tau^N$ are used as our deliberation cost functions. The results shown in this paper are based upon the following parameters: $M \sim [0.5, 5.0]$, $K \sim [0.05, 0.5]$, $Q \sim [0.5, 5.0]$, $J \sim [1.0, 2.0]$, $D \sim [1.0, 3.0]$, $C \sim [0.05, 0.5]$, and $N \sim [1.3, 1.6]$, where $x \sim [L, U]$ represents that x is uniformly distributed in the range $[L, U]$. These parameter distributions are chosen to avoid overly-simple problems, such as the problem where the deliberation costs are so high that it is obvious that none of deliberations should be done. We run our simulation with *cplex* 9.1 LP/MILP solver (www.ilog.com) on a Pentium IV machine. Each data point is the average value from 100 runs, and curves are smoothed to improve readability.

Figure 7 shows the scalability of our algorithms in solving deliberation scheduling problems with deterministic phase transitions, where m in the figure denotes the number of pieces for each function if using piecewise linear approximation, and denotes the number of points for each function if using discretization. The y -axis of the figure specifies the total amount of runtime, including the time for making piecewise linear approximation or discretization, and the time for constructing and solving an LP/MILP. We can see that, although slower than the myopic algorithm that attempts to maximize $V_i(t) - C_i(t)$ at each individual phase without worrying about future phases, our algorithms compute near-optimal solutions reasonably fast, especially when m is small, and their solutions significantly outperform that of the myopic algorithm as shown in Figure 8.

Not surprisingly, using a small m will reduce the approximation accuracy, and thus impair optimality. In Figure 8, we evaluate the optimality of our algorithms with various

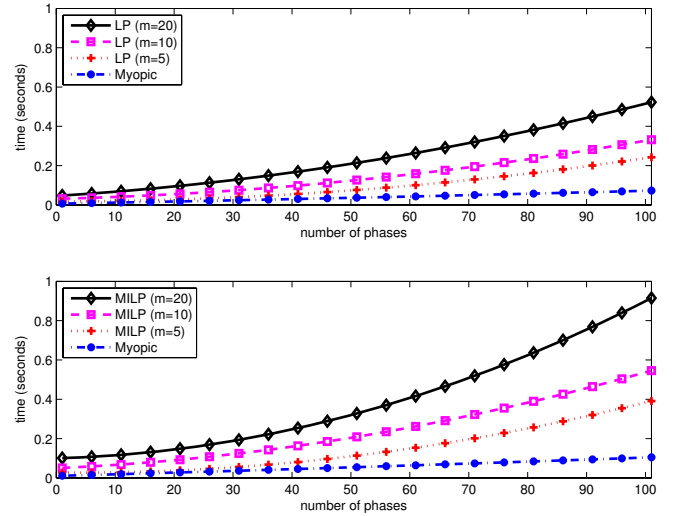


Figure 7: Scalability of the LP-based algorithm (top) and the MILP-based algorithm (bottom) for deterministic phase transitions.

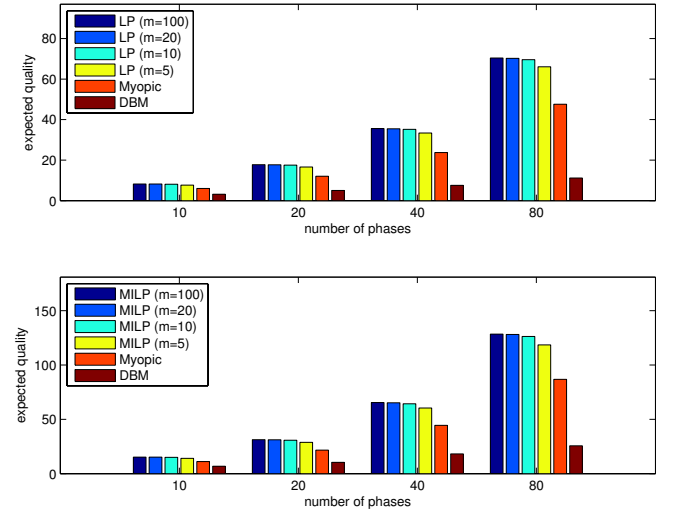


Figure 8: Optimality of the LP-based algorithm (top) and the MILP-based algorithm (bottom) for deterministic phase transitions.

m , the myopic algorithm, and a simple deliberation-before-mission (DBM) algorithm where all deliberations are done before starting the mission (i.e., $\tau_0 \geq 0$ and $\tau_{i \in \{1,2,\dots\}} \equiv 0$). Since, to the best of our knowledge, there are no existing algorithms that are able to compute optimal deliberation schedules in our experimental settings, we use the solution of our algorithms with a large m (i.e., $m = 100$, which usually makes the approximation error less than 1%) as the baseline. These empirical results show that, with $m = 20$, our algorithms are close to optimal. It is also shown that our algorithms, even with a small m (such as $m=5$), outperform the myopic algorithm and the DBM algorithm.

When testing our algorithms in problems with uncertain

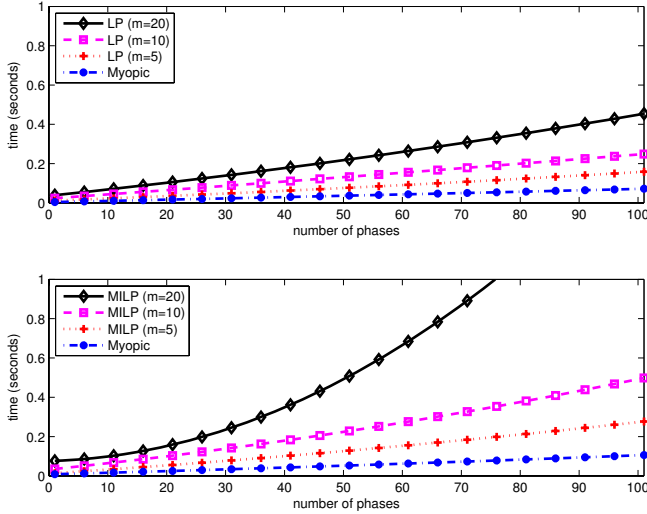


Figure 9: Scalability of the LP-based algorithm (top) and the MILP-based algorithm (bottom) for uncertain phase transitions.

phase transitions, we assume that the phase transitions can be represented as a complete binary tree. When the agent leaves a phase with 2 children, it will reach its left child and its right child with probability ρ and $1 - \rho$, respectively, where ρ is uniformly distributed in the range $[0, 1.0]$. Figure 9 shows the scalability of our algorithms, which is similar to Figure 7.¹

It might be interesting to see that the myopic algorithm also performs pretty well in Figure 10. This is because, for uncertain phase transitions, the time spent on a future phase is less valuable since it is possible that the agent will never reach that future phase. In other words, in such cases, the agent is more intent to act myopically. However, it is important to emphasize that the myopic algorithm does not provide any quality guarantee. In Figure 11, we can see that the myopic algorithm might return unsatisfactory results sometimes (e.g., less than 60% of the optimal quality). In contrast, the performance of our algorithms is much more stable.

Finally, we evaluate our algorithm in solving problems with controllable phase transitions. It is again assumed that the phase transitions can be represented as a complete binary tree. At each phase with two children, there are two possible actions a_1 and a_2 . a_1 moves the agent to the left child with probability 0.9 and to the right child with probability 0.1, and a_2 achieves the opposite effect. Note that problems with controllable phase transitions are in general much more complex than those with deterministic or uncertain phase transitions. When the number of phases is large, the MILP-based algorithm might need a pretty long time to compute an optimal deliberation schedule. However, state-of-the-art MILP solvers (such as cplex) are usually able to

¹It should be noted that, for uncertain phase transitions, if neither EVC functions nor deliberation cost functions can be approximated into piecewise linear constraints, it might take a pretty long time to compute an optimal schedule. However, state-of-the-art MILP solvers are usually able to find a good result within limited time.

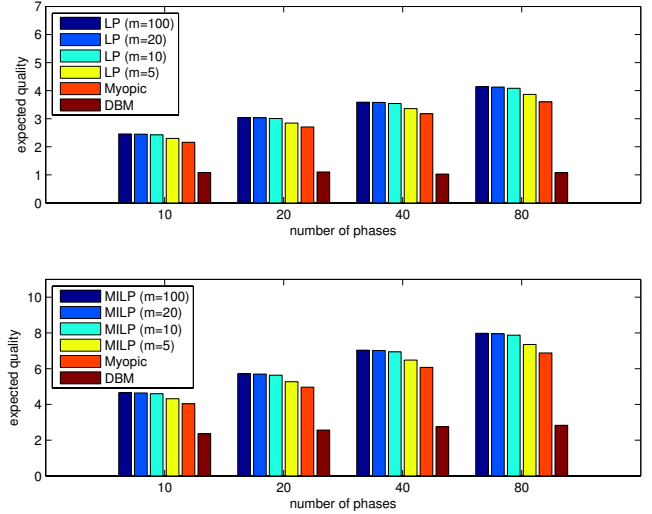


Figure 10: Optimality of the LP-based algorithm (top) and the MILP-based algorithm (bottom) for uncertain phase transitions.

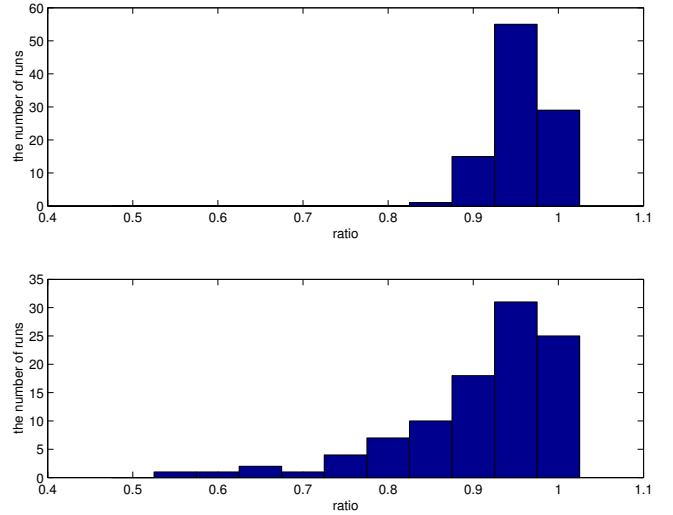


Figure 11: Histograms of ratios of the computed quality to the optimal quality for the LP-based algorithm with $m = 5$ (top) and the myopic algorithm (bottom) in problems with uncertain phase transitions.

return a good solution using much less time. Thus, for on-line applications, we can adopt a two-step algorithm. That is, we first derive a policy by solving Eq.7 with limited time. With the policy, the problem is reduced to an easier one with deterministic/uncertain phase transitions, and then we can solve the reduced problem again and return a deliberation schedule. As shown in Figure 12, this two-step algorithm is usually able to compute a good schedule within a short time, which makes it applicable in computational-time limited domains.

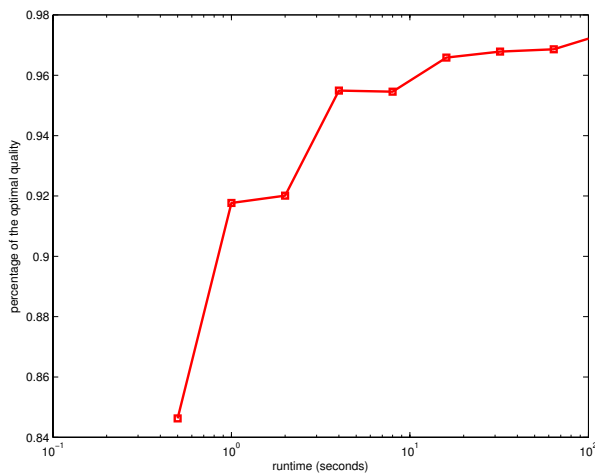


Figure 12: Anytime performance of the two-step algorithm for controllable phase transitions (30 phases and $m = 10$).

6. CONCLUSION AND FUTURE WORK

In this paper, we have presented mathematical programming formulations for deliberation scheduling problems in computational time-limited domains, and have illustrated them through several increasingly complex classes of deliberation scheduling problems. In comparison with previous work, our algorithms are able to solve deliberation scheduling problems in more general settings, and are applicable in complex domains with uncertainty.

The key contribution of this work has been to integrate a variety of established ideas in mathematical programming into a novel approach for solving deliberation scheduling problems. Because most interesting deliberation scheduling problems involve nonlinear constraints, a fundamental technical hurdle that had to be overcome in this work has been to identify transformations that approximate these nonlinear constraints with linear constraints. The deliberation scheduling problems are then solved with existing highly-optimized LP/MILP techniques, rather than building algorithms from scratch that work on the original formulation. By linking deliberation scheduling with LP/MILP, this work lays the foundation for further efforts of exploiting sophisticated techniques in state-of-art LP/MILP solvers [3, 9] for deliberation scheduling problems.

The deliberation scheduling framework studied in this work can be extended to suit more general problems. First, in real-time domains with a large number of phases, a complete mathematical formulation might be too complex to be solvable under time constraints. Considering the fact that probabilities of reaching phases in the far future are often low due to uncertainties, we can restrict the number of phases to be put into the mathematical formulation. As time passes, deliberation scheduling can be repeated if necessary. Second, recall that this work makes a simplifying assumption that the quality of a phase is only determined by the computational time scheduled for it. This is not always true in real-world problems. For example, $phase_i$ might also gain utility when allocating time on another (relevant) $phase_j$ because $phase_i$ and $phase_j$ might have some similar sub-

problems. Thanks to the mathematical formulation, it is often easy to model such relations (e.g., $v_i = V_i(t_i + t_j)$). Finally, with techniques of limiting look-ahead and modeling inter-influence between phases, we might be able to solve deliberation scheduling problems with loops in phase transitions by *unrolling* the graph to make it acyclic (such that a phase can be repeated in the graph). In the future, we will dig deeper in these directions.

This work, like much previous research, assumes that the agent has required knowledge (such as performance profiles) *a priori*. We will look into the problem where an agent needs to acquire such information in practice, and study how the accuracy of the information affects the system performance. Additionally, in recent years, multi-agent deliberation scheduling problems have been of increasing interest, in which an agent can compute ways to improve solutions to its own problems, or joint problems with other agents. Extending our algorithm to multi-agent domains is thus another promising future direction.

7. ACKNOWLEDGEMENTS

This material is based upon work supported in part by the DARPA/IPTO COORDINATORS program and the Air Force Research Laboratory under Contract No. FA8750-05-C-0030. The views and conclusions contained in this document are those of the authors, and should not be interpreted as representing the official policies, either expressed or implied, of the Defense Advanced Research Projects Agency or the U.S. Government.

8. REFERENCES

- [1] M. S. Boddy and T. Dean. Solving time-dependent planning problems. In *IJCAI*, pages 979–984, 1989.
- [2] M. S. Boddy and T. Dean. Deliberation scheduling for problem solving in time-constrained environments. *Artif. Intell.*, 67(2):245–285, 1994.
- [3] W. Cook, W. Cunningham, W. Pulleyblank, and A. Schrijver. *Combinatorial Optimization*. John Wiley & Sons, New York, 1998.
- [4] D. A. Dolgov and E. H. Durfee. Constructing optimal policies for agents with constrained architectures. Technical Report CSE-TR-476-03, University of Michigan, 2003.
- [5] R. P. Goldman, D. J. Musliner, and K. D. Krebsbach. Managing online self-adaptation in real-time environments. In *IWSAS*, pages 6–23, 2001.
- [6] E. Horvitz. Principles and applications of continual computation. *Artif. Intell.*, 126(1-2):159–196, 2001.
- [7] D. J. Musliner, R. P. Goldman, and K. D. Krebsbach. Deliberation scheduling strategies for adaptive mission planning in real-time environments. In M. Anderson and T. Oates, editors, *Metacognition in Computation*, volume SS-05-04 of *AAAI Technical Report*, pages 98–105. AAAI Press, 2005.
- [8] M. J. D. Powell. *Approximation Theory and Methods*. Cambridge University Press, Cambridge UK, 1981.
- [9] L. A. Wolsey. *Integer Programming*. John Wiley & Sons, New York, 1998.
- [10] J. Wu and E. H. Durfee. Automated resource-driven mission phasing techniques for constrained agents. In *AAMAS*, pages 331–338, 2005.