A Risk-Constrained Markov Decision Process Approach to Scheduling Mixed-Criticality Job Sets

Bader Alahmad, Sathish Gopalakrishnan

A Risk-Constrained Markov Decision Process Approach to Scheduling Mixed-Criticality Job Sets

Bader Alahmad
Department of Electrical and Computer Engineering
The University of British Columbia
Vancouver, BC, Canada V6T 1Z4
Email: bader@ece.ubc.ca

Sathish Gopalakrishnan
Department of Electrical and Computer Engineering
The University of British Columbia
Vancouver, BC, Canada V6T 1Z4
Email: sathish@ece.ubc.ca

***Abstract*—We consider the problem of scheduling Mixed Criticality (MC) job systems with an arbitrary number of criticality levels on a single processing platform, when job demands are probabilistic and their distributions are known. We develop a probabilistic framework for MC scheduling, where feasibility is defined as the risk of missing deadlines, which we express in terms of (chance) constraints on the probabilities that jobs of every criticality miss their deadlines. Our goal is to identify and compute "efficiently implementable" scheduling policies under which the given probabilistic constraints are satisfied. We model the problem as a Constrained Markov Decision Process (CMDP), and we show that a feasible Markov randomized scheduling policy exists if the given instance is feasible in a probabilistic sense. A feasible policy can be obtained by solving a linear program. To counter the potential state space explosion, we outline an approximation method that might trade feasibility for efficiency, but which performs well in practice.**

I. Introduction

We consider a problem related to scheduling computational jobs in an embedded system that supports jobs of different criticality levels. *The criticality of a job refers to its semantic importance: how crucial is it that this job complete successfully (correctly and meet timing constraints) for the system to meet its safety requirements and other objectives?* This importance is captured, typically, by safety standards (such as DO-178B, RTCA SC-167 and EUROCAE WG-12) through failure probabilities. For example, the DO-178B avionics system standard uses five criticality levels for tasks with the failure rate (which can be used to derive the failure probability) varying from 10^{-9} per hour to best effort, depending on the criticality level.

From a *job scheduling* perspective, one would like to schedule jobs so that they meet their timing constraints or deadlines. If a job does not meet its deadline—there is a deadline miss—then we deem that event a failure event. In a mixed-criticality system, the scheduling policy should ensure that the tolerable failure rate at each criticality level is respected. For instance, if one were to use the DO-178B standard then jobs at Criticality Level B should only have a failure rate of 10^{-7} per hour of operation.

If one needs to ensure that jobs *always* meet their deadlines then one would need to schedule jobs using the worst-case execution time estimates (WCET) of the jobs. Assuming that all jobs will consume the time indicated by the WCET estimate may lead to system overload but this situation is rarely realized. It is, therefore, possible to satisfy the failure rates at each criticality level by considering the execution time distribution of each job.

We consider the case of n (non-recurrent) *jobs*, each of which is released at time $t = 0$. Each of these jobs has a deadline as well as a criticality that specifies a tolerable failure probability. (In the case of a problem with single-shot jobs, the failure probability represents the number of problem instances when a job may miss its deadline given many problem instances.) Our goal is to obtain a scheduling policy that helps us achieve the required failure probabilities given the execution time distribution for the n jobs.

Contributions

We propose an approach to deriving scheduling policies for mixed-criticality job systems that uses the probability distributions of job execution times. We model MC-job systems as a **chance-Constrained Markov Decision Process (CMDP)** that then allows us to provide guarantees around jobs meeting their timing constraints with

high probability. The chance constraints are sample path constraints on the trajectories of the MDP induced by executing a scheduling policy, and they represent the *risk* of missing deadlines at the various criticality levels. We are then able to formulate an optimization problem whose solution is a scheduling policy that guarantees the failure probabilities at each criticality level. We also briefly discuss an effective approximation scheme to handle the state space explosion, which is based on LP approximations and factored MDP representations.

To the best of our knowledge, there is no work that aims at identifying feasible scheduling policies for MC job systems where job execution times are random. Alahmad et al. [1] were the first to propose the consideration of probabilistic execution times for MC systems. The authors of [2] carried out schedulability analysis of EDF applied to recurrent MC task systems, wherein lower priority tasks are given guarantees against failure. The latter is the closest work we are aware of to our efforts in this article. However, our problem is substantially harder, because it is concerned with *synthesizing* MC scheduling policies, as opposed to *analyzing* existing (fixed) scheduling policies.

Our goal is to improve on the Vestal-Baruah job-dropping model and develop a model and analysis framework that corresponds more closely to the MC standards.

II. System Model

The system we consider is that of n jobs $J_1,\ldots,J_n$ executing upon a single processor, and all jobs are ready to execute at time 0.

Some Notation: We define $\mathbb{N} = \{1,2,\ldots\} \subset \{0,1,2,\ldots\} = \mathbb{Z}_+$, and $\mathbb{R}_+ = [0,\infty)$. For integer $m \in \mathbb{N}$, we let $[m]$ denote the set $\{1,\ldots,m\}$. For a function φ, we use $\varphi(dx) \equiv d\varphi(x)$ to denote a differential (infinitesimal rate of change of φ). If $\mathbb{P}$ is a probability measure on a measurable space $(\Omega,\mathcal{M})$ and φ is a random variable on the same space, we denote φ's distribution as $\mathbb{P}_\varphi \equiv \mathbb{P}\circ\varphi^{-1}$.

In our problem there are $L \geq 2$ criticality levels, and each job J_i has the following parameters associated with it: (1) $\chi_i \in [L]$ is job J_i's **criticality**; (2) $c_i \in \mathbb{N}$ is job J_i's **worst-case execution time (WCET)** estimate; and (3) $d_i > 0$ is job J_i's **deadline**.

The execution time demanded by every job J_i, $i \in [n]$, is described by an integer-valued[1] random variable $Z_i : \Omega \to [c_i]$, defined on a common probability space $(\Omega,\mathcal{M},\mathbb{P})$. Since job demands are integers, we consider scheduling at integer boundaries. Also we assume that job demands are independent, and that we are given the distribution of Z_i, $\mathbb{P}_{Z_i} \equiv \mathbb{P}\circ Z_i^{-1}$, for every $i \in [n]$.

The actual execution time that a job consumes at run-time upon completion is a **job demand realization**; the latter is not known prior to job completion. A job **completes** execution when it announces, or signals, that it has finished execution. This happens when the job has been allocated enough execution time to produce its output entirely. In the probabilistic setting, every $\omega \in \Omega$ is a **scenario** of execution, and if we let $Z = (Z_1,\ldots,Z_n) : \Omega \to \prod_{i=1}^n [c_i]$ be the (random) demand vector, then $Z(\omega)$ is the corresponding **system demand realization**.

A **scheduling policy** is a rule that at every time instant decides which job, from the set of available jobs (those that have not finished execution), is assigned the processor. At every time instant, a scheduling policy may use the characterizing parameters of all the jobs, as well as its previous decisions, in making its next job allocation decision.

III. Problem Definition

We are given L error parameters $p_1,\ldots,p_L$ where, for $\ell \in [L]$, $p_\ell \in [0,1]$ is a lower bound on the probability that all jobs whose criticality is ℓ finish before their deadlines.

We consider an execution model where the scheduler is invoked at every $t \in \mathbb{Z}_+$, and must then, and only then, make a decision as to which job to occupy the processor in the interval $[t,t+1]$. Markov Decision Processes (MDPs) [3] provide a framework for modeling such sequential decision making processes, and they facilitate identifying policies for planning under uncertainty. Our execution model lends itself naturally to the MDP framework, and we shall set up our scheduling problem as one. To accommodate the chance constraints—specified by the given probabilities $p_1,\ldots,p_L$—we impose constraints on the sample paths induced by executing policies. An online policy is favorable if it satisfies the sample path constraints, and our aim is to compute such policy. To incorporate the constraints into the MDP, we shall resort to the framework of *Constrained MDPs* (CMDP) [4]. Whereas unconstrained MDPs can be solved via *dynamic programming* (i.e., Bellman's optimality equations) and greedy decisions extracted using algorithms such as policy and value iteration, CMDPs do not admit such solution methods, and they are best solved via Linear Programming (LP) formulations. We shall take this route.

A. MDP Setup

First, we describe the variables that together represent the system state. Let $\mathsf{Y} = \{0:\mathsf{finished}, 1:\mathsf{not\ finished}\}^n$. For $y_t \in \mathsf{Y}$, $y_t^i = 1$ iff job J_i still requires execution at time

[1] One may equally well work with rational times by regarding time as being divided into integer multiples of some fixed rational quantum $q > 0$, and using scaling arguments to convert to integers.

t, and $y_t^i = 0$ iff J_i has finished execution. That is, y_t^i is the **finish signal** of job i. We shall assume that $y_0 = \mathbf{1}$, the vector of all 1s.

Let A be the set of control actions (jobs) available to the scheduler. We will let $\mathsf{A} = \{e_1, \ldots, e_n\} \cup \{\mathbf{0}\}$, where $\mathbf{0}$ is the vector of all 0s, and $\{e_1, \ldots, e_n\}$ is the standard basis for $\mathbb{R}^n$; e_i is the unit vector that is 1 at the ith coordinate and 0 elsewhere. If the action taken at time $t \in \mathbb{Z}_+$ is $a_t \in \mathsf{A}$, then $a_t = e_i$ means that job J_i occupies the processor during $[t, t+1]$. If $a_t = \mathbf{0}$, then no job is scheduled and the processor is kept idle.

Let $x_t = (x_t^1, \ldots, x_t^n)$ encode the amount of execution time that every job has been allocated up to time t just before acting at time t: $x_0 = \mathbf{0}$, and $x_t = \sum_{m=0}^{t-1} a_m$ for $t \in \mathbb{N}$, where the sum is *vector addition* component-wise. Then for every $t \in \mathbb{Z}_+$ and $i \in [n]$, $x_t^i \in \{0, \ldots, c_i\}$. We will let $\mathsf{X} = \prod_{i=1}^n \{0, \ldots, c_i\}$.

We utilize a variable $r_t \in \{\mathsf{no\ error}, \mathsf{error}, \mathsf{error}'\}^L \equiv \mathsf{R}$ to "mark" the state as "error"; $r_t^\ell = \mathsf{error}$ iff an ℓ-criticality job has missed its deadline at time t. If an ℓ-criticality job misses its deadline at time t, the ℓth error flag will make a transition into an *absorbing* error state error', and it will stay there forever.

We will restrict our attention to **work-conserving** scheduling policies. A non-work-conserving scheduling policy will only delay job completions and may cause jobs to miss their deadlines. Accordingly, the epoch $N = \sum_{i=1}^n c_i$ is an upper bound on the planning, or scheduling, horizon. The **state** of the scheduling system at time $t \in \mathbb{Z}_+$ is $s_t = (t, y_t, x_t, r_t) \in \mathsf{S}$, where $\mathsf{S} \subset \{0, \ldots, N\} \times \mathsf{Y} \times \mathsf{X} \times \mathsf{R}$.

Define the set of **admissible histories** up to time t as $H_0 = \mathsf{S}$, and $H_t = (\mathsf{S} \times \mathsf{A})^t \times \mathsf{S}$ for $t \in \mathbb{N}$. Every element of H_t is called a ***t*-history**, and is of the form

$$h_t = (s_0, a_0, \ldots, s_{t-1}, a_{t-1}, s_t).$$

t-histories are the information available to the scheduler before making its job selection decision at time t.

Let $\mathsf{A}(s_t) \subset \mathsf{A}$ be the set of actions that the scheduler is allowed to apply at time t when the scheduling system is in state s_t. We shall call $\mathsf{A}(s_t)$ the set of **admissible actions** in state s_t. A **scheduling policy** is a sequence $\pi = \{\pi_t : t \in \mathbb{Z}_+\}$, where $\pi_t \equiv \pi_t(da_t|h_t) : 2^{\mathsf{A}(s_t)} \times H_t \to [0,1]$.

The state s_t summarizes all allocation decisions and remaining demands up to time t, and it can be readily verified that $\pi_t(a_t|h_t) = \pi_t(a_t|s_t)$ for every policy π. That is, every policy is *Markov*, and we will be concerned exclusively with Markov scheduling policies ([5] Definition 2.3.2 a).

We implement the requirement that the scheduling policy be work-conserving by specifying that $\mathsf{A}(s_t)$ includes only vectors e_i for which J_i has not finished execution. That is,

$$\mathsf{A}(s_t) = \begin{cases} \{e_i : y_t^i = 1\} & \text{if } \exists i \in [n] \text{ such that } y_t^i = 1 \\ \{\mathbf{0}\} & \text{otherwise.} \end{cases}$$

Control Model: Scheduling decisions are made at every $t \in \{0, \ldots, N-1\}$ exclusively. If a certain job is chosen to execute at some t, then this job occupies the processor for the duration $[t, t+1]$, without interruption, until the scheduler is invoked again at $t+1$. We call $[t, t+1]$ the tth **control interval**. At any $t > 0$, if job J_i was chosen to occupy the processor during $[t-1, t]$ (i.e., $a_{t-1} = e_i$), then the scheduler knows at time t whether or not job J_i requires more execution by observing the value of y_t^i, which will be set to finished if job J_i signals that it has finished execution at time t. The information available to the scheduler at the beginning of the tth control interval is a_{t-1}, y_t, x_t, and r_t. Let $\mathcal{I}_\ell = \{i \in [n] : \chi_i = \ell\}$, $\ell \in [L]$. In summary,

1. At $t = 0$, all jobs are ready to execute and they all demand execution, and the scheduler needs to pick a job to schedule for exactly one time unit before it is invoked again at $t = 1$ (i.e., a_0 needs to be set). Then $y_0 = \mathbf{1}$, $x_0 = \mathbf{0}$, and $r_0^\ell = \mathsf{no\ error}\ \forall \ell \in [L]$;

2. At the beginning of the tth control interval:

2.1 Update the cumulative system allocation by setting $x_t \leftarrow x_{t-1} + a_{t-1}$;

2.2 Observe (acquire) y_t;

2.3 Set Error: For every $\ell \in [L]$

i) If $r_{t-1}^\ell = \mathsf{no\ error}$ and there is $i \in \mathcal{I}_\ell$ such that both $t \geq d_i$ and $y_t^i = 1$, set $r_t^\ell \leftarrow \mathsf{error}$;

ii) If $r_{t-1}^\ell = \mathsf{error}$, set $r_t^\ell \leftarrow \mathsf{error}'$;

2.4 Act: Set a_t to one of the vectors in $\mathsf{A}(s_t)$.

A **transition** $(s, a, \hat{s})$ is **valid** iff it can be generated by the control model.

The Transition Probabilities

We describe the evolution of the system state by a transition kernel (transition matrix) $Q(d\hat{s}|s,a) : 2^{\mathsf{S}} \times (\mathsf{S} \times \mathsf{A}) \to [0,1]$. We shall abuse notation and write $Q(\hat{s}|s,a)$ for $Q(\{\hat{s}\}|s,a)$.

Let $(s, a, \hat{s})$ be a valid transition. If $s = (t, y, x, r)$, then $\hat{t} = t + 1$, and we shall use the more time-suggestive notation $\hat{s} \equiv s_{t+1}$, where $\hat{y} \equiv y_{t+1}$, $\hat{x} \equiv x_{t+1}$ and $\hat{r} \equiv r_{t+1}$; we similarly denote s as s_t. If $(s, a, \hat{s})$ is not valid, then $Q(\hat{s}|s,a) = 0$. Fix an action $a_t = e_i$. Then for transition (s_t, e_i, s_{t+1}) to be valid, we must have $y_t^i = 1$ (not finished) and $x_{t+1}^i = x_t^i + 1$. Also, scheduling J_i does not affect the execution time demands of the other jobs, so s_{t+1} must satisfy $y_t^j = y_{t+1}^j$ for every $j \neq i$. For our fixed state-action

pair (s_t, e_i), we know at time t that $Z_i > x_t^i$, and for $j \neq i$, all we know is that $Z_j \in \mathcal{Z}_j$ for some $\mathcal{Z}_j \subset [c_j]$ that depends on x_t^j and y_t^j. The following is a complete list of all the possible next states s_{t+1} and the corresponding transition probabilities for the fixed action-state pair $(s_t, a_t = e_i)$, and it is here where we fully utilize the assumption of independent job demands:

- $y_{t+1}^i = 1$ (not finished): This says that the scenario ω is such that $Z_i(\omega) > x_{t+1}^i = x_t^i + 1$; moreover, for every $j \neq i$, our knowledge about Z_j does *not* change at time $t+1$, because neither x_t^j nor y_t^j has changed as a result of scheduling J_i. Then,

$$Q(s_{t+1}|s_t, e_i) = \frac{\mathbb{P}(Z_i > x_t^i + 1)}{\mathbb{P}(Z_i > x_t^i)} \tag{1}$$

  if $x_t^i < c_i - 1$, and $Q(s_{t+1}|s_t, e_i) = 0$ otherwise. This equation follows by independence of job demands.
- $y_{t+1}^i = 0$ (finished): Using reasoning similar to the previous case

$$Q(s_{t+1}|s_t, e_i) = \begin{cases} \frac{\mathbb{P}(Z_i = x_t^i + 1)}{\mathbb{P}(Z_i > x_t^i)} & \text{if } x_t^i < c_i - 1, \\ 0 & \text{otherwise.} \end{cases} \tag{2}$$

B. Problem Statement

Let H_∞ denote the canonical **trajectory space** induced by all work-conserving scheduling policies. Every $h \in H_\infty$ is a realization of a schedule, and is of the form $h = (s_0, a_0, s_1, a_1, \dots)$. For initial state s_0, fixed policy π and transition kernel Q, there is a *unique* probability measure $\mathbb{P}^\pi$—that depends on the initial state s_0—on H_∞ such that

$$\mathbb{P}^\pi\big(S_t \in U \mid h_{t-1}, a_{t-1}\big) = Q\big(U | s_{t-1}, a_{t-1}\big) \qquad (U \subset \mathsf{S}).$$

We denote expectation with respect to $\mathbb{P}^\pi$ as $\mathbb{E}^\pi$. Because every policy is Markov, it follows that the induced state process $\{S_t : t \in \mathbb{Z}_+\}$ is a Markov chain for every policy π.

The **finish time** of job J_i with respect to policy π is the earliest time instant at which J_i signals that it has finished execution. Formally, it is a stopping time $F_i \equiv F_i^\pi : H_\infty \to \mathbb{N}$, where $F_i = \min\big\{t \in \mathbb{N} : Y_t^i = 0 \ (\text{finished})\big\}$.

A **probabilistic MC (pMC)** instance is described by a tuple $I = \big(\{J_1, \dots, J_n\}, (p_1, \dots, p_L)\big)$, where every job J_i is characterized by $(\chi_i, c_i, d_i, \mathbb{P}_{Z_i})$.

Problem: Identify a scheduling policy π such that the following L constraints are satisfied:

$$\mathbb{P}^\pi\big(\textstyle\bigcap_{i \in \mathcal{I}_\ell} \{F_i \leq d_i\}\big) \geq p_\ell \qquad \forall \ell \in [L]. \tag{3}$$

Now that we have formalized the problem in the CMDP framework, we are able to give precise definitions of what it means for an instance with probabilistic information to be feasible in the MC setting.

Definition 1 (Probabilistic MC-feasibility). A pMC instance I is probabilistically MC-feasible (pMC-feasible) if there is a policy π such that the constraints (3) are satisfied.

A scheduling policy under which a pMC instance I is pMC-feasible is said to **correctly pMC-schedule** I.

IV. Solution Approach: Risk-Constrained MDP

A. The Risk Constraints

Here we transform the deadline miss probabilities into "risk" constraints that take the form of expectations of immediate costs [6].

We may write the ℓth constraint in (3) as

$$\mathbb{P}^\pi(F_i > d_i \text{ for some } i \in \mathcal{I}_\ell) \leq 1 - p_\ell. \tag{4}$$

Let $\{R_t : t \in \mathbb{Z}_+\}$ denote the induced error process, where $R_t(h) = r_t$ for $h \in H_\infty$. We may write the LHS of inequality (4) as

$$\mathbb{P}^\pi\big(\exists t \in \mathbb{N} \text{ such that } R_t^\ell = \mathsf{error}\big). \tag{5}$$

The probability (5) defines the **risk** across ℓ-criticality jobs associated with executing policy π [6]. The trajectories

$$\big\{h \in H_\infty^\pi : \exists t \in \mathbb{N} \text{ such that } r_t^\ell = \mathsf{error} \text{ for some } \ell \in [L]\big\}$$

are the **error trajectories** that we want to avoid with high probabilities.

Next we write each risk constraint (5) as an expectation under $\mathbb{E}^\pi$ of immediate costs. We define the per-stage *constraint cost* functions $\overline{\kappa}_\ell$, $\ell \in [L]$, as $\overline{\kappa}_\ell(s,a) \equiv \overline{\kappa}_\ell(s) = \mathbb{1}\{r^\ell = \mathsf{error}\}$, where $\mathbb{1}\{\cdot\}$ is the indicator function that evaluates to 1 if the enclosed condition is true, and to 0 otherwise. We note that if $r_t^\ell = \mathsf{error}$ for some $t \in \mathbb{N}$, then $r_{t'}^\ell = \mathsf{error}'$ for all $t' > t$. Thus, if the MDP is moving along an error trajectory, say $h = (s_0, a_0, s_1, a_1, \dots)$, then the constraint cost sequence $\overline{\kappa}_\ell(s_0), \dots, \overline{\kappa}_\ell(s_N)$ is such that there is $t \in \{0, \dots, N\}$ such that $\overline{\kappa}_\ell(s_0) = 0, \dots, \overline{\kappa}_\ell(s_{t-1}) = 0, \underbrace{\overline{\kappa}_\ell(s_t) = 1}, \overline{\kappa}_\ell(s_{t+1}) = 0, \dots, \overline{\kappa}_\ell(s_N) = 0$.

If the trajectory is not error, then $\overline{\kappa}_\ell(s_t) = 0$ for all $t \in \{0, \dots, N\}$. Then for every $\ell \in [L]$ and $h \in H_\infty^\pi$, $\sum_{t=0}^N \overline{\kappa}_\ell(s_t) \in \{0, 1\}$. If we let $\overline{C}_\ell = \sum_{t=0}^N \overline{\kappa}_\ell(S_t) = \sum_{t=0}^N \mathbb{1}\{R_t^\ell = \mathsf{error}\}$, then $\overline{C}_\ell$ is a Bernoulli random variable on H_∞^π with probability of success $\mathbb{P}^\pi\big(\overline{C}_\ell = 1\big)$. Success of the underlying Bernoulli trial happens if there is $t \in \mathbb{N}$ such that $R_t^\ell = \mathsf{error}$. That is, $\{\overline{C}_\ell = 1\} = \big\{\exists t : R_t^\ell = \mathsf{error}\big\}$, from which it follows that $\mathbb{P}^\pi\big(\overline{C}_\ell = 1\big) = \mathbb{P}^\pi\big(\exists t : R_t^\ell = \mathsf{error}\big)$. Since $\overline{C}_\ell$ is Bernoulli, $\mathbb{E}^\pi \overline{C}_\ell = \mathbb{P}^\pi\big(\overline{C}_\ell = 1\big)$, so we may

write the ℓth risk constraint as $\mathbb{E}^\pi \overline{C}_\ell = \mathbb{E}^\pi \sum_{t=0}^N \overline{\kappa}_\ell(S_t) \leq 1 - p_\ell$. Since we are only seeking a feasible policy, all feasible policies for the given instance are equally favorable, and we are free to choose any reasonable objective. Because the system is deadline-driven, we will choose to favor policies that minimize the *sum of expected tardiness* [7]. In our probabilistic framework, the **tardiness** of job J_i is the (policy-dependent) random variable $T_i \equiv T_i^\pi : H_\infty^\pi \to \mathbb{Z}_+$ defined as $T_i = \max(0, F_i - d_i)$. Our objective is to minimize $\sum_{i=1}^n \mathbb{E}^\pi T_i$. Toward this end, we define the immediate objective cost as a function on S such that if job J_i has not finished execution at its deadline, it incurs a unit cost for every time instant past its deadline until it finishes execution. That is, for $s = (t, x, y, r)$ and $i \in [n]$, we define the local objective cost for job J_i as the function

$$\kappa_i(s) = \mathbb{1}\{t \geq d_i \text{ and } y^i = 1 \text{ (not finished)}\}. \qquad (6)$$

For trajectory $h = (s_0, a_0, s_1, \dots)$, if job J_i finishes at some $m > d_i > 1$, then $y_t^i = 1$ (not finished) for all $t \in \{d_i, \dots, m-1\}$, so by (6), the objective cost sequence relating to job J_i is $\kappa_i(s_0) = 0, \dots, \kappa_i(s_{d_i - 1}) = 0, \underbrace{\kappa_i(s_{d_i}) = 1, \dots, \kappa_i(s_{m-1}) = 1}, \kappa_i(s_m) = 0, \dots, \kappa_i(s_N) = 0$, and this sequence sums to $m - d_i$, the tardiness of job J_i with respect to trajectory h. Thus, if we let $\kappa(s) = \sum_{i=1}^n \kappa_i(s)$, then $\sum_{t=0}^N \kappa(s_t)$ is the *total* tardiness for trajectory h.

If we let $\varepsilon_\ell = 1 - p_\ell$, $\ell \in [L]$, then we may write our problem as

$$\textbf{CMDP}: \underset{\pi \in \Pi}{\text{minimize}} \left[\mathbb{E}^\pi \sum_{t=0}^N \kappa(S_t) = \mathbb{E}^\pi \sum_{t=0}^N \sum_{i=1}^n \kappa_i(S_t) \right]$$
$$\text{subject to} \quad \mathbb{E}^\pi \sum_{t=0}^N \overline{\kappa}_\ell(S_t) \leq \varepsilon_\ell, \quad \forall \ell \in [L].$$

B. The Linear Programming (LP) Approach

Recall that the schedule concludes when all jobs finish execution. Moreover, costs (both objective and constraint) can be incurred only until all jobs finish execution. That is, from the costs' perspective, we are concerned with the state process of the MDP while it is in the set

$$\mathsf{S}' = \left\{ s \in \mathsf{S} : y^i = 1 \text{ (not finished) for some } i \in [n] \right\}.$$

Costs keep (possibly) accruing until the state process hits $\mathsf{S} \setminus \mathsf{S}'$. Moreover, the set $\mathcal{M} \equiv \mathsf{S} \setminus \mathsf{S}'$ is always reached under any work-conserving policy in time that is finite and bounded above by the **makespan** (length) of the schedule, which is defined as the (policy-independent) time $\max_i\{F_i\}$ on H_∞^π (or $\sum_{i=1}^n Z_i$ on Ω.) Since we are concerned exclusively with work-conserving policies, it follows that the makespan of any schedule is finite and bounded above by N. Once the state process hits $\mathcal{M}$, it stays there forever; that is; $\mathcal{M}$ is absorbing under any work-conserving policy. In this case, our MDP is called **S'-transient** [8]. In fact, our MDP is in a more restricted class that is a subset of S'-transient MDPs. If we let $T_\mathcal{M}$ be the hitting time of set $\mathcal{M}$; i.e., $T_\mathcal{M} = \inf\{t \in \mathbb{N} : S_t \in \mathcal{M}\}$, then $\mathbb{E}^\pi T_\mathcal{M} \leq \mathbb{E}^\pi \max_{i \in [n]}\{F_i\} \leq N < \infty$ for any work-conserving policy π, and our MDP is said to be S'-**absorbing**, or absorbing to $\mathcal{M}$.

By [4] equations (8.18) and (8.2), we may obtain an optimal policy by solving the following linear program:

$$\textbf{DLP}: \text{minimize} \sum_{s \in \mathsf{S}} \kappa(s) \sum_{a \in \mathsf{A}(s)} \rho(s,a), \qquad \text{subject to}$$
$$\sum_{s \in \mathsf{S}} \overline{\kappa}_\ell(s) \sum_{a \in \mathsf{A}(s)} \rho(s,a) \leq \varepsilon_\ell, \ \ell \in [L]$$
$$\sum_{s' \in \mathsf{S}} \sum_{a \in \mathsf{A}(s')} \rho(s',a)\big(\delta_s(s') - Q(s|s',a)\,\mathbb{1}_{\mathsf{S}'}(s')\big) = \delta_{s_0}(s), s \in \mathsf{S}$$
$$\rho(s,a) \geq 0, s \in \mathsf{S}, a \in \mathsf{A}(s).$$

If we let $M = \sum_{s \in \mathsf{S}} |\mathsf{A}(s)| \leq n|\mathsf{S}|$, then the decision variables in **DLP** are $\big(\rho(s,a) : s \in \mathsf{S}, a \in \mathsf{A}(s)\big) \in \mathbb{R}_+^M$. The functions ρ in **DLP** are nonnegative finite measures on the set $\mathbb{K} = \big\{(s,a) : s \in \mathsf{S}, a \in \mathsf{A}(s)\big\}$. We point out that here $\rho(s,a) = \mathbb{P}^\pi(S_m = s, A_m = a)$, the probability under policy π that state s is occupied and action a is taken at the time instant corresponding to s.

Altman [4] showed in Theorem 8.5 (iii) that **SCMDP** is feasible if and only if **DLP** is feasible. By Theorems 8.2 and 8.5 (iii) of Altman [4], if ρ is a vector in $\mathbb{K}$ that is feasible and optimal for **DLP**, then a feasible (and minimum total expected tardiness) policy is the following: When the state is s, if $\rho\big(s, \mathsf{A}(s)\big) > 0$, then π chooses action $a \in \mathsf{A}(s)$ with probability

$$\frac{\rho(s,a)}{\rho\big(s, \mathsf{A}(s)\big)} = \frac{\rho(s,a)}{\sum_{a' \in \mathsf{A}(s)} \rho(s,a')}, \qquad (7)$$

and otherwise chooses a arbitrarily from $\mathsf{A}(s)$.

According to the previous discussion, we have

Theorem 1. A pMC instance is pMC-feasible if and only if the corresponding linear program **DLP** is feasible. If a pMC instance is pMC-feasible, then the policy given by (7) is feasible and minimizes the sum of expected tardiness.

Computational Complexity: A satisfying assignment for the variables $\big\{\rho(s,a) : s \in \mathsf{S}, a \in \mathsf{A}(s)\big\}$ can be found by solving the linear program **DLP** using any variant of the simplex algorithm, which, in practice, is efficient in the number of decision variables and the number of constraints. **DLP**, however, requires explicit enumeration of the state space S, and therein lies the trouble. The

size of S is exponential in both n and L, so it is not computationally practical to solve **DLP** directly, no matter how efficient the LP solver is. **DLP** can have as many as $n|\mathsf{S}|$ decision variables and $L+|\mathsf{S}|$ constraints. If we let $c = \max_{i\in[n]}\{c_i\}$, then a "very" crude estimate of the size of S is $2^n 3^L (c+1)^n$. Note that the latter bound does not take the time variable into account; although we embedded time in the state to leverage results for computing stationary policies, the fact that all jobs are ready to execute at time 0 and that we are considering only work-conserving policies imply that the state without explicit time, i.e., $s = (x, y, r)$, encodes the time in the schedule already: It is $\sum_{i=1}^{n} x^i$. An MC instance might look like the following: $n = 10$ jobs, $c = 100$, and $L = 2$ criticality levels (dual-criticality systems); for this instance, $|\mathsf{S}|$ might be as large as $2^{10} \times 3^2 \times 101^{10} \approx 10^{23}$, which is astronomical.

V. An Approximation Method

Despite the potentially enormous state space—that would render **DLP** hopelessly intractable for all but the modest pMC instances—there exist approximation techniques that can be utilized to represent the state space compactly and to significantly reduce the number of decision variables and constraints of **DLP**.

We may utilize the *factored MDP representation* to represent the state space and the transition kernel compactly [9]. Instead of enumerating $Q(s_{t+1}|s_t, a_t)$ explicitly for every transition (s_t, a_t, s_{t+1}) (**DLP** requires that Q be given as a $|\mathsf{S}||\mathsf{A}| \times |\mathsf{S}|$ matrix), we identify the *features*—a subset of the variables making up s_{t+1}—on which every component of s_{t+1} depends probabilistically, and we write the transition probabilities for these components only. This representation can then be effectively used to approximate the feasible regions of both **DLP** and its dual program simultaneously [10]. The simultaneous approximation of both the primal and the dual LPs is necessary so that both the number of the decision variables and the number of constraints of **DLP** are reduced, since the latter is at least as big as the former. The LP approximation is based on identifying basis functions defined on the state space or on subsets thereof, which can then be used to represent the LP variables as linear combinations of these basis functions. To wit, with judicious choices of $K_D \ll |\mathsf{S}|$ basis functions on $\mathbb{K}$ to represent ρ and $K_B \ll |\mathsf{S}||\mathsf{A}|$ basis functions on subsets of S to represent the decision variables of the dual of **DLP**, we obtain an approximate LP that has K_D decision variables and $K_B + L$ constraints. The factored MDP representation is then the key tool to efficiently compute the coefficients of the resulting approximate LP.

In a fuller version of the paper, we discuss how to use the factored MDPs to represent Q compactly, and show how to efficiently compute the objective as well as the constraint coefficients of the approximate LP. We point out that a possible heuristic for basis function selection is to solve **DLP** and its dual exactly for the subMDP induced by pairs of jobs *for every pair of jobs*, and use the exact solutions of the two-job subMDPs as basis functions.

VI. Conclusions

We developed a probabilistic framework for reasoning about mixed-criticality jobs systems when job demand distributions are given. We transformed the problem of constructing feasible scheduling policies into a risk-constrained MDP, where risk is the probability, taken over the trajectories induced by the MDP, of missing deadlines at every criticality level. We solved the constrained MDP using a Linear Programming formulation, and we showed how to construct a feasible Markov randomized policy from a solution to the Linear Program. We assumed complete knowledge of job demand distributions. It is natural, however, to consider a variant of our problem where only samples of the demand distributions are available instead of full fledged distributions. Lastly, further study is needed to reason equivalently about recurring tasks under the periodic or sporadic task model.

References

[1] B. Alahmad, S. Gopalakrishnan, L. Santinelli, and L. Cucu-Grosjean, "Probabilities for Mixed-Criticality Problems: Bridging the Uncertainty Gap," in *The Work in Progress session of the 32nd IEEE Real-time Systems Symposium - RTSS 2011*, Wien, Austria, Nov. 2011. [Online]. Available: https://hal.inria.fr/hal-00646586

[2] Z. Guo, L. Santinelli, and K. Yang, "Edf schedulability analysis on mixed-criticality systems with permitted failure probability," in *2015 IEEE 21st International Conference on Embedded and Real-Time Computing Systems and Applications*, Aug 2015, pp. 187–196.

[3] M. L. Puterman, *Markov Decision Processes: Discrete Stochastic Dynamic Programming*, 5th ed. New York: John Wiley and Sons, 2005.

[4] E. Altman, *Constrained Markov Decision Processes*. Chapman and Hall/CRC, 1999.

[5] O. Hernández-Lerma and J. B. Lasserre, *Discrete-Time Markov Control Processes: Basic Optimality Criteria*, ser. Stochastic Modelling and Applied Probability. New York: Springer-Verlag, 1996.

[6] P. Geibel and F. Wysotzki, "Risk-sensitive reinforcement learning applied to control under constraints." *J. Artif. Intell. Res. (JAIR)*, vol. 24, pp. 81–108, 2005.

[7] M. L. Pinedo, *Scheduling: Theory, Algorithms, and Systems*, 3rd ed. Springer, July 24, 2008.

[8] E. Altman, "Constrained markov decision processes with total cost criteria: Lagrangian approach and dual linear program," *Mathematical Methods of Operations Research*, vol. 48, no. 3, pp. 387–417, 1998. [Online]. Available: http://dx.doi.org/10.1007/s001860050035

[9] C. Boutilier, R. Dearden, and M. Goldszmidt, "Stochastic dynamic programming with factored representations," *Artif. Intell.*, vol. 121, no. 1-2, pp. 49–107, Aug. 2000.

[10] D. A. Dolgov and E. H. Durfee, "Symmetric approximate linear programming for factored mdps with application to constrained problems," *Annals of Mathematics and Artificial Intelligence*, pp. 273–293, 2006.