Noname manuscript No.
(will be inserted by the editor)

Computing the Execution Probability of Jobs with Replication in Mixed-Criticality Schedules

Antonin Novak · Zdenek Hanzalek

Received: date / Accepted: date

Abstract Mixed-criticality scheduling addresses the problem of sharing common resources among jobs of different degrees of criticality and uncertain processing times. The processing time of jobs is observed during the online execution of the schedule with the prolongations of critical jobs being compensated by the rejection of less critical ones. One of the central questions in the field of mixed-criticality scheduling is ensuring the high reliability of the system with as few resources as possible.

In this paper, we study the computation of the execution probability of jobs with uncertain processing times in a static mixed-criticality schedule. The aim is to compute the execution probability of jobs (i.e., the objective function of a schedule), which is a problem solvable by a closed-form formula when the jobs are not replicated. We introduce the job replication, i.e., scheduling a single job multiple times, as a new mechanism for increasing the execution probability of jobs. We show that the general problem with job replication becomes $\#\mathcal{P}$-hard, which is proven by the reduction from the counting variant of 3-SAT problem. To compute the execution probability, we propose an algorithm utilizing the framework of Bayesian networks. Furthermore, we show that cases of practical interest admit a polynomial-time algorithm and are efficiently solvable. The proposed methodology demonstrates an interesting connection between schedules with uncertain execution and probabilistic graphical models and opens a new approach to the analysis of mixed-criticality schedules.

Antonin Novak
Czech Institute of Informatics, Robotics, and Cybernetics, Czech Technical University in Prague, CZ and Faculty of Electrical Engineering of Czech Technical University in Prague, CZ
E-mail: antonin.novak@cvut.cz
`https://orcid.org/0000-0003-2203-4554`

Zdenek Hanzalek
Czech Institute of Informatics, Robotics, and Cybernetics, Czech Technical University in Prague, CZ

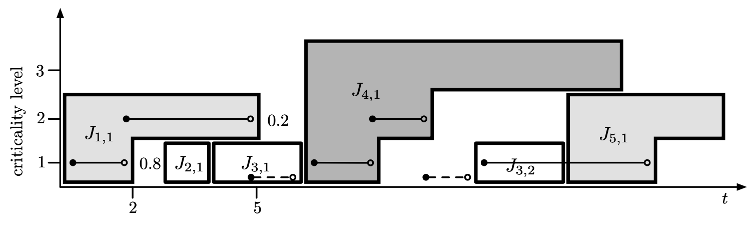
Keywords mixed-criticality · job replication · scheduling · Bayesian networks · computational complexity · uncertain processing time

1 Introduction

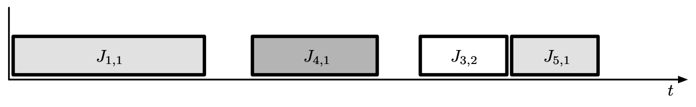
This paper addresses the problem of computing the execution probability of jobs in mixed-criticality schedules. Mixed-criticality systems [41, 8, 9, 29] share resources among jobs with a given degree of importance, i.e., a criticality represented by a positive integer number. Although this paradigm reduces the costs of the system, it introduces new challenges, such as undesired interactions (causing delays or deadline misses) between jobs with uncertain processing times.

The traditional systems typically consider the allocation of critical jobs to a dependable resource (e.g., a dedicated communication line for critical messages) to achieve isolation of critical jobs, and thus, higher reliability of the system; however, the efficiency of such systems becomes an issue. Hence, the key challenge of mixed-criticality systems is to isolate jobs such that low-critical jobs (e.g., $J_{2,1}$, $J_{3,1}$ and $J_{3,2}$ in Figure 1a) do not influence any high-critical job (e.g., $J_{4,1}$ in Figure 1a) without the use of additional resources. Typically, the isolation of critical jobs with uncertain duration is achieved with the two aspects: (i) static scheduling of jobs [38, 5, 29] and (ii) online rejection of low-critical jobs to compensate for processing delays of more critical ones [8, 37, 2]. Although the static scheduling increases the predictability of the system (i.e., its behavior is given by a static schedule which can be analyzed offline), flawless execution of critical jobs may require occasionally to reject less critical jobs during online execution (e.g., $J_{2,1}$ and $J_{3,1}$ in Figure 1b). Thus, careful scheduling of jobs should be used [5, 37] to mitigate the degradation of the execution probability of non-critical jobs without affecting the requirements of critical jobs. To optimize the execution probability of jobs in a schedule, a scheduling algorithm needs to assess the quality (i.e., the objective function) of the current schedule in order to drive the search towards a good solution [20]. Hence, the computation of the objective function of a schedule is the central component of any algorithm that produces high-quality schedules.

In this paper, we introduce the concept of *job replication* to mixed-criticality schedules. The job replication is a mechanism that utilizes unused time slots in a static schedule for additional jobs' execution attempts if the previous attempts have failed. This elegant mechanism increases the execution probability of jobs and does not require additional system resources. In fact, job replication acts as a natural generalization of the mixed-criticality model used, e.g., by Vestal [41] or Seddik *et al.* [37], but it introduces additional complexity to the computation of the objective function of a schedule. Currently known methods (such as [37]) that are used for the computation of the execution probability in mixed-criticality schedules do no longer work when the replication is introduced. Therefore, in this paper, we study the complexity of computing the execution probability of jobs in mixed-criticality schedules with replication (i.e., an objective function of a schedule related to reliability of the system), and we propose an algorithm to compute it that utilizes the theoretical framework of Bayesian networks.



(a) Mixed-criticality schedule with replication with five jobs where J_3 has two replicas.



(b) An execution scenario.

Fig. 1: Mixed-criticality schedule with three criticality levels with one of the possible execution scenarios.

1.1 Contribution and outline

In this paper, we study the job replication, which is a mechanism for increasing the execution probability of jobs in mixed-criticality schedules. Specifically, the main contributions are:

- We introduce the concept of replication to mixed-criticality schedules as a mechanism for increasing the execution probability of jobs.
- We show that the general problem of computing execution probability for a job in a mixed-criticality schedule with replication is $\#\mathcal{P}$-hard, in contrast to the known polynomial-time algorithm for the case without replication. The proof shows that the problem remains hard even if one of the numbers of criticality levels or the maximum number of replicas is equal to a certain constant while the other is bounded by a polynomial in the number of jobs.
- We solve the problem by a reduction to the probabilistic inference in a suitably defined Bayesian network.
- Finally, we show that the cases of reasonable interests can be solved in polynomial time in the number of jobs, which enables practical usage of the job replication.

The rest of the paper is structured as follows. In the subsequent section, we first describe the functionality and an application example of mixed-criticality systems with replicated jobs. Then, we demonstrate the effect of job replication on the execution probability, and we rigorously define the problem statement and survey the related work in this area.

In Section 2, we study the complexity of the general problem. Section 3 deals with the actual algorithm for the computation of execution probability. In Section 3.1, we show the reduction to the probabilistic inference in Bayesian networks, and in Section 3.2, it is shown that a practical case of the problem admits an efficient computation algorithm. Finally, conclusions are drawn in Section 4.

1.2 Mixed-criticality systems with job replication

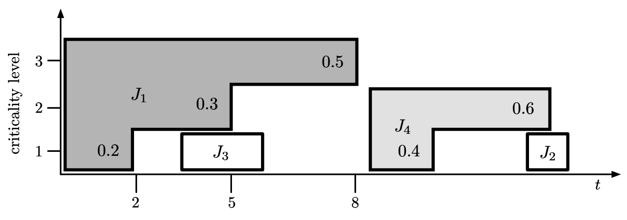
In this section, we describe an application example to illustrate the main concepts of mixed-criticality systems with job replication. Then, we show how the execution probability can be computed in the case without replication and how the job replication increases it.

Application example Consider a message scheduling problem on a shared communication bus in modern cars. Safety-related standards such as ASIL (Automotive Safety Integrity Levels) [6] introduce the existence of messages with several levels of criticality:

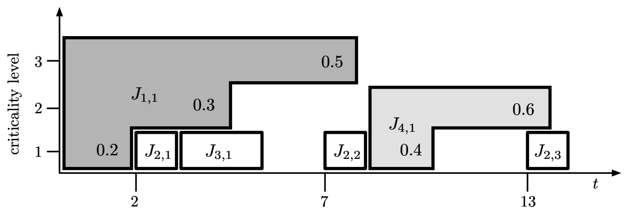
– messages of high criticality (criticality 3) are used for safety-related functionalities (their failure may result in death or severe injury to people), such as steering;
– messages of medium criticality (criticality 2) are used for mission-related functionalities (their failure may prevent activity from being successfully completed), such as parking assist;
– messages of low criticality (criticality 1) are typically used for infotainment functionalities, such as automotive navigation system.

The messages are transmitted via the bus at the moments defined by the static time-triggered schedule [25] which improves determinism and predictability. The goal is to compute objective function reflecting statistical properties of a given static schedule that accounts for disruption of the communication according to the message criticality. In real-life environments, the execution of jobs is affected by various sources of uncertainty, causing, e.g., transmission delays. In the above example, the criticality expresses the commitment to the transmission when the original transmission is prolonged. Therefore, several transmission attempts are awarded to messages with a high criticality, whereas for low-criticality messages, it might be just a single one.

Vestal's [41] widely adopted model of mixed-criticality considers jobs with the criticality given by an integer number and set of different processing times associated with criticality levels. Let us demonstrate the main concepts with



(a) Schedule without replication.



(b) Schedule with three replicas of job J_2.

Fig. 2: Effect of job replication to the execution probability.

an example before providing a formal definition in Section 1.3. Figure 1a shows an example of a mixed-criticality static schedule with six job replicas. Each job has a given integer criticality as it is seen on the vertical axis. For example, job $J_{4,1}$ has criticality of three, job $J_{1,1}$ has criticality of two while $J_{2,1}$ has criticality of one. Notice that $J_{3,1}$ and $J_{3,2}$ are two replicas of the same job, hence, they have the same parameters but different start times. Job $J_{1,1}$ has the considered processing time 2 time units with probability 0.8, and 5 time units with probability 0.2. The considered values of processing times are derived from the (empirical) cumulative distribution function with respect to the selected probability thresholds [41,29].

Mixed-criticality schedules contain several alternative execution scenarios, with the one being selected based upon the realized processing times of jobs that occur during the runtime execution. To compensate for unexpected prolongations of critical jobs observed at the runtime, some of the less critical ones might not be executed under the specific realization of processing times. This can be seen in Figure 1b, where jobs $J_{2,1}$ and $J_{3,1}$ are rejected if realized processing time of $J_{1,1}$ is equal to 5, happening with probability 0.2. However, the second replica $J_{3,2}$ was executed later on. Finally, we note that, e.g., $J_{1,1}$ is never rejected since it does not share its execution time with any other job with higher criticality.

The formal definition of the policy that guides the execution of the schedule will be given in Section 1.3. Next, let us illustrate how the execution probability can be computed and optimized in the case without replication and we show how does the job replication improve it.

Execution probability and job replication The first method for the optimization of the execution probability in mixed-criticality schedules was proposed by [37]. Given a schedule with jobs subject to deadlines, the start times of mixed-criticality jobs are shifted in the schedule, such that low-critical jobs do not share execution time with high-critical ones as long as the deadlines are not violated. We illustrate the shifting of start time with the following example. Consider the schedule in Figure 2a. There, job J_3 has execution probability equal to 0.2 since with probability $0.3 + 0.5 = 0.8$ job J_1 will take longer than 2 time units to process which would reject J_3. If we would shift the start time of J_3 to time 5, then its execution probability would be increased to $0.2 + 0.3 = 0.5$. For more details, we refer the reader to [37].

The disadvantage of this approach, i.e., the shifting the start times, is that the resulting schedule might be too sparse, which results in low utilization of the resource (e.g., CPU in an embedded system). This is an undesired property when the resource is statically scheduled and no jobs are arriving dynamically (since it leads to more empty CPU cycles). Compare the utilization of schedules in Figure 2a (i.e., a lower) and Figure 2b (i.e., a higher).

The main idea in this paper is to utilize unused time slots in the schedule by replicating some of the scheduled jobs which increases their execution probability. First, let us introduce the formal definition of mixed-criticality jobs used thorough the paper. We use a standard mixed-criticality model [41, 37]:

Definition 1 (Mixed-criticality job) *Let $J_i = (\boldsymbol{\pi}_i, \boldsymbol{\mu}_i, \mathcal{X}_i)$ be a mixed-criticality job, where $\mathcal{X}_i \in \mathbb{N}$ is a positive integer criticality, $\boldsymbol{\pi}_i = \left(\pi_i^{(1)}, \ldots, \pi_i^{(\mathcal{X}_i)}\right) \in \mathbb{N}^{\mathcal{X}_i}$ is a vector of processing times of job J_i such that $\pi_i^{(1)} < \ldots < \pi_i^{(\mathcal{X}_i)}$ and $\boldsymbol{\mu}_i = \left(\mu_i^{(1)}, \ldots, \mu_i^{(\mathcal{X}_i)}\right) \in [0,1]^{\mathcal{X}_i}$ is a conditional probability distribution over $\boldsymbol{\pi}_i$ given that J_i is executed.*

See the example in Figure 2a. There, we can see, e.g., job J_1 has $\boldsymbol{\pi}_1 = (2, 5, 8)$, $\boldsymbol{\mu}_1 = (0.2, 0.3, 0.5)$ and $\mathcal{X}_1 = 3$. Let us denote q-th replica of job J_i as $J_{i,q}$, i.e., all replicas of the same job J_i have the same parameters $\mathcal{X}_i$, $\boldsymbol{\pi}_i$, and $\boldsymbol{\mu}_i$, but different start times.

Next, we demonstrate how the replication leads to an increased execution probability over the schedules without replication. For example, consider Figure 2b, where job replica $J_{2,2}$ can be executed at time 7 if $J_{2,1}$ is not executed at time 2. When the execution at time 7 fails as well, $J_{2,3}$ can be still executed at time 13. More generally, the replication of jobs allows us to achieve higher execution probabilities, since the replication may be seen as a form of relaxation, where we relax on the scheduling constraint "*each job is scheduled exactly once*" to "*each job is scheduled at least once*".

In this paper, we deal with the problem of computing execution probabilities of jobs with replication for the given fixed schedule. It is assumed that the schedule is a solution to the scheduling problem denoted in three-field scheduling notation [17] as $1|mc = \mathcal{L}, rep = \mathcal{R}| \sum P_i$. The first field stands for a single resource, $mc = \mathcal{L}$ stands for mixed-criticality jobs with an unspecified number of criticality levels, [21] and $rep = \mathcal{R}$ for the replication considering an unspecified number of replicas per job. The last field $\sum P_i$ indicates that the objective function is the sum of execution probabilities [37]. Note that without loss of generality, we can restrict ourselves to schedules with a single resource only. Indeed, as the schedule is fixed, then each resource can be treated separately.

In the following section, we define the problem statement of computing objective function $\sum P_i$ for the given schedule of problem $1|mc = \mathcal{L}, rep = \mathcal{R}|\sum P_i$, and potentially other constraints. Therefore, we assume that the start times of the jobs are provided, and the goal is to compute the execution probabilities of jobs.

1.3 Problem statement

First, we define an instance of the problem given by a set of mixed-criticality jobs and their schedule.

Definition 2 (Mixed-criticality instance) *Let $I_{\mathcal{MC}} = \{J_1, \dots, J_n\}$ be a set of independent mixed-critical jobs. Let us denote the maximal criticality as $\mathcal{L} = \max_i \mathcal{X}_i$. Let $\mathcal{R} > 1$ be the maximal number of replicas of any job in a feasible schedule for $I_{\mathcal{MC}}$.*

Further in the text, we will refer to the original job J_i in the instance $I_{\mathcal{MC}}$ as a "*job*", whereas their individual occurrences $J_{i,q}$ in the schedule as "*(job) replicas*". Therefore, each job has at least one replica in the schedule, and each replica corresponds to exactly one job. We assume that we are provided with a feasible schedule containing up to $\mathcal{R}$ replicas per job with criticality up to $\mathcal{L}$:

Definition 3 (Schedule with replication) *Let us denote the schedule for a set of jobs $I_{\mathcal{MC}}$ as $\boldsymbol{s} = (s_{1,1}, \dots, s_{1,n_1}, s_{2,1}, \dots, s_{2,n_2}, \dots, s_{n,1}, \dots, s_{n,n_n})$, where $n_k \leq \mathcal{R}$ is the number of replicas of the job J_k in schedule $\boldsymbol{s}$. Let $s_{i,q} \in \mathbb{N}_0$ be the start time of replica $J_{i,q}$. Furthermore, we say that schedule $\boldsymbol{s}$ is feasible if and only if $\forall J_i, J_j \in I_{\mathcal{MC}}, \forall q \leq n_i, r \leq n_j$:*

$$\left(s_{i,q} + \pi_i^{(\min\{\mathcal{X}_i, \mathcal{X}_j\})} \leq s_{j,r}\right) \vee \left(s_{j,r} + \pi_j^{(\min\{\mathcal{X}_i, \mathcal{X}_j\})} \leq s_{i,q}\right),$$

i.e., replicas do not overlap on any criticality level.

Furthermore, we will assume without loss of generality that the start times of the first replicas are ordered $s_{1,1} \leq s_{2,1} \leq \ldots \leq s_{n,1}$ and replicas of each job $J_i \in I_{\mathcal{MC}}$ are ordered as $s_{i,1} \leq s_{i,2} \leq \ldots \leq s_{i,n_i}$.

The above definition of a feasible schedule comes from the strict requirement that the processing of a job cannot be affected by delays of jobs with less or equal criticality. The motivation for it comes from the modular system design principles, where it is desired to achieve so-called *error containment* [30] which is the property of the system that guarantees that the malfunction of

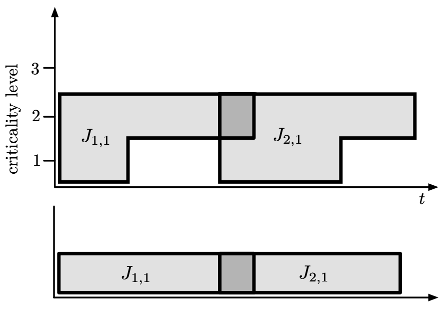


Fig. 3: Example of an infeasible schedule. All realizations are not feasible.

a single component of the system does not arbitrarily propagate through the system. This in turn facilitates *modular certification* [30], which is yet another desirable design principle. To demonstrate the meaning of the definition of the feasibility, consider the example in Figure 3, where we see a schedule with two jobs with two criticality levels and one of the possible execution scenarios of the schedule. If job replica $J_{1,1}$ is executed at the second level, it may correspond to an abnormal state and can be seen as an error. If any two jobs would overlap at the second criticality level (as it is shown in the figure), then this error would disrupt job replica $J_{2,1}$ with equal criticality, which is not desirable, especially if this job implements functionality that has been certified in the isolation. Thus, its certification would not be longer valid when deployed together with the functionality implemented by $J_{1,1}$. Therefore, for the above reasons, it is useful to require that in *all* execution scenarios, a job cannot be affected by delays of less or equally critical jobs.

On the other hand, the definition allows that more critical jobs can affect the execution of less critical jobs. Indeed, each schedule contains so-called *coverage sets* which specify which replicas might be rejected during the execution of the schedule.

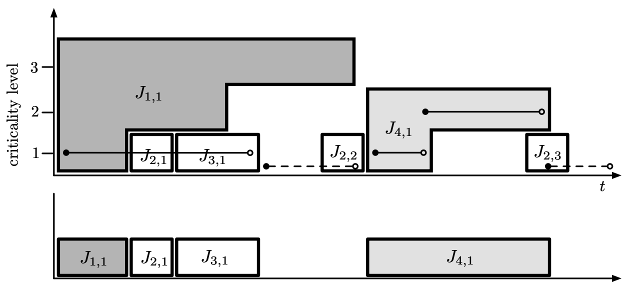
Definition 4 (Coverage set) *We say that replica $J_{i,q}$ is covered by replica $J_{j,r}$ at level ℓ, $\mathcal{X}_j \geq \ell > 1$ in schedule $\boldsymbol{s}$, denoted by $J_{i,q} \in \mathrm{cov}_\ell\left(J_{j,r}\right)$, if and only if*

$$s_{j,r} + \pi_{j,r}^{(\ell-1)} \leq s_{i,q} < s_{j,r} + \pi_{j,r}^{(\ell)}.$$

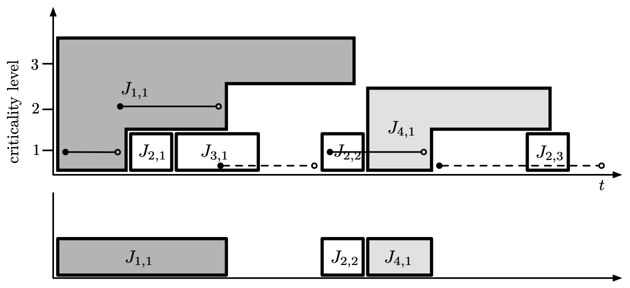
For example, in Figure 4a we have $J_{2,2} \in \mathrm{cov}_3\left(J_{1,1}\right)$. For simplicity, we use the notation $J_{i,q} \in \mathrm{cov}(J_{j,r})$ to denote that there exists a level $\ell \in \{2, \dots, \mathcal{X}_k\}$ such that $J_{i,q} \in \mathrm{cov}_\ell\left(J_{j,r}\right)$. Note that the notation omits the dependence on schedule $\boldsymbol{s}$ as well, as it is clear from the context.

Since the processing times of jobs are uncertain, each job replica can have a different realized processing time, which is revealed during the execution of the schedule.

Definition 5 (Execution level of a replica) *We say that the execution level of replica $J_{i,q}$ is ℓ, denoted as $J_{i,q} \sim \ell$, if and only if its realized processing time*



(a) The first replica of J_2 is executed, hence, the others are rejected.



(b) The second replica of J_2 is executed, hence, the last is rejected.

Fig. 4: Schedule of mixed-criticality jobs with replication under different scenarios.

$\overline{\pi}_{i,q}$ is equal to the processing time of the job J_i at ℓ-th level, i.e., $\overline{\pi}_{i,q} = \pi_i^{(\ell)}$ if and only if $J_{i,q} \sim \ell$.

See the example in Figure 4. In the scenario depicted in Figure 4a, the realized processing time of replica $J_{1,1}$ was $\overline{\pi}_{1,1} = \pi_1^{(1)}$, hence $J_{1,1} \sim 1$. On the other hand, in the scenario in Figure 4b, we have that $J_{1,1} \sim 2$, thus $\overline{\pi}_{1,1} = \pi_1^{(2)}$.

Next, we describe how the mixed-criticality schedules with replicated jobs are executed. To execute replicas in a schedule with respect to their criticalities, the runtime execution of the schedule $\boldsymbol{s}$ is guided by the policy given in Algorithm 1. The design of the execution policy is driven by the implementation aspects of real-life embedded systems. It uses the following rules:

- the replica is started at its start time if no other replica is executed at the moment (line 3: $h(t) = 0$),
- when a replica is started, then it is completed on one of its criticality levels (lines 6–9, $\overline{e} \leftarrow \overline{e} \cup \{J_{i,q} \sim \ell\}$),
- when a replica is completed, then all of the following replicas of the job are rejected (line 3: $\nexists q', \ell : (J_{i,q'} \sim \ell) \in \overline{e}$).

The role of function $h(t)$ is to keep track of the current execution level of the schedule. See an example of the $h(t)$ function in Figure 4 where it is depicted on the y-axis with solid and dashed lines. The value $h(t) = 0$ denotes that the resource is available, and a replica can be executed at its start time. During the replica's execution, its realized processing time is observed, and the execution level of the schedule $h(t)$ is appropriately updated while the execution $J_{i,q} \sim \ell$ of the job replica is stored to the list of observed realizations $\overline{e}$. When the replica is completed, the execution matches up with the base value (i.e., $h(t) = 0$), signaling that the next job replica in the order can be executed.

Algorithm 1 Execution policy of mixed-criticality schedules.

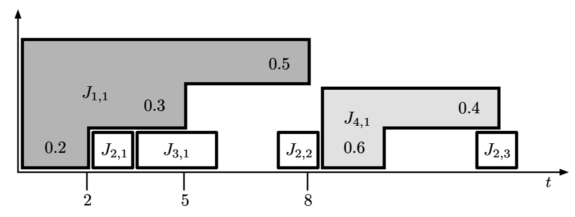
```
    Input: a schedule $\boldsymbol{s}$ and online observations of realized processing times
    Output: the sequence of decisions which replicas to execute and to reject
 1: $t \leftarrow 0$, $h(0) \leftarrow 0$, $\overline{e} \leftarrow \emptyset$
 2: while $t < \max \boldsymbol{s}$ do
 3:     if $h(t) = 0$ and $\exists s_{i,q} \in \boldsymbol{s} : s_{i,q} = t$ and $\nexists q', \ell : (J_{i,q'} \sim \ell) \in \overline{e}$ then
 4:         $\overline{\pi}_{i,q} \leftarrow$ execute $J_{i,q}$ at time $s_{i,q}$   ▷ observe realized processing time
 5:         $\overline{e} \leftarrow \overline{e} \cup \{J_{i,q} \sim \ell\}$   ▷ exec. level of $q$-th replica of job $J_i$ is $\ell$
 6:         for $k \leftarrow 1$ to $\ell$ do   ▷ update exec. level of schedule $h(t)$
 7:             $h(t + t') \leftarrow k \quad \forall t' \in \left[0, \pi_i^{(k)}\right]$
 8:             $t \leftarrow t + \pi_i^{(k)}$
 9:         end for
10:     else
11:         $t \leftarrow t + 1$
12:     end if
13:     $h(t) \leftarrow 0$   ▷ the execution matches-up with the base value
14: end while
```

Note that the execution policy in Algorithm 1 distinguishes only two states of $h(t)$ function, i.e., $h(t) = 0$ and $h(t) > 0$. Indeed, for the decision, which jobs to execute given the observed realizations of processing times, it is sufficient to observe whether the resource is available ($h(t) = 0$) at the time t if t equals to the start time of a job replica. Within the scope of this paper, we use $h(t)$ execution function only inside the execution policy in Algorithm 1 and to define an *execution scenario.* Essentially, any admissible execution of the schedule, i.e., the evolution of $h(t)$ function, forms an execution scenario. We define an execution scenario as a sequence of outcomes (i.e., the list of observations $\overline{e}$) that is admissible under the execution policy in Algorithm 1.

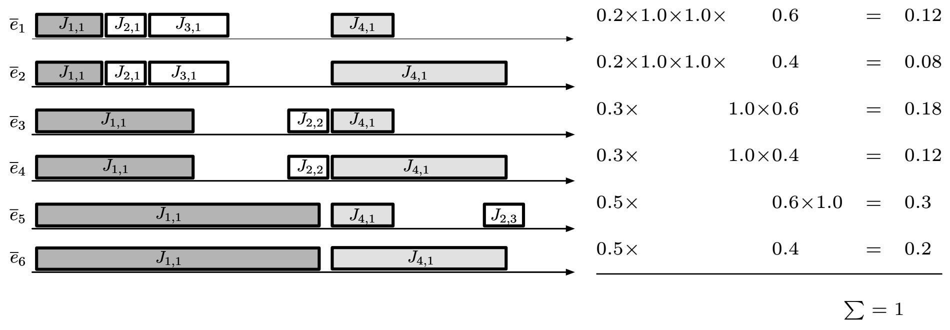
We note that additional functionalities of the system such as recovery policies (e.g., *graceful degradation*) may require fine-grained access to the specific execution level of the schedule $h(t)$ rather than whether the resource is busy at time t or not. These are beyond the scope of this paper but motivates us to introduce $h(t)$ function in its more general form.

Definition 6 (Execution scenario) *The execution scenario $\overline{e}$ is a sequence of replicas' execution levels $J_{i,q} \sim \ell$, i.e., specific realization of processing* *times, which can be realized under the execution policy given by Algorithm 1. We denote the set of all possible execution scenarios in schedule* $\boldsymbol{s}$ *by* $E_{\boldsymbol{s}}$.

For example, the scenario corresponding to Figure 4b is $\bar{e} = (J_{1,1} \sim 2, J_{2,2} \sim 1, J_{4,1} \sim 1)$. Furthermore, notice that the execution policy ensures that at most one replica of each job is executed under any scenario. Consider an example in Figure 4, where the schedule is executed under two different scenarios. There, the solid line displays the current execution level of the schedule $h(t)$ while being dashed when it is 0. In the scenario in Figure 4b, $J_{1,1}$ is executed at its second level, which leads to rejection of replicas $J_{2,1}$ and $J_{3,1}$. Then, the second replica $J_{2,2}$ was executed; thus, the last replica $J_{2,3}$ is rejected. A different scenario is depicted in Figure 4a, where the first replica $J_{2,1}$ is executed; thus, the second (i.e., $J_{2,2}$) and the third (i.e., $J_{2,3}$) replicas are rejected.



(a) Schedule of mixed-criticality jobs with replication.


$\bar{e}_1$ $J_{1,1}$ $J_{2,1}$ $J_{3,1}$ $J_{4,1}$
$\bar{e}_2$ $J_{1,1}$ $J_{2,1}$ $J_{3,1}$ $J_{4,1}$
$\bar{e}_3$ $J_{1,1}$ $J_{2,2}$ $J_{4,1}$
$\bar{e}_4$ $J_{1,1}$ $J_{2,2}$ $J_{4,1}$
$\bar{e}_5$ $J_{1,1}$ $J_{4,1}$ $J_{2,3}$
$\bar{e}_6$ $J_{1,1}$ $J_{4,1}$
0.2×1.0×1.0× 0.6 = 0.12
0.2×1.0×1.0× 0.4 = 0.08
0.3× 1.0×0.6 = 0.18
0.3× 1.0×0.4 = 0.12
0.5× 0.6×1.0 = 0.3
0.5× 0.4 = 0.2
$\sum = 1$

(b) All possible execution scenarios and their probabilities.

Fig. 5: Example schedule and all of its possible execution scenarios.

Finally, we define the execution probability of a job as follows:

Definition 7 (Execution probability of a job) *The execution probability* P_i *of job* J_i *in schedule* $\boldsymbol{s}$ *is given as*

$$P_i = \sum_{\forall \bar{e} \in \{e' \in E_{\boldsymbol{s}} :\, \exists q, \ell : (J_{i,q} \sim \ell) \in e'\}} \; \prod_{\forall (J_{k,r} \sim \ell) \in \bar{e}} \mu_k^{(\ell)}. \qquad (1.1)$$

In other terms, the execution probability of a job is the sum of probabilities of scenarios where a replica of that job was executed. See the example in Figure 5.

There, all six possible scenarios and their probabilities, are depicted. Note that the lowest critical job replicas (i.e., $J_{2,1}$, $J_{2,2}$, $J_{2,3}$, and $J_{3,1}$ in Figure 5a) have the probability 1.0 of being completed at the first level given that they are started (see Definition 1). For example, the execution probability of replica $J_{3,1}$ in this schedule is $P_3 = 0.12 + 0.08 = 0.2$. This can be seen as well as from the fact that $J_{3,1}$ has probability $0.3+0.5 = 0.8$ of being rejected by $J_{1,1}$. Similarly, the execution probability of J_2 is $P_2 = (0.12+0.08)+(0.18+0.12)+(0.3) = 0.8$.

The aim of this paper is to compute the sum of execution probabilities for all jobs in the schedule.

Definition 8 (Problem statement) *The problem is to compute $\sum P_i$ of the given schedule $\boldsymbol{s}$ with replication, where P_i is the execution probability of job $J_i \in I_{\mathcal{MC}}$.*

We assume that the provided schedule is not trivial in the sense that at least one replica is covered; otherwise, every job would be executed with a probability of one. Indeed, it is realistic to assume that the jobs in the provided feasible schedule are constrained by deadlines and other constraints, which likely leds to non-empty coverage sets. Finally, as a remark, let us note that the definition of a mixed-criticality job assumes that $\boldsymbol{\mu}_i$ is a probability distribution, i.e., it sums to one. This assumption might not hold in cases where, e.g., the worst-case execution time cannot be bounded. On the other hand, one can always select finite values of processing times $\boldsymbol{\pi}_i$ such that the selected approximation is arbitrary close to the reality [1]. Therefore, this assumption is not crucial, but it simplifies the presentation of the main ideas.

Remark 1 Note that in this paper, we deal with the computation of the objective function for a given schedule, but we do not touch the question of how to decide which jobs to replicate and which start times to assign them. There are many possibilities to do that, e.g., in a greedy fashion, by a metaheuristic algorithm [42, 16] or by some exact method, but these require a method providing objective value for the current solution. However, the above question goes well beyond the scope of this paper.

One source of difficulty of computing the execution probability from Definition 7 comes from the fact that the set $E_{\boldsymbol{s}}$ of all scenarios in schedule $\boldsymbol{s}$ may contain an exponential number of scenarios, i.e., $|E_{\boldsymbol{s}}| \in \mathcal{O}\left(\mathcal{L}^{n\mathcal{R}}\right)$, where $n = |I_{\mathcal{MC}}|$. Indeed, we will show that the job replication fundamentally changes (as opposed to the case without replication) the computational complexity of the problem and requires a new algorithm for the computation. Even though we will show that the exact computation of the execution probability is hard in general, we propose an efficient method for a tractable class of schedules.

1.4 Related work

The study of mixed-criticality systems originates from the real-time scheduling community due to its practical applications. The most widely adopted mixed-criticality scheduling model was proposed by Vestal in his seminal paper [41].

Table 1: Contributions of this paper.

	replication	complexity result	algorithm
Seddik *et al.* [37]	no	finding start times of jobs with a fixed order to maximize $\sum w_i P_i$ is weakly $\mathcal{NP}$-hard	DP for $mc = 2$, MIP for $mc = \mathcal{L}$
This work	yes	computing $\sum P_i$ is $\#\mathcal{P}$-hard, fixed-parameter tractable with respect to $\max\{\mathcal{L}, \mathcal{R}\}$	inference in Bayesian networks

There, he proposes a model of mixed-criticality jobs that assumes that each job has an integer criticality with each criticality level associated with processing time for that level of assurance. This understanding of mixed-criticality was later adopted by the majority of follow-up works, e.g., Baruah [3], Burns [9], and Davis [14]. This line of research mostly deals with response time analysis of various scheduling policies considering preemptive jobs [24] in so-called event-triggered environment [25]. For a comprehensive review of mixed-criticality scheduling literature, we refer the reader to [8].

A known challenge for complex event-triggered systems is the difficulty of certification for safety-critical applications [2,15]. Therefore, a new stream has emerged and turned the attention toward static scheduling in mixed-criticality systems [29,38,4] that simplifies the certification. A problem with preemptive jobs with two criticality levels was addressed in [5]. The authors proposed a heuristic algorithm that constructs a static schedule for multiple resources while considering precedence constraints. Novak *et al.* [29] dealt with non-preemptive mixed-criticality jobs up to three criticality levels to minimize the length of the schedule, i.e., the makespan. They proposed an approximation algorithm and a branch-and-price decomposition to solve the problem. Seddik [37] noted that makespan minimization with mixed-criticality jobs decreases the probability of the execution. Thus, instead of minimizing the length of the schedule, they proposed a non-regular scheduling criterion that maximizes the execution probability of jobs — spreading them as much as possible under the deadline constraints. They derived a closed-form formula for the computation of the execution probability of jobs, given that the start times are fixed. Furthermore, they gave proof that finding optimal start times with the fixed permutation remains $\mathcal{NP}$-hard, and they proposed (i) a dynamic programming (DP) for the case of two criticality levels and (ii) a mixed-integer linear programming (MIP) model for the general problem. To find an optimal permutation, they developed a branch-and-bound algorithm. However, the used criterion may lead to schedules with low utilization of resources, as they did not consider the job replication.

The problem in this paper is related to stochastic scheduling due to the uncertainty of processing times [22]. There are a plethora of works focusing on processing time uncertainty, often assuming a uniform distribution, normal distribution, an uncertainty set [19,33], or they are distributionally robust [10]. Many of these problems can be formulated as β-robust problems [13] with the goal of, e.g., maximizing the probability that all jobs are completed before the given due date. To the best of our knowledge, the existing approaches in the literature addressing stochastic and robust scheduling cannot be used to solve this problem.

The concept of job replication in scheduling is studied in works concerning parallel algorithms and communication delays [7,31]. The idea is that replication of a job to another processor avoids extra communication and reduces system overhead. Hence, they observe that job replication can decrease the makespan of a schedule when multiple processing units are considered [23]. This is similar to our case, where we allow to consume resource time (that would not be utilized in any way) to improve the objective value by scheduling the same job more than once.

Best to our knowledge, the effect of job replication in static mixed-criticality systems has not been studied before, although it arises as a natural generalization of former mixed-criticality models, such as [41,37]. We summarize the main contributions of this paper with respect to the most similar work [37] highlighted in Table 1.

In the following section, we prove the main complexity result of this paper concerning the computation of the execution probability in a mixed-criticality schedule with replication.

2 Time complexity of the problem

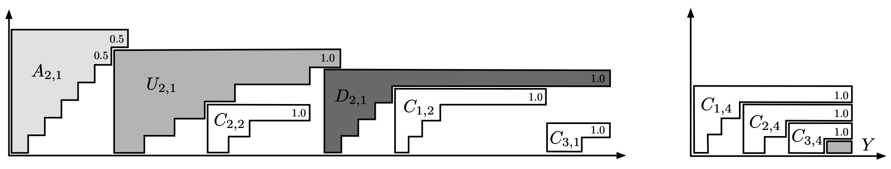
We show that the general problem where either the maximum number of criticality levels $\mathcal{L}$ or the maximum number of replicas per job $\mathcal{R}$ is bounded by a polynomial in the number of jobs and the other is equal to some chosen constant, remains $\#\mathcal{P}$-hard. We remind that $\#\mathcal{P}$ is a class of *counting problems*, i.e., a set of problems that count the number of accepting paths in a polynomial-time non-deterministic Turing machine [39]. An example of a problem contained in $\#\mathcal{P}$ is the following: What is the number of spanning trees in the given connected simple graph? A problem is said to be $\#\mathcal{P}$-hard, if for every problem in $\#\mathcal{P}$, there exists a polynomial-time counting reduction to it [11].

First, we show that to decide whether a job has a non-zero probability of being executed is as hard as determining whether a CNF (*conjunctive normal form*) formula is satisfiable.

Proposition 1 *There exists a finite number of maximum replicas per job $\mathcal{R}$ such that deciding whether $P_i > 0$ for some job J_i is $\mathcal{NP}$-complete.*

Proof First of all, it can be seen that the problem is contained in $\mathcal{NP}$. Indeed, having an execution scenario, we verify in polynomial time in the length of the input (number of all replicas and criticality levels) whether the given job J_i is executed (e.g., similarly as in Algorithm 1).

Next, we will show a polynomial reduction from 3-SAT to our problem. Consider a 3-CNF propositional formula $\mathcal{T} = \bigwedge_{k=1}^{m} c_k = \bigwedge_{k=1}^{m} \left(u_1^k \vee u_2^k \vee u_3^k\right)$ with m clauses and n variables $u_j, j \in \{1,\dots,n\}$, where u_q^k denotes q-th literal of k-th clause. For such a formula, we construct the schedule in the following way. We use five sets of jobs. We have a set $U = \{U_1, \dots, U_n\}$ that denotes jobs corresponding to variables $u_1, \dots, u_n$ and a set $C = \{C_1, \dots, C_m\}$ that corresponds to clauses $c_1, \dots, c_m$. Next, we have jobs denoted as $A = \{A_1, \dots, A_n\}$ that serve as an apriori setting of the execution of jobs U, i.e., they generate execution scenarios corresponding to all possible variable assignments of the formula $\mathcal{T}$. Furthermore, we have a set of jobs $D = \{D_1, \dots, D_n\}$ where D_j is present if the variable u_j acts as a positive literal in some clause. Finally, we have a single job Y, whose execution probability is used to decide whether $\mathcal{T}$ is satisfiable or not. Hence, we have that $I_{\mathcal{MC}} = U \cup C \cup A \cup D \cup \{Y\}$. Job Y takes the role of job J_i, i.e., the job for which we investigate whether it will be executed with a non-zero probability.



(a) Gadget for variable u_2. (b) Gadget for AND gate.

Fig. 6: Basic gadgets used in the reduction utilizing a constant number $\mathcal{R}$ of maximum replicas count.

Now, we will describe the parameters of the jobs. The jobs C_k have criticality $\mathcal{X}_{C_k} = m + 2 - k$, jobs U_j have criticality $\mathcal{X}_{U_j} = m + 3$, A_j have criticality $\mathcal{X}_{A_j} = m + 4$, jobs D_j have criticality $\mathcal{X}_{D_j} = m + 2$ and job Y has criticality $\mathcal{X}_Y = 1$. All jobs from sets U, D, and C have a *Dirac distribution* of probability over processing times — the probability of the execution at their top level is 1, i.e., $\boldsymbol{\mu} = (0, \dots, 0, 1)$. Jobs A_j have probability distribution $\boldsymbol{\mu}_j = (0, \dots, 0, 0.5, 0.5)$. Furthermore, we have that jobs in A, U, D, and job Y have a single replica while jobs in C have exactly four replicas; see Figure 7 where all gadgets are shown in one schedule.

The schedule consists of n copies of the gadget shown in Figure 6a. For each variable u_j we have one gadget containing a replica of all jobs $C_{k,\{1,2,3\}} \in \operatorname{cov}(U_{j,1})$ for clauses c_k where u_j is a negative literal and $C_{k,\{1,2,3\}} \in \operatorname{cov}(D_{j,1})$ for clauses where u_j is a positive literal. The indices $\{1, \dots, 3\}$ of replicas $C_{k,\{1,2,3\}}$ are given according to appearance of variables u_j, $j \in \{1, \dots, n\}$ in clauses c_k, $k \in \{1, \dots, m\}$. The purpose of jobs $A_{j,1}$ is to ensure that for each variable u_j, job replica $U_{j,1}$ will be executed with probability 0.5. If $U_{j,1}$ is rejected, then replicas C_k, where that u_j acts as a negative literal in clause c_k will be executed. If $U_{j,1}$ is executed, then $D_{j,1}$ is rejected; thus, all C_k in its coverage are executed since those correspond to clauses where u_j acts as a positive literal. Please note that it is crucial that jobs both in $\operatorname{cov}(U_{j,1})$ and $\operatorname{cov}(D_{j,1})$ do not overlap in each. The schedule is concluded with the gadget shown in Figure 6b, which implements AND gate, ensuring that job Y

is executed if only if for each $k \in \{1, \dots, m\}$ a replica $C_{k,\{1,2,3\}}$ was executed in the schedule before AND gate.

Consider an example formula $\mathcal{T} = c_1 \wedge c_2 \wedge c_3 = (u_1 \vee u_2 \vee u_3) \wedge (\neg u_1 \vee \neg u_2 \vee u_3) \wedge (u_2 \vee \neg u_3 \vee u_4)$. The resulting schedule derived from the reduction can be seen in Figure 7. The resulting schedule has $\mathcal{O}(n+m)$ replicas with at most $\mathcal{O}(m)$ criticality levels and the maximum replica count per job $\mathcal{R} = 4$.

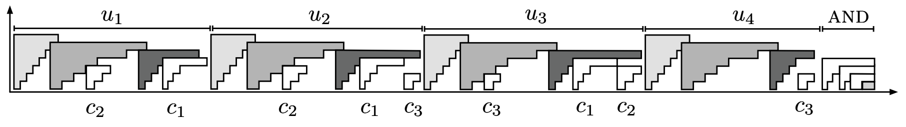


Fig. 7: Example schedule of the reduction from $\mathcal{T} = c_1 \wedge c_2 \wedge c_3 = (u_1 \vee u_2 \vee u_3) \wedge (\neg u_1 \vee \neg u_2 \vee u_3) \wedge (u_2 \vee \neg u_3 \vee u_4)$.

Next, we need to show that in such a schedule $P_Y > 0$ if and only if $\mathcal{T}$ is satisfiable.

SAT $\implies P_Y > 0$. Let ϕ be an assignment of formula $\mathcal{T}$ that is true. We will show that the probability of the execution of job Y is greater than zero, i.e., there is an execution scenario $\overline{e}$ with a non-zero probability that includes Y. Let us define the execution scenario $\overline{e}$ such that $U_{j,1}$ is executed if and only if $\phi(u_j) = \top$. This scenario occurs with probability 2^{-n}, which follows from the choice of the probability distribution of jobs A. We will show that under scenario $\overline{e}$, the job Y is executed.

If ϕ is a true assignment of $\mathcal{T}$, then $\forall c_k \in \mathcal{T}$ there exists a literal u_j that makes clause c_k satisfied. Let u_j be such literal for clause c_k:

- If u_j acts as a positive literal in c_k, then $\phi(u_j) = \top$. Thus, $U_{j,1}$ is executed, therefore $D_{j,1}$ is rejected, since from the schedule construction follows that $C_k \in \text{cov}(D_{j,1})$. Hence, a replica $C_{k,\{1,2,3\}}$ is executed.
- If u_j acts as a negative literal in c_k, then $\phi(u_j) = \bot$. From the definition of scenario $\overline{e}$ follows that $U_{j,1}$ is rejected; thus, a $C_{k,\{1,2,3\}}$ is executed since $C_k \in \text{cov}(U_{j,1})$, which follows from the construction of the schedule.

Since for all jobs C_k, one replica is executed before AND gate, then the last replica of each C_k is rejected; therefore, Y is executed. Hence, job Y is executed in at least one scenario, thus $P_Y > 0$.

$P_Y > 0 \implies$ SAT. If job Y has a non-zero probability of execution, then there is an execution scenario $\overline{e}$ such that for every C_k, one replica is executed before AND gate, otherwise Y would be rejected. We will show, that this scenario defines an assignment of formula $\mathcal{T}$ that is true.

Let us define the assignment ϕ such that $\phi(u_j) = \top$ if and only if $U_{j,1}$ is executed under scenario $\overline{e}$. We will show that ϕ is a model of $\mathcal{T}$. Let $C_{k,\{1,2,3\}}$ be a job replica that is executed in scenario $\overline{e}$ in the schedule gadget corresponding to variable u_j. Then,

– If $C_k \in \mathrm{cov}(D_{j,1})$, then job replica $U_{j,1}$ was executed, therefore $\phi(u_j) = \top$. From the construction of the schedule, u_j acts as the positive literal in c_k; therefore, clause c_k is satisfied.
– If $C_k \in \mathrm{cov}(U_{j,1})$, then $U_{j,1}$ was rejected, therefore $\phi(u_j) = \bot$. From the construction of the schedule it holds that u_j acts as the negative literal in c_k; therefore, clause c_k is satisfied.

Since ϕ is true in all clauses, it is a model of $\mathcal{T}$, and, thus $\mathcal{T}$ is satisfiable. To illustrate how assignments of the formula are mapped to execution scenarios, consider the following two examples. Let us have an assignment ϕ, such that $\phi(u_1) = \top$, $\phi(u_2) = \bot$, $\phi(u_3) = \bot$ and $\phi(u_4) = \top$. The execution scenario corresponding to assignment ϕ is given as $\overline{e} = (A_{1,1} \sim 6, U_{1,1} \sim 6, C_{1,1} \sim 4, A_{2,1} \sim 7, C_{2,2} \sim 3, D_{2,1} \sim 5, A_{3,1} \sim 7, C_{3,2} \sim 2, D_{3,1} \sim 5, A_{4,1} \sim 6, U_{4,1} \sim 6, Y \sim 1)$. An another assignment ϕ' for which $\phi'(u_1) = \top$, $\phi'(u_2) = \top$, $\phi'(u_3) = \bot$ and $\phi'(u_4) = \bot$ translates into the scenario $\overline{e}' = (A_{1,1} \sim 6, U_{1,1} \sim 6, C_{1,1} \sim 4, A_{2,1} \sim 6, U_{2,1} \sim 6, C_{3,1} \sim 2, A_{3,1} \sim 7, D_{3,1} \sim 5, A_{4,1} \sim 7, D_{4,1} \sim 5, C_{2,4} \sim 3)$. □

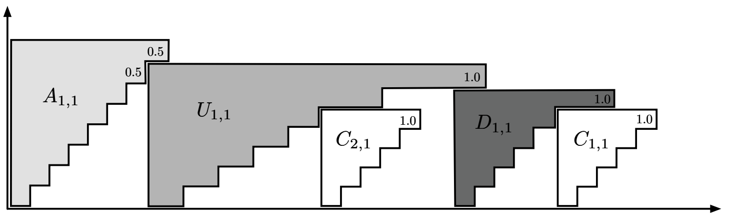
The reduction suggests that the problem remains hard even for a constant number of the maximum job replicas, i.e., $\mathcal{R} = 4$. Moreover, we will show that a non-constant number of criticality levels is not the only source of hardness. Indeed, the problem remains hard, assuming a constant number of criticality levels when the maximum number of replicas is not fixed to a constant.

Proposition 2 *There exists a finite number of criticality levels $\mathcal{L}$ such that determining whether $P_i > 0$ for some job J_i is $\mathcal{NP}$-complete.*

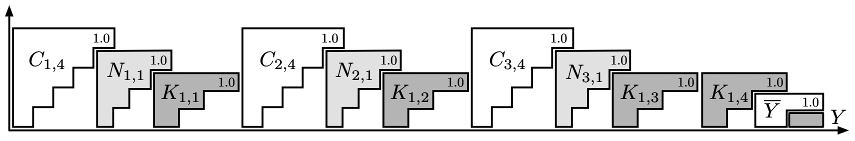
Proof The reduction uses a similar idea as the one described in Proposition 1, where the main difference lies in the construction of AND gate. The resulting schedule again consists of two parts. The first part represents variables $u_1, \ldots, u_n$ and uses $\mathcal{L} = 8$ criticality levels. A replica of job C_k (representing clause c_k) is executed when clause c_k contains a literal u_q^k that makes the clause satisfied. The difference from the structure shown in Figure 6a is that here all C_k's have criticality $\mathcal{X}_{C_k} = 5$, and thus $\mathcal{X}_{D_j} = 6$, $\mathcal{X}_{U_j} = 7$, and $\mathcal{X}_{A_j} = 8$. See the example gadget corresponding to variable u_1 from formula $\mathcal{T}$ in Figure 8a.

The other difference lies in the second part of the schedule, i.e., AND gate, which has to be redesigned to have a constant number of criticality levels. This can be done using $\mathcal{R} \in \mathcal{O}(m)$ maximum replicas per job in a way shown in Figure 8b. The structure of the new AND gate introduces three additional job types: K, N and $\overline{Y}$. Job K has $m + 1$ replicas in the schedule, and the job $\overline{Y}$ a single one. Furthermore, jobs $N = \{N_1, N_2, \ldots, N_m\}$ have a single replica in the schedule for each of them. Jobs in N, K, and $\overline{Y}$ have Dirac distribution of processing times with the probability of one at their top level. The purpose of jobs N_k is to ensure that at least one of the first m replicas of job K is executed if no replica of each C_k, $k \in \{1, \ldots, m\}$ is executed before AND gate. If this happens, then we need to reject Y, which is done by executing the last replica $K_{1,m+1}$. The job replica $\overline{Y}$ ensures that Y is executed if and only if $K_{1,m+1}$ is executed.

Next, we show that job replica Y is executed if and only if a replica for every job C_k was executed in the schedule before AND gate.



(a) Gadget for variable u_1 that uses $\mathcal{L} = 8$ criticality levels.



(b) Gadget for AND gate that uses $\mathcal{L} = 5$ criticality levels.

Fig. 8: Basic gadgets used in the reduction utilizing a constant number of criticality levels $\mathcal{L}$.

- Let us assume that for every scenario $\overline{e} \in E_{\boldsymbol{s}}$ there is a job C_k with no replica executed before AND gate. Therefore, the last replica $C_{k,4}$ is executed in AND gate. Then $N_{k,1}$ is rejected and, hence, the replica $K_{1,k}$ is executed. Therefore, in the last section of AND gate, the last replica $K_{1,m+1}$ is rejected; thus, $\overline{Y}$ is executed, which leads to rejection of Y. Furthermore, if for all scenarios $\overline{e} \in E_{\boldsymbol{s}}$ hold that there is a C_k with no replica executed before AND gate, then $P_Y = 0$.
- Next, let us assume that there is a scenario such that all C_k's are executed before AND gate. Hence, $C_{k,4}$, $k \in \{1, \ldots, m\}$ are rejected, and the corresponding $N_{k,1}$ are executed. Therefore, all replicas of job K are rejected except the last one (i.e., $K_{1,m+1}$). Thus $\overline{Y}$ is rejected, and then, finally, Y is executed; therefore, $P_Y > 0$ follows. □

Remark 2 Notice that jobs in sets U, C, D, N, and K in the reductions above have assumed a Dirac distribution, hence their processing times are, in fact, deterministic, which might feel a bit unnatural. However, it implies several interesting consequences. Namely, if there would be a polynomial-time algorithm that would decide whether a job can be executed, then such an algorithm either has to exploit the fact that distributions of processing times are not Dirac or it would not be sound under $\mathcal{P} \neq \mathcal{NP}$ assumption. Nevertheless, the usage of Dirac distribution for some of the jobs is not the crucial idea in the reduction. One could use a distribution that is *almost* Dirac except for some small positive ϵ. The value of ϵ would be set such that for an unsatisfiable formula the probability of execution of job Y would be non-zero, but arbitrarily small (e.g., $\ll 2^{-\mathcal{O}(\mathrm{poly}(nm))}$) while a satisfiable formula would admit the execution of Y with probability at least $2^{-n} - f(\epsilon)$ where $f(\epsilon)$ is an arbitrarily small non-negative value that incorporates the sum of probabilities of *false-negative* scenarios (i.e., scenarios corresponding to true valuations of the formula but do not execute Y).

Finally, it can be seen that an algorithm computing the exact value of P_Y could be used to solve #3-SAT problem [32] (i.e., the number of satisfiable assignments for the given 3-SAT formula) as it preserves the number of YES certificates in each problem. Indeed, under the considered reduction, each true assignment of the formula $\mathcal{T}$ with n variables increases the execution probability of job Y by 2^{-n} while a false assignment does not increase it and vice versa. Since it is known that #3-SAT is $\#\mathcal{P}$-complete, we have the following corollary:

Corollary 1 *Computing P_i remains $\#\mathcal{P}$-hard when $\mathcal{L}$ or $\mathcal{R}$ is bounded by a polynomial in the number of jobs while the other is equal to a sufficiently large constant.*

Remark 3 Note that it is known that #2-SAT is $\#\mathcal{P}$-complete as well [40], although the decision problem of 2-SAT is polynomially solvable. This suggests that the exact computation of execution probability is hard already for $\mathcal{R} = 3$ maximum replicas, which follows from the reduction introduced in the proof of Proposition 1. On the other hand, for the case $\mathcal{R} = 1$, a polynomial-time solution is known [37]. Hence, the complexity of the case $\mathcal{R} = 2$ stays as an open problem. For the number of criticality levels $\mathcal{L}$ we know that $\mathcal{L} = 8$ already leads to intractability if $\mathcal{R}$ is not a constant.

In the next section, we show that the general problem (with arbitrary $\mathcal{L}$ and $\mathcal{R}$) of computing the execution probability can be reduced to probabilistic inference in Bayesian networks, which gives us an algorithm for the computation. The inference in the resulting network has complexity parametrized by the number of criticality levels $\mathcal{L}$ and by the number of maximum replicas $\mathcal{R}$. Furthermore, we show that the problem becomes tractable, when both $\mathcal{L}$ and $\mathcal{R}$ are bounded by a constant, which is often a realistic case for real-life applications.

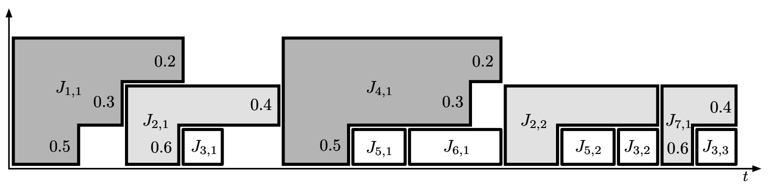
3 Algorithm for computation of the execution probability

In this section, we show how the statistical properties of mixed-criticality schedules with replication can be described with Bayesian networks. A Bayesian network $G = (V, A)$ is a probabilistic directed acyclic graphical model that represents the joint distribution over the set V of random variables using conditional dependencies defined by edges A. Since the processing time of jobs is uncertain, we can view the job replicas as random variables. Then, the execution policy (i.e., Algorithm 1) defines a joint probability distribution $\Pr\{J_{1,1}, \ldots, J_{n,n_i}\}$ over the given schedule that assigns a probability to each execution scenario. However, the representation of the full joint distribution is costly as it has $\mathcal{O}(\mathcal{L}^{n\mathcal{R}})$ parameters. To overcome this, we use Bayesian networks (BN) [34], which can be seen as an efficient way of representing joint distributions.

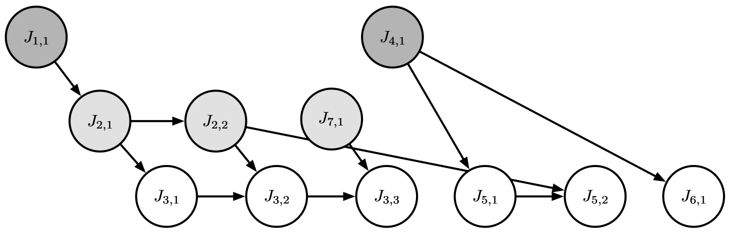
We show that any schedule for scheduling problem $1|mc = \mathcal{L}, rep = \mathcal{R}|\sum P_i$ defines a Bayesian network. We use this network for the computation of the execution probability of jobs using the existing inference algorithms for BNs. Moreover, we utilize the theoretical framework of Bayesian networks to analyze which case of the problem admits an efficient algorithm for probability computation.

3.1 Reduction to Bayesian networks

Under the considered mixed-criticality model, replica $J_{i,q}$ with $\mathcal{X}_i$ criticality levels can be either: (i) executed at one of its levels (i.e., $\mathcal{X}_i$ outcomes), (ii) rejected by some other job replica, or (iii) rejected due to successfully completed previous replica of the same job. The probabilities of its outcomes are given by the start times of other replicas in the schedule. If the replica is executed, then its outcomes have probabilities given by the job specification, i.e., $\boldsymbol{\mu}_i$ distribution. When the replica is rejected, then all these outcomes have a probability of zero. However, the probability of being executed or rejected depends on the preceding jobs replicas in the schedule and their coverage.



(a) Example schedule $\boldsymbol{s}$.



(b) Corresponding Bayesian network $G(\boldsymbol{s})$.

Fig. 9: Representation of a schedule by a Bayesian network.

Network structure The structure of Bayesian network $G(\boldsymbol{s}) = (V_{\boldsymbol{s}}, A_{\boldsymbol{s}})$ corresponding to a schedule $\boldsymbol{s}$ is defined in the following way. We assume that each job replica in the schedule corresponds to a single discrete random variable. Hence, the set of vertices $V_{\boldsymbol{s}}$ is equal to the set of job replicas in schedule $\boldsymbol{s}$. A random variable (i.e., a vertex) corresponding to job replica $J_{i,q}$ has outcome space given as $\{\dagger, \star, 1, \dots, \mathcal{X}_i\}$ with the total ordering defined as $\dagger \prec \star \prec 1 \prec \dots \prec \mathcal{X}_i$. The outcomes $\ell \succeq 1$ coincide with the execution levels of the job. That is, $J_{i,q}$ has the outcome ℓ, i.e., $J_{i,q} \sim \ell$ for $\mathcal{X}_i \succeq \ell \succeq 1$ if and only if its execution level is ℓ. The outcome $\ell = \star$ denotes that the job replica was rejected by a preceding replica of a different job with higher criticality (i.e., it was covered by it). Finally, the outcome $\ell = \dagger$ denotes that a preceding replica of the same job was completed in the schedule before. Note that we use two different outcomes for the rejection of a job replica, i.e., $\star$ and $\dagger$. This distinction is necessary to model the fact that the execution policy of mixed-criticality schedules executes at most one replica of each job (see Algorithm 1). Next, as an example, let us demonstrate how the outcomes of random variables relate to possible execution scenarios in the schedule. Consider the execution scenario $\overline{e}_3$ displayed in Figure 5b. The sequence of outcomes corresponding to this scenario $\overline{e}_3$ is $(2, \star, \star, 1, 1, \dagger)$.

Next, we describe how vertices in the network are connected. Vertex $J_{k,r}$ is connected to $J_{i,q}$ (i.e., $J_{k,r} \to J_{i,q} \in A_{\boldsymbol{s}}$) if and only if

$$J_{i,q} \in \operatorname{cov}(J_{k,r}) \text{ or } (i = k \wedge q = r + 1)\,. \tag{3.1}$$

In other words, for each vertex, its parent is a vertex covering it in the schedule or its immediate preceding replica. The intuition behind connections defined like that is that a job replica needs to be connected to all other job replicas that influence its execution. See an example schedule in Figure 9a and the corresponding BN displayed in Figure 9b. Let us note that we may assume that the network $G(\boldsymbol{s})$ is connected; otherwise, one could deal with each component separately.

Conditional distributions The other parameters of BNs are *conditional probability tables* (CPTs). For each random variable, CPT defines a distribution of outcomes conditioned by the outcomes of all its parents (i.e., the *evidence*). In our case, CPTs are defined as follows. If vertex $J_{i,q}$ has no parent, then its CPT is simply the distribution over its criticality levels $\{1, \dots, \mathcal{X}_i\}$ with the outcomes $\star$ and $\dagger$ having zero probability. If $J_{i,q}$ is the first replica of a job (i.e., $q = 1$), then for the outcomes of its parents that reject $J_{i,q}$, its CPT assigns the probability of 1 for $\star$ outcome. For the outcomes of its parents that permit its execution, the probabilities of outcomes $\{1, \dots, \mathcal{X}_i\}$ are given by $\boldsymbol{\mu}_i$ distribution.

Consider the schedule in Figure 9a with the parameters of distributions for job J_1 and J_4 are $\boldsymbol{\mu}_1 = \boldsymbol{\mu}_4 = (0.5, 0.3, 0.2)$, for job J_2 and J_7 are $\boldsymbol{\mu}_2 = \boldsymbol{\mu}_7 = (0.6, 0.4)$, and for J_3, J_5 and J_6 are $\boldsymbol{\mu}_3 = \boldsymbol{\mu}_5 = \boldsymbol{\mu}_6 = (1.0)$. If $J_{1,1}$ is executed at criticality level 3, then $J_{2,1}$ will be rejected, and, given that $J_{1,1} \sim 3$, the probability of $J_{2,1}$ being rejected is 1. On the other hand, when $J_{1,1} \prec 3$, then the probability of $J_{2,1}$ being rejected is zero. More precisely, the conditional probabilities for replica $J_{i,q}$ are given as

$$\Pr\left\{ J_{i,q} \sim \ell \;\middle|\; \bigwedge_{\substack{(j,r,k): \\ J_{i,q} \in \mathrm{cov}_k(J_{j,r})}} (J_{j,r} \prec k) \wedge (J_{i,q-1} \sim \star) \right\} = \mu_i^{(\ell)},$$

$$\Pr\left\{ J_{i,q} \sim \star \;\middle|\; \left(\bigvee_{\substack{(j,r,k): \\ J_{i,q} \in \mathrm{cov}_k(J_{j,r})}} (J_{j,r} \succeq k) \right) \wedge (J_{i,q-1} \sim \star) \right\} = 1,$$

$$\Pr\left\{ J_{i,q} \sim \dagger \,|\, J_{i,q-1} \succ \star \vee J_{i,q-1} \sim \dagger \right\} = 1,$$

where without loss of generality, we assume that $J_{i,0} \sim \star$.

Example CPTs for some of the replicas for the schedule in Figure 9a can be seen in Table 2. The replicas $J_{1,1}$, $J_{4,1}$, and $J_{7,1}$ do not have any evidences (i.e., the outcomes of all its parents) since their sets of parents are empty. Similarly, the blank evidence, e.g., the first row for $J_{2,2}$ in Table 2f, is used to denote that an arbitrary outcome (i.e., $\dagger, \star, 1, \ldots$) of its parents applies. Note that we have used a compact representation of conditional probabilities instead of full CPTs, which can be in fact exponential in $\mathcal{L}$. Therefore, BNs admit also different representations of conditional distributions (e.g., decision trees) that lead to more concise representation in some cases.

Marginalization To compute the probability of execution of jobs, one needs to perform marginal inference in the constructed network. There are known algorithms that can perform such inference and their implementations are widely available. For the exact inference in BNs, multiple algorithms exist [18], such as *variable elimination* or *junctions trees*. One can also trade the precision for the computational time and use approximate inference methods [28], such as *Markov Chain Monte Carlo* or *Gibbs sampling*. To compute the execution probability of replica $J_{i,q}$, we utilize the property of Bayesian networks [27,34] that states

$$\Pr\{J_{i,q}\} = \sum_{\forall s_{j,r} \in \boldsymbol{s} \setminus \{s_{i,q}\}} \prod_{i'=1}^{n} \prod_{q'=1}^{n_{i'}} \Pr\{J_{i',q'} \,|\, parents(J_{i',q'})\}, \tag{3.2}$$

where $parents(J_{i',q'})$ denotes the set of all immediate predecessors of $J_{i',q'}$ in BN $G(\boldsymbol{s})$. The equation (3.2) suggests that the probability distribution $\Pr\{J_{i,q}\}$ can be computed with the marginalization over all variables except $J_{i,q}$ of the joint distribution $\Pr\{J_{1,1}, \ldots, J_{n,n_n}\}$. The joint distribution is factorized using the conditional independence relations defined by the network. The complexity of the computation is hidden in the marginalization step where we need to perform the summation over all combinations of replicas' outcomes and multiply their probabilities altogether. However, one can notice that vertices that are not ancestors of $J_{i,q}$ in $G(\boldsymbol{s})$ do not influence the distribution

Table 2: CPTs for the example schedule $\boldsymbol{s}$.

outcome					evidence
$\dagger$	$\star$	1	2	3	
0.0	0.0	0.5	0.3	0.2	$\emptyset$

(a) Replicas $J_{1,1}$ and $J_{4,1}$.

outcome				evidence
$\dagger$	$\star$	1	2	
0.0	0.0	0.6	0.4	$\emptyset$

(b) Replica $J_{7,1}$.

outcome				evidence
$\dagger$	$\star$	1	2	$J_{1,1}$
0.0	0.0	0.6	0.4	$\dagger \lor \star \lor 1 \lor 2$
0.0	1.0	0.0	0.0	3

(c) Replica $J_{2,1}$.

outcome				evidence
$\dagger$	$\star$	1	2	$J_{2,1}$
1.0	0.0	0.0	0.0	$\dagger \lor 1 \lor 2$
0.0	0.0	0.6	0.4	$\star$

(d) Replica $J_{2,2}$.

outcome			evidence
$\dagger$	$\star$	1	$J_{2,1}$
0.0	0.0	1.0	$\dagger \lor \star \lor 1$
0.0	1.0	0.0	2

(e) Replica $J_{3,1}$.

outcome			evidence	
$\dagger$	$\star$	1	$J_{3,1}$	$J_{2,2}$
1.0	0.0	0.0	$\dagger \lor 1$	
0.0	0.0	1.0	$\star$	$\dagger \lor \star \lor 1$
0.0	1.0	0.0	$\star$	2

(f) Replica $J_{3,2}$.

outcome			evidence	
$\dagger$	$\star$	1	$J_{3,2}$	$J_{7,1}$
1.0	0.0	0.0	$\dagger \lor 1$	
0.0	0.0	1.0	$\star$	$\star \lor 1$
0.0	1.0	0.0	$\star$	2

(g) Replica $J_{3,3}$.

outcome			evidence
$\dagger$	$\star$	1	$J_{4,1}$
0.0	0.0	1.0	$\dagger \lor \star \lor 1$
0.0	1.0	0.0	$2 \lor 3$

(h) Replica $J_{5,1}$.

outcome			evidence	
$\dagger$	$\star$	1	$J_{5,1}$	$J_{2,2}$
1.0	0.0	0.0	$\dagger \lor 1$	
0.0	0.0	1.0	$\star$	$\dagger \lor \star \lor 1$
0.0	1.0	0.0	$\star$	2

(i) Replica $J_{5,2}$.

outcome			evidence
$\dagger$	$\star$	1	$J_{4,1}$
0.0	0.0	1.0	$\dagger \lor \star \lor 1$
0.0	1.0	0.0	$2 \lor 3$

(j) Replica $J_{6,1}$.

Table 3: Execution probabilities of individual replicas in schedule $\boldsymbol{s}$.

	$J_{1,1}$	$J_{2,1}$	$J_{2,2}$	$J_{3,1}$	$J_{3,2}$	$J_{3,3}$	$J_{4,1}$	$J_{5,1}$	$J_{5,2}$	$J_{6,1}$	$J_{7,1}$
$\Pr\{J_{i,q} \succeq 1\}$	1.0	0.8	0.2	0.68	0.32	0.0	1.0	0.5	0.46	0.5	1.0

$\Pr\{J_{i,q}\}$ as they are marginalized out during the computation of (3.2). Hence, for each job in the schedule, we can define a smaller Bayesian network containing only its ancestor vertices. In fact, this is a concept related to the question of relevant nodes in Bayesian inference for a query with the given set of evidence nodes.

Definition 9 (Bayesian network with respect to a job replica) *Bayesian network with respect to job replica $J_{i,q}$, denoted by $G'_{i,q}$, is a subgraph of $G(\boldsymbol{s})$ induced by the set of vertices reachable from $J_{i,q}$ with the reversed orientation of edges in $G(\boldsymbol{s})$.*

For example, in Figure 9b, the Bayesian network with respect to $J_{3,2}$ has vertices $V(G'_{3,2}) = \{J_{1,1}, J_{2,1}, J_{2,2}, J_{3,1}, J_{3,2}\}$ and edges incident with $V(G'_{3,2})$. The inference is then performed in the network $G'_{i,q}$ for all replicas $J_{i,q}$ independently. Note that in specific cases, the Bayesian network with respect to some job replica might be as large as the original network $G(\boldsymbol{s})$. Indeed, for the schedule used in hardness reduction in Figure 7, the network with respect to job Y is identical to the whole network, i.e., $G'_{Y,1} = G(\boldsymbol{s})$. However, in the next section, we show that under realistic assumptions, the restricted networks are much smaller.

The complete algorithm that computes $\sum P_i$ is given in Algorithm 2. The

Algorithm 2 Computation of $\sum P_i$ in schedule $\boldsymbol{s}$.

```
    Input: a schedule $\boldsymbol{s}$
    Output: the sum of execution probabilities of jobs $\sum P_i$
1: for $J_i \in I_{\mathcal{MC}}$ do
2:     for $q = 1$ to $n_i$ do
3:         $G'_{i,q} \leftarrow$ BN with respect to $J_{i,q}$ in $\boldsymbol{s}$
4:         $P_{i,q} \leftarrow \Pr\{J_{i,q} \succeq 1\}$ in $G'_{i,q}$          $\triangleright$ inference in $G'_{i,q}$
5:     end for
6:     $P_i \leftarrow \sum_q P_{i,q}$
7: end for
8: return $\sum P_i$
```

computed execution probabilities for the example in Figure 9a can be seen in Table 3.

Remark 4 Note that there are three aspects that affect the particular values of the computed probabilities. Namely, it is the probability distribution of processing times of the jobs involved, the maximum number of replicas $\mathcal{R}$, and their actual schedule. For example, with $\mathcal{R} = 2$, it would be possible to achieve exactly the same execution probabilities as presented in Table 3 since the presence of $J_{3,3}$ replica does not increase the execution probability of job J_3 any further. However, with the identical set of jobs but with a different schedule, the result might be different. Therefore, it is a complex question with what value of $\mathcal{R}$ the schedules should be constructed. Another related question is whether with the increasing value of $\mathcal{R}$, the marginal improvements in the execution probability are non-increasing and if yes, how quickly the marginal improvements become negligible to the point where it is effectively meaningless to increase it further. However, in practice, the number of maximum replicas is often treated as a design parameter, not a variable to be optimized. Thus, the cases of the practical interest contain problems where $\mathcal{R}$ is fixed to a particular constant.

3.2 Tractable case

In this section, we show that when both $\mathcal{L}$ and $\mathcal{R}$ are bounded by a constant independent from the input length, the computation becomes tractable. We note that this case is arguably the most practical one concerning real-world applications. Additionally, this case is also tight in the sense that when one of the $\mathcal{L}$ or $\mathcal{R}$ is not fixed, then the problem becomes intractable, as shown by Propositions 1 and 2.

The main idea is to realize that computationally the most intensive step in Algorithm 2 is the inference in BN $G'_{i,q}$. However, it can be shown that the size of the restricted network is limited by parameters $\mathcal{L}$ and $\mathcal{R}$. When both $\mathcal{L}$ and $\mathcal{R}$ are bounded by a constant, then the number of vertices $V(G'_{i,q})$ is independent of the number of *all* replicas in the schedule (i.e., does not depend on $n = |I_{\mathcal{MC}}|$, but only on $\mathcal{R}$ and $\mathcal{L}$).

Proposition 3 *Let $G'_{i,q}$ be a Bayesian network with respect to arbitrary $J_{i,q}$. Then it holds that*

$$\left|V(G'_{i,q})\right| \in \mathcal{O}\left((\mathcal{R}\mathcal{L})^{\mathcal{L}}\right).$$

Proof The general idea is to follow the definition of how the networks are built in order to construct the largest possible network with the given fixed values of $\mathcal{R}$ and $\mathcal{L}$. Then, we bound the number of vertices in the resulting graph.

Let us consider a BN with respect to some job $J_{i,q}$ for the problem with $\mathcal{L}$ criticality levels, and $\mathcal{R}$ maximum replicas. Without loss of generality, let us assume that $\mathcal{X}_i = 1$. Let us organize $G'_{i,q}$ into levels, where level $L_\ell \subseteq V(G'_{i,q})$ contains all job replicas $J_{k,r}$ with criticality $\mathcal{X}_k = \ell$, such that $V(G'_{i,q}) = \bigcup_{\ell=1}^{\mathcal{L}} L_\ell$. Assuming we have at most $\mathcal{R}$ replicas per job, we have that $|L_1| \leq \mathcal{R}$. By (3.1), we have that every vertex has in-degree at most $(\mathcal{L}-1)+1 = \mathcal{L}$. Hence, the upper bound on the number of vertices in the second level is $|L_2| \leq \mathcal{R}^2 \times \mathcal{L}$. For level ℓ, we have that $|L_\ell| \leq \mathcal{R}^\ell \times \mathcal{L}^{\ell-1}$. Since we have $\mathcal{L}$ criticality levels, graph $G'_{i,q}$ contains at most $\left|V(G'_{i,q})\right| \leq \sum_{\ell=1}^{\mathcal{L}} |L_\ell| = \mathcal{R}+\mathcal{R}^2\times\mathcal{L}+\ldots+\mathcal{R}^{\mathcal{L}}\times\mathcal{L}^{\mathcal{L}-1} = \mathcal{R}\times\frac{(\mathcal{R}\mathcal{L})^{\mathcal{L}}-1}{\mathcal{R}\mathcal{L}-1}$ vertices. Therefore, the cardinality of the vertex set is a function of $\mathcal{R}$ and $\mathcal{L}$ only. □

Remark 5 Note that the upper bound suggested by Proposition 3 is overly pessimistic since we have assumed that each job replica $J_{i,q}$ is covered by $\mathcal{L}-1$ different job replicas of criticality $\mathcal{X}_i + 1$, which cannot occur. Hence, with more detailed analysis the upper bound could be reduced.

Next, we discuss what is the size of CPTs in $G'_{i,q}$. Since in-degree of each vertex is at most $\mathcal{L}$ by (3.1), then its CPT has size at most $\mathcal{O}((\mathcal{L}+2)^{\mathcal{L}})$, even under a trivial encoding given by a full CPT. Hence, the total size of the representation of BN $G'_{i,q}$ is independent of the number of jobs n. The total time complexity of Algorithm 2 is $\mathcal{O}(n \cdot \mathcal{R} \cdot f(\mathcal{L}, \mathcal{R}))$, where $f(\mathcal{L}, \mathcal{R})$ is complexity of the used inference algorithm in BN $G'_{i,q}$.

Finally, let us discuss inference complexity term $f(\mathcal{L}, \mathcal{R})$ in BN $G'_{i,q}$. The efficiency of exact inference algorithms is limited by the properties of conditional distributions as well as by the structure of networks. It is known that networks satisfying *local variance bound* (LVB), i.e., a requirement that forbids extreme conditional distributions, admit a subexponential deterministic inference algorithm [12]. Unfortunately, in our case, conditional distributions do not satisfy LVB due to the execution policy (a replica is rejected with the probability of one when the previous replica is executed). Concerning the structural properties of the network, it is known that they are mostly related to the treewidth of a graph. Informally, the treewidth of a graph is a quantity related to its connectivity. For example, the treewidth of the least possible connected graph (i.e., a tree) is equal to 1 whereas the complete graph with n vertices has treewidth equal to $n-1$. The result of [26] suggests that under reasonable assumptions, no polynomial algorithm exists for networks with unbounded treewidth. A similar idea to avoid intractability was applied in [36], where they focus on tree networks. In our case, the networks have bounded treewidth by a function of $\mathcal{L}$ and $\mathcal{R}$ which avoids the dependence on the total number of jobs n. Thus, its complexity does not depend on the number of jobs n. What is more, the practical experience suggests that, in an average case of mixed-criticality schedules, the treewidth of our networks achieves much smaller values.

Finally, we note that the current implementations of Bayesian network solvers easily handle computations in networks with hundreds to thousands of vertices (i.e., replicas) within seconds [35]. Hence, we see the proposed method as a computationally efficient way of solving the problem.

4 Conclusion

In this work, we have introduced the replication of jobs as a mechanism for increasing the execution probability in mixed-criticality schedules. We have studied the complexity of the computation of execution probability in the given static schedule. Our main result shows that there are two primary parameters that influence the complexity — the maximum number of replicas per job $\mathcal{R}$ and the number of criticality levels $\mathcal{L}$. We have shown that although the replication significantly improves the execution probability in mixed-criticality schedules, it introduces additional complexity to the problem and the exact computation becomes $\#\mathcal{P}$-hard. In fact, the problem remains hard if either the maximum number of replicas $\mathcal{R} \geq 3$ or the number of criticality levels $\mathcal{L} \geq 8$ is fixed while the other quantity is bounded by a polynomial in the number of jobs.

To solve the problem, we have proposed a reduction to probabilistic inference in Bayesian networks, showing an interesting connection between schedules with uncertain execution and probabilistic graphical models. The analysis of the resulting networks shows that for the practical case when both $\mathcal{R}$ and $\mathcal{L}$ are bounded by a constant, the computation becomes tractable. Considering available implementations of exact and approximate inference algorithms for Bayesian networks, the problem can be efficiently solved in practice as well, offering a viable choice for improving the efficiency of static schedules for mixed-criticality systems.

Acknowledgment

This work was supported by the EU and the Ministry of Industry and Trade of the Czech Republic under the Project OP PIK CZ.01.1.02/0.0/0.0/20_321/0024399, and by the European Regional Development Fund under the project AI&Reasoning (reg. no. CZ.02.1.01/0.0/0.0/15_003/0000466). Next, we want to thank Daniel Slunecko for his early work on the problem and inspiring discussions.

References

1. Agirre, I., Cazorla, F.J., Abella, J., Hernandez, C., Mezzetti, E., Azkarate-askatsua, M., Vardanega, T.: Fitting software execution-time exceedance into a residual random fault in ISO-26262. IEEE Transactions on Reliability **67**(3), 1314–1327 (2018). DOI 10.1109/TR.2018.2828222
2. Baruah, S.: Predictability Issues in Mixed-Criticality Real-Time Systems, pp. 77–87. Springer International Publishing, Cham (2018). DOI 10.1007/978-3-319-95246-8˙5. URL `https://doi.org/10.1007/978-3-319-95246-8_5`
3. Baruah, S., Bonifaci, V., D'angelo, G., Li, H., Marchetti-Spaccamela, A., Van Der Ster, S., Stougie, L.: Preemptive uniprocessor scheduling of mixed-criticality sporadic task systems. Journal of the ACM (JACM) **62**(2), 14 (2015)
4. Baruah, S., Fohler, G.: Certification-cognizant time-triggered scheduling of mixed-criticality systems. In: Real-Time Systems Symposium (RTSS), 2011 IEEE 32nd, pp. 3–12. IEEE (2011)
5. Behera, L., Bhaduri, P.: Time-triggered scheduling for multiprocessor mixed-criticality systems. In: Distributed Computing and Internet Technology, pp. 135–151. Springer International Publishing, Cham (2018)
6. Bell, R.: Introduction to IEC 61508. In: Proceedings of the 10th Australian workshop on Safety critical systems and software-Volume 55, pp. 3–12. Australian Computer Society, Inc. (2006)
7. Blazewicz, J., Ecker, K.H., Pesch, E., Schmidt, G., Weglarz, J.: Handbook on scheduling: from theory to applications. Springer Science & Business Media (2007)
8. Burns, A., Davis, R.I.: A survey of research into mixed criticality systems. ACM Comput. Surv. **50**(6), 82:1–82:37 (2017). DOI 10.1145/3131347. URL `http://doi.acm.org/10.1145/3131347`
9. Burns, A., Davis, R.I., Baruah, S., Bate, I.: Robust mixed-criticality systems. IEEE Transactions on Computers **67**(10), 1478–1491 (2018)
10. Chang, Z., Ding, J.Y., Song, S.: Distributionally robust scheduling on parallel machines under moment uncertainty. European Journal of Operational Research **272**(3), 832 – 846 (2019). DOI https://doi.org/10.1016/j.ejor.2018.07.007. URL `http://www.sciencedirect.com/science/article/pii/S037722171830612X`

11. Creignou, N., Hermann, M.: On P completeness of some counting problems. Ph.D. thesis, INRIA (1993)
12. Dagum, P., Luby, M.: An optimal approximation algorithm for Bayesian inference. Artificial Intelligence **93**(1-2), 1–27 (1997)
13. Daniels, R.L., Carrillo, J.E.: β-robust scheduling for single-machine systems with uncertain processing times. IIE transactions **29**(11), 977–985 (1997)
14. Davis, R.I., Altmeyer, S., Burns, A.: Mixed criticality systems with varying context switch costs. In: 2018 IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS), pp. 140–151 (2018). DOI 10.1109/RTAS.2018.00024
15. Draskovic, S., Huang, P., Thiele, L.: On the safety of mixed-criticality scheduling. In: Proceedings of the 4th International Workshop on Mixed Criticality Systems, RTSS, pp. 19 – 24. IEEE, Porto, Portugal (2016)
16. El-Hajj, R., Guibadj, R.N., Moukrim, A., Serairi, M.: A PSO based algorithm with an efficient optimal split procedure for the multiperiod vehicle routing problem with profit. Annals of Operations Research pp. 1–36 (2020)
17. Graham, R.L., Lawler, E.L., Lenstra, J.K., Kan, A.R.: Optimization and approximation in deterministic sequencing and scheduling: a survey. Annals of discrete mathematics **5**, 287–326 (1979)
18. Guo, H., Hsu, W.: A survey of algorithms for real-time Bayesian network inference. In: Join Workshop on Real Time Decision Support and Diagnosis Systems (2002)
19. Hamaz, I., Houssin, L., Cafieri, S.: A branch-and-bound procedure for the robust cyclic job shop problem. In: International Symposium on Combinatorial Optimization, pp. 228–240. Springer (2018)
20. Hanzalek, Z., Sucha, P.: Time symmetry of resource constrained project scheduling with general temporal constraints and take-give resources. Annals of Operations Research **248**(1-2), 209–237 (2017)
21. Hanzalek, Z., Tunys, T., Sucha, P.: An analysis of the non-preemptive mixed-criticality match-up scheduling problem. Journal of Scheduling **19**(5), 601–607 (2016). DOI 10.1007/s10951-016-0468-y. URL `https://doi.org/10.1007/s10951-016-0468-y`
22. Herroelen, W., Leus, R.: Project scheduling under uncertainty: Survey and research potentials. European Journal of Operational Research **165**(2), 289 – 306 (2005). DOI https://doi.org/10.1016/j.ejor.2004.04.002. URL `http://www.sciencedirect.com/science/article/pii/S0377221704002401`. Project Management and Scheduling
23. Ishfaq Ahmad, Yu-Kwong Kwok: On exploiting task duplication in parallel program scheduling. IEEE Transactions on Parallel and Distributed Systems **9**(9), 872–892 (1998). DOI 10.1109/71.722221
24. Jaramillo, F., Keles, B., Erkoc, M.: Modeling single machine preemptive scheduling problems for computational efficiency. Annals of Operations Research **285**(1), 197–222 (2020)
25. Kopetz, H.: Event-triggered versus time-triggered real-time systems, pp. 86–101. Springer Berlin Heidelberg, Berlin, Heidelberg (1991). DOI 10.1007/BFb0024530. URL `https://doi.org/10.1007/BFb0024530`
26. Kwisthout, J., Bodlaender, H.L., van der Gaag, L.C.: The necessity of bounded treewidth for efficient inference in Bayesian networks. In: ECAI, vol. 215, pp. 237–242 (2010)
27. Li, Y.F., Huang, H.Z., Mi, J., Peng, W., Han, X.: Reliability analysis of multi-state systems with common cause failures based on Bayesian network and fuzzy probability. Annals of Operations Research pp. 1–15 (2019)
28. Murphy, K.P., Weiss, Y., Jordan, M.I.: Loopy belief propagation for approximate inference: An empirical study. In: Proceedings of the Fifteenth conference on Uncertainty in artificial intelligence, pp. 467–475. Morgan Kaufmann Publishers Inc. (1999)
29. Novak, A., Sucha, P., Hanzalek, Z.: Scheduling with uncertain processing times in mixed-criticality systems. European Journal of Operational Research **279**(3), 687 – 703 (2019). DOI https://doi.org/10.1016/j.ejor.2019.05.038. URL `http://www.sciencedirect.com/science/article/pii/S0377221719304680`
30. Obermaisser, R., Kopetz, H., El Salloum, C., Huber, B.: Error containment in the time-triggered system-on-a-chip architecture. In: Embedded System Design: Topics, Techniques and Trends, pp. 339–352. Springer (2007)

31. Papadimitriou, C.H., Yannakakis, M.: Towards an architecture-independent analysis of parallel algorithms. SIAM journal on computing **19**(2), 322–328 (1990)
32. Paredes, R., Dueñas-Osorio, L., Meel, K., Vardi, M.: Principled network reliability approximation: A counting-based approach. Reliability Engineering & System Safety **191**, 106472 (2019). DOI https://doi.org/10.1016/j.ress.2019.04.025. URL `http://www.sciencedirect.com/science/article/pii/S0951832018305209`
33. Ranjbar, M., Davari, M., Leus, R.: Two branch-and-bound algorithms for the robust parallel machine scheduling problem. Computers & Operations Research **39**(7), 1652 – 1660 (2012). DOI https://doi.org/10.1016/j.cor.2011.09.019. URL `http://www.sciencedirect.com/science/article/pii/S0305054811002802`
34. Russell, S.J., Norvig, P.: Artificial intelligence: a modern approach. Malaysia; Pearson Education Limited, (2016)
35. Sang, T., Bearne, P., Kautz, H.: Performing bayesian inference by weighted model counting. In: Proceedings of the 20th National Conference on Artificial Intelligence - Volume 1, AAAI'05, pp. 475–481. AAAI Press (2005). URL `http://dl.acm.org/citation.cfm?id=1619332.1619409`
36. Santiváñez, J.A., Melachrinoudis, E.: Reliable maximin–maxisum locations for maximum service availability on tree networks vulnerable to disruptions. Annals of Operations Research **286**(1), 669–701 (2020)
37. Seddik, Y., Hanzalek, Z.: Match-up scheduling of mixed-criticality jobs: Maximizing the probability of jobs execution. European Journal of Operational Research **262**(1), 46 – 59 (2017). DOI http://dx.doi.org/10.1016/j.ejor.2017.03.054. URL `http://www.sciencedirect.com/science/article/pii/S037722171730276X`
38. Theis, J., Fohler, G., Baruah, S.: Schedule table generation for time-triggered mixed criticality systems. Proceedings of the 1st International Workshop on Mixed Criticality Systems, RTSS pp. 79–84 (2013)
39. Valiant, L.: The complexity of computing the permanent. Theoretical Computer Science **8**(2), 189 – 201 (1979). DOI https://doi.org/10.1016/0304-3975(79)90044-6. URL `http://www.sciencedirect.com/science/article/pii/0304397579900446`
40. Valiant, L.: The complexity of enumeration and reliability problems. SIAM Journal on Computing **8**(3), 410–421 (1979). DOI 10.1137/0208032. URL `https://doi.org/10.1137/0208032`
41. Vestal, S.: Preemptive scheduling of multi-criticality systems with varying degrees of execution time assurance. In: Real-Time Systems Symposium, 2007. RTSS 2007. 28th IEEE International, pp. 239–243. IEEE (2007)
42. Yeh, C.T.: Binary-state line assignment optimization to maximize the reliability of an information network under time and budget constraints. Annals of Operations Research **287**(1), 439–463 (2020)