Response Time Analysis for Probabilistic DAG Tasks in Multicore Real-Time Systems

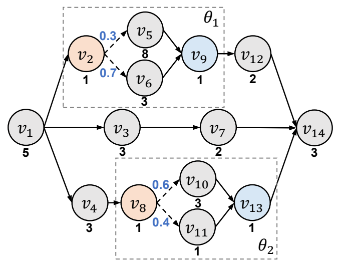
Shuai Zhao[†], Yiyang Gao[†], Zhiyang Lin[†], Boyang Li[†], Xinwei Fang[‡], Zhe Jiang[§], Nan Guan[¶]
[†]Sun Yat-sen University [‡]University of York [§]Southeast University [¶]City University of Hong Kong

***Abstract*—Parallel real-time systems often contain functionalities with complex dependencies and execution uncertainties, leading to significant timing variability which can be represented as a probabilistic distribution. However, existing timing analysis either produces a single conservative bound or incurs high computational costs due to the exhaustive enumeration of every execution scenario. This significantly hinders the exploitation of the probabilistic timing behaviours during system design, leading to sub-optimal design solutions. Modelling the system as a probabilistic directed acyclic graph (*p*-DAG), this paper presents a probabilistic response time analysis based on different longest paths of the *p*-DAG across all execution scenarios, enhancing the capability of the analysis by eliminating the need for enumeration. We first identify every longest path candidate based on the structure of *p*-DAG and compute the probability of its occurrence, where each candidate is the longest under certain execution scenarios. Then, the worst-case interfering workload is computed for each longest path candidate, forming a complete probabilistic response time distribution with correctness guarantees. Experiments show that compared to the enumeration-based approach, the proposed analysis reduces the computation cost by six orders of magnitude while maintaining a low deviation (1.04% on average and below 5% for most *p*-DAGs).**

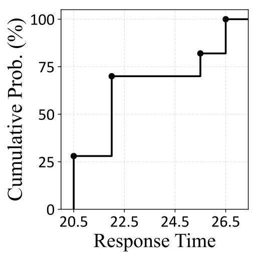
I. Introduction

The parallel tasks in multicore real-time systems (*e.g.*, automotive, avionics and robotics) often contain complex dependencies [1]–[4], and more importantly, the execution uncertainties [5]–[7], *e.g.*, the "if-else" statements that execute different functions under varied conditions. Most design and verification methods consider task dependencies by modelling the system as a directed acyclic graph (DAG) [1], [3], [4], [8], [9]. However, these methods often apply a single conservative timing bound for the system [1], [3], neglecting execution uncertainties that can cause different timing behaviours.

The execution uncertainties widely exist both within the execution of a task and between parallel tasks. Numerous studies have been conducted on the probabilistic worst-case execution time (p-WCET) of a single task [10]–[13]. However, limited results are reported that address the problem of DAGs with probabilistic executions (p-DAG), which are commonly found in real-world applications such as autonomous driving systems [6], [14]–[17]. Fig. 1(a) presents an example p-DAG with two probabilistic structures, each containing branches with different execution probabilities. In each release, only one



(a) The structure of a p-DAG.



(b) Response time distribution.

Fig. 1: A p-DAG with two probabilistic structures and its cumulative probability distribution in response time *(numbers in black: node WCET; numbers in blue: execution probability)*.

branch of every probabilistic structure can execute, yielding different DAG structures with varied response times.

In systems modelled by a p-DAG, significant variability may arise in its timing behaviours, which can be represented as a probabilistic timing distribution [18], [19] (see Fig. 1(b)). For such systems, more informative decision-making and design solutions can be made by leveraging the probabilistic timing behaviours. For instance, it is demonstrated in [6] that given the maximum number of tolerable consecutive deadline misses, the number of cores and the budgets reserved on each core for executing a p-DAG can be effectively reduced based on its probabilistic response time distribution. However, the traditional analysis is less effective for such systems due to the insufficient and overly pessimistic analytical results [6]. Therefore, a timing analysis that can capture the probability of different timing behaviours of a p-DAG is essential.

The existing p-DAG analysis (*e.g.*, the one used in [6] for system optimisation) produces the probabilistic response time distribution by enumerating through all execution scenarios of the p-DAG, with Graham's bound [8] applied to analyse the traditional DAG of each scenario. Fig. 1(b) illustrates the cumulative probability distribution of the response times for the example p-DAG. However, this approach incurs high computational costs due to the need for exhaustive recursion of all execution scenarios (as shown in Sec. VI). This significantly limits its application in facilitating system design and optimisations, especially for large-scale design space exploration, posing challenges for the design and verification of systems involving p-DAGs.

Contributions. This paper presents a probabilistic response time analysis of a p-DAG, which eliminates the need for

This work is supported by the National Key Research and Development Program under Grant 2024YFB4405600, National Natural Science Foundation of China (NSFC) under Grant 62302533 and Grant 92464301, Guangdong Basic and Applied Basic Research Foundation under Grant 2024A1515010240. Corresponding author: Zhe Jiang, zhejiang.uk@gmail.com.

enumeration by leveraging the longest paths across all the execution scenarios. To achieve this, (i) we first identify the set of longest path candidates of the p-DAG by determining a lower bound on the candidates of the p-DAG and its substructures, where a candidate is the longest path in a certain scenario. (ii) For each identified candidate, an analysis is constructed to compute the probability where it is executed and is the longest path. (iii) Finally, the worst-case interfering workload associated with each longest path candidate is computed, forming a complete probabilistic response time distribution of the p-DAG with correctness guarantees. (iv) Experiments show that compared to the enumeration-based approach, the proposed analysis reduces the computation cost by six orders of magnitude while maintaining a deviation of only 1.04% on average (below 5% for most p-DAGs), enhancing the capability in analysing large and complex p-DAGs. In addition, we demonstrate that given a pre-defined time limit and a miss rate, the proposed analysis effectively empowers design solutions that demand fewer cores, including systems with large p-DAGs.

To the best of our knowledge, this is the first probabilistic response time analysis for p-DAGs. Notably, the analysis can be used in combination with a probabilistic WCET analysis (*e.g.*, the one in [20]), in which a node with a probabilistic WCET can be effectively modelled as a probabilistic structure with the associated WCETs and probabilities. With the p-DAG analysis, the probabilistic timing behaviours of parallel tasks can be fully leveraged to produce flexible and optimised design solutions (*e.g.*, through feedback-based online decision-making) that can meet specific timing requirements.

Organisation. The rest of the paper is organised as follows. Sec. II presents the system model, the existing analysis for p-DAGs and the motivation of this work. Sec. III identifies the longest path candidates of a p-DAG. Then, Sec. IV presents the probabilistic analysis on every candidate with correctness guarantee. Sec. V presents the complete probabilistic response time distribution of p-DAGs with the worst-case interfering workload considered. Finally, Sec VI presents the experimental results, and Sec VII draws the conclusion.

II. Preliminaries

This work focuses on the timing analysis of periodic p-DAG tasks running on m symmetric cores. Below we provide the task model of a p-DAG, the existing response time analysis, and the motivation of this work. The notations introduced in this section are summarised in Tab. I.

A. Task Model of a p-DAG

As with the traditional DAG model [1], [3], a p-DAG task is defined by $\tau = \{\mathcal{V}, \mathcal{E}, T, D\}$, where $\mathcal{V}$ represents a set of nodes, $\mathcal{E}$ denotes a set of directed edges, T is the period and D is the deadline. A node $v_i \in \mathcal{V}$ indicates a series of computations that must be executed sequentially. The worst-case execution time (WCET) of v_i is defined as C_i[1]. An edge $e_{i,j} \in \mathcal{E}$ represents the execution dependency between v_i and v_j, where v_j can start only after the completion of v_i. A path $\lambda_h = \{v_{src}, ..., v_{snk}\}$ is a sequence of nodes in which every two consecutive nodes are connected by an edge, *i.e.*, $e_{i,i+1} \in \mathcal{E}, \forall v_i \in \lambda_h \setminus \{v_{snk}\}$. The set of all paths in τ is denoted as Λ. As with [1], [21], we assume that τ has a single source node v_{src} and a single sink node v_{snk}. For p-DAGs with multiple source or sink nodes, this assumption can hold by adding a dummy source or sink node with a WCET of zero [1].

[1]Nodes with probabilistic WCETs can be supported by the p-DAG model, in which each of such nodes is modelled as a probabilistic structure.

TABLE I: Notations introduced in task model.

Notation	Description
$\tau = \{\mathcal{V}, \mathcal{E}, T, D\}$	A p-DAG task τ with a set of nodes $\mathcal{V}$, a set of edges $\mathcal{E}$, a period T, and a deadline D.
R	The response time of τ.
m	The number of cores in the system.
v_i	A node in τ.
C_i	The worst-case execution time of v_i.
v_{src} / v_{snk}	The source / sink node of τ.
$e_{i,j}$	A directed edge from v_i to v_j.
λ_h	A path in τ.
Λ	The set of all paths in τ.
Θ	The set of probabilistic structures of τ.
θ_x	A probabilistic structure in Θ.
θ_x^k	A probabilistic branch in θ_x.
$F(\theta_x^k)$	The execution probability of θ_x^k.
v_x^{ent} / v_x^{ext}	The entry / exit node of θ_x.
$\mathcal{G}$	The set of graphs that can be yielded by τ.
$\mathcal{G}_u$	A non-conditional graph in $\mathcal{G}$.
$len(\cdot)$ / $vol(\cdot)$	The length / workload of a given graph.
$H(\cdot)$	The set of associated probabilistic branches in a given graph (or a path).
$S(\cdot)$	The set of associated probabilistic structures in a given graph (or a path).
Λ^*	The set of longest path candidates of τ.
$\lvert\cdot\rvert$	The size of a given set.

In addition to the fundamental characteristics, a p-DAG contains a set of *probabilistic structures* $\Theta = \{\theta_1, \theta_2, ...\}$, *e.g.*, the "if-else" statements or nodes with a probabilistic WCET. A probabilistic structure θ_x has an entry node v_x^{ent}, an exit node v_x^{ext}, and a set of probabilistic branches $\{\theta_x^1, \theta_x^2, ...\}$. As with [21], we assume that there exists no execution dependency between nodes in different probabilistic branches. The execution of a θ_x starts from v_x^{ent} and ends at v_x^{ext}. However, during an execution, only one branch θ_x^k in θ_x is executed, which is decided by the entry node v_x^{ent}. Notation $|\theta_x|$ gives the number of branches in θ_x. For instance, θ_1 of the p-DAG in Fig. 1(a) starts from v_2 and ends at v_9, with two branches $\theta_1 = \{\theta_1^1 = \{v_5\}, \theta_1^2 = \{v_6\}\}$.

A probabilistic branch in θ_x is a subset of nodes in V that forms an unconditional sub-graph (denoted as θ_x^k), in which all nodes in θ_x^k are either executed unconditionally or not being executed at all in a release of τ. Function $F(\theta_x^k)$ denotes the execution probability of θ_x^k in θ_x, which can be obtained based on measurements and the static code analysis in [11], [13], [22]. As with [6], we assume that the execution probabilities of the branches are independent. For a $\theta_x \in \Theta$, it follows $\sum_{\theta_x^k \in \theta_x} F(\theta_x^k) = 1$. For the example p-DAG in Fig. 1(a), the execution probability of each branch is $F(\theta_1^1) = 0.3$ and

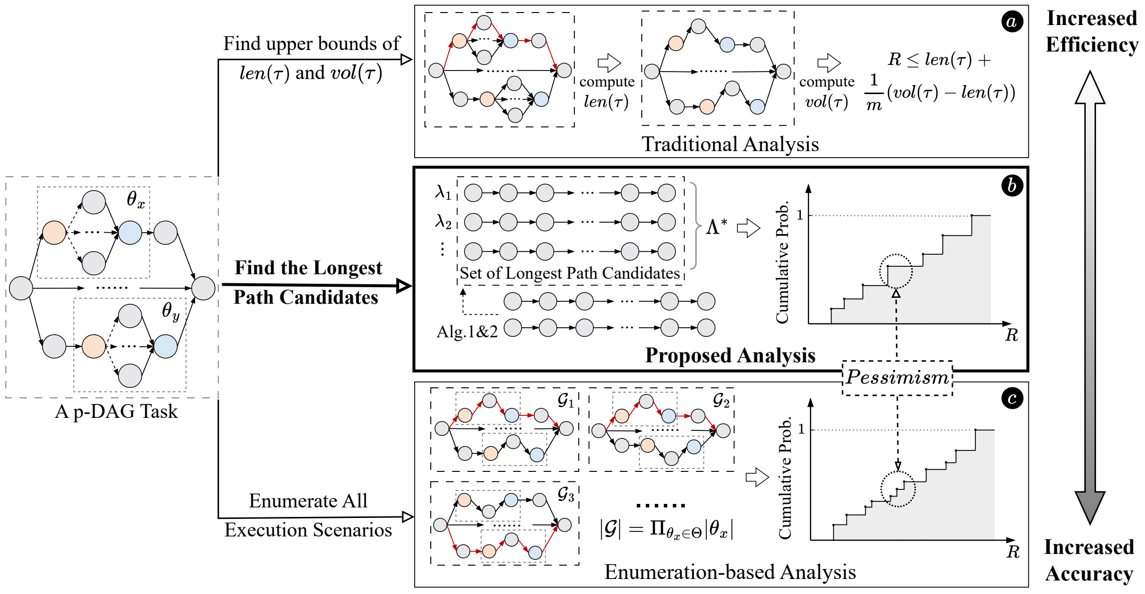


Fig. 2: Overview of the existing and the proposed analysis for p-DAGs *(a path in red indicates the longest path in a graph).*

$F(\theta_1^2) = 0.7$, respectively. Note, the proposed analysis can support p-DAGs with nested probabilistic structures by a pre-processing step that transforms the nested structures into an equivalent flattened, non-nested form using a dynamic programming algorithm. The execution probability of each branch in the flattened structure is then computed by recursively aggregating probabilities along the nesting hierarchy.

During execution, task τ can give rise to a series of jobs. In each of these jobs, one θ_x^k is chosen to be executed in every θ_x, leading to non-conditional graphs with varied structures. For instance, the p-DAG in Fig. 1(a) can yield jobs with four different graphs. The set of unique non-conditional graphs that can be yielded by τ is denoted by $\mathcal{G} = \{\mathcal{G}_1, \mathcal{G}_2, ...\}$. The number of graphs in $\mathcal{G}$ is exponential to the number of probabilistic structures and the number of probabilistic branches in each structure, *i.e.*, $|\mathcal{G}| = \Pi_{\theta_x \in \Theta}|\theta_x|$. For a non-conditional graph $\mathcal{G}_u$, $len(\mathcal{G}_u)$ gives the length of its longest path and $vol(\mathcal{G}_u)$ gives the workload. In addition, to facilitate the construction of the analysis, two helping functions are defined: Taking a graph (or a path) as input, function $H(\cdot)$ provides the set of associated branches in the given graph (path), whereas $S(\cdot)$ returns the set of associated probabilistic structures. For instance, for a path $\lambda_h = \{v_1, v_2, v_6, ..., v_{14}\}$ in Fig. 1(a), $H(\lambda_h) = \{\{v_6\}\}$ and $S(\lambda_h) = \{\{v_5\}, \{v_6\}\}$.

For all $\mathcal{G}_u \in \mathcal{G}$, notation Λ^* provides the set of unique longest paths of these graphs, termed as the *longest path candidates*. Def. 1 provides the definition of Λ^* of a p-DAG τ. For two graphs $\mathcal{G}_a$ and $\mathcal{G}_b$, they share a candidate if they have the same longest path.

Definition 1. *A p-DAG τ has a set of longest path candidates $\Lambda^* \subseteq \Lambda$, in which exactly one candidate in Λ^* is the longest path in any of the jobs released by τ.*

B. *Response Time Analysis for p-DAGs*

Existing response time analysis that handles p-DAGs can be classified by the following two approaches: (i) the traditional analysis originally developed for non-conditional DAGs [8], [21], which provides a single worst-case response time, and (ii) the enumeration-based approach [6] that examines every possible execution scenario of τ to derive a probabilistic response time distribution, providing the probability for every scenario and its associated probability. Fig. 2 presents the overall workflow of both the existing and the proposed analysis.

The majority of the existing analysis [1], [3], [7], [21], [23] provides a single timing bound on the response time. The traditional Graham's bound [8] can be applied to compute the response time R of a p-DAG by $R \leq len(\tau) + \frac{1}{m}(vol(\tau) - len(\tau))$, in which $len(\tau)$ is the length of the longest path, and $vol(\tau)$ gives the worst-case workload of τ assuming each θ_x chooses the θ_x^k with the highest workload. For p-DAGs, this approach is highly efficient, where the computational cost is caused by traversing the p-DAG to find the longest path and the worst-case workload. However, this analysis neglects the execution variability in τ, in which different execution scenarios of τ (*i.e.*, $\mathcal{G}_u \in \mathcal{G}$) can occur in a probability with varied response time. For instance, the p-DAG in Fig. 1(b) has over 70% and 25% of probability to finish within time 26 and 21, respectively. Hence, the traditional analysis fails to depict such execution uncertainties of a p-DAG, resulting in a number of design limitations such as resource over-provisioning given predefined time limits.

To account for the execution uncertainties in a p-DAG, an enumeration-based approach is applied in [6], which produces the probabilistic response time distribution by iterating through every $\mathcal{G}_u \in \mathcal{G}$, as illustrated in Fig. 2c . For each $\mathcal{G}_u$, its response time is computed by Graham's bound with a probability of occurrence computed by $\prod_{\theta_x^k \in H(\mathcal{G}_u)} F(\theta_x^k)$, *i.e.*, the execution probability in which all $\theta_x^k \in H(\mathcal{G}_u)$ are executed. With the response time and probability of every $\mathcal{G}_u$ determined, a complete probabilistic response time distribution of a p-DAG can be established, *e.g.*, the cumulative distribution in Fig. 2ⓒ . Although this approach achieves high accuracy, computing the response time and execution probability on a vast number of execution scenarios can impose significant computation cost. As described in Sec. II-A, the number of execution scenarios of τ grows exponentially with the increase of $|\Theta|$ and $|\theta_x|$ of every θ_x. Therefore, the enumeration-based approach faces significant computation costs that significantly undermine its applicability. Especially, as shown in Sec. VI, this analysis fails to provide any results for large p-DAGs with complex probabilistic structures.

C. Motivation

To address the above issues, this paper presents a timing analysis for p-DAGs by exploiting the set of the longest path candidates (Fig. 2ⓑ), providing the response time distribution of p-DAGs while eliminating the need for enumerating through every execution scenario of τ. To achieve this, the proposed analysis first identifies the exact set of longest path candidates (*i.e.*, Λ^*) by leveraging the lower bound on the length of these paths (see Alg. 1 and 2 in Sec. III). Then, for each longest path candidate, Sec. IV computes the execution probability of scenarios in which the candidate is executed as the longest path. Finally, Sec. V produces the probabilistic response time distribution of τ with the worst-case interfering workload of each longest path candidate determined, and proves the correctness of the proposed analysis.

By utilising the longest path candidates, the proposed analysis can address the efficiency issue incurred by the enumeration-based approach. However, we note that as illustrated in Fig. 2, the proposed analysis is not exact, in which pessimism can exist during the computation of (i) the execution probability of a candidate (see Sec. IV-B) and (ii) the worst-case interfering workload (Sec. V). However, compared to the enumeration-based approach [6], the proposed analysis effectively balances the efficiency and accuracy, providing a highly scalable solution for analysing p-DAG tasks with only 1.04% accuracy loss on average (Sec. VI).

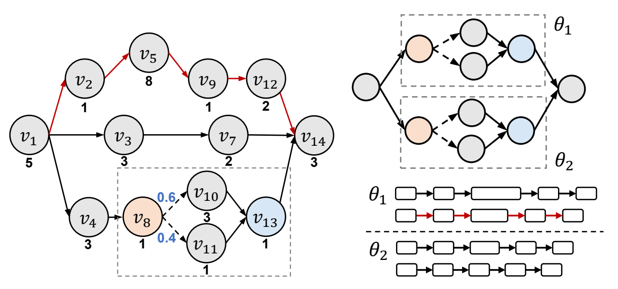
III. Identifying the Longest Paths Candidates

With existing methods, determining the longest path candidates Λ^* of τ would require the enumeration of every $\mathcal{G}_u \in \mathcal{G}$ [6], which significantly intensifies the complexity of the analysis. In addition, this can cause redundant computations as certain scenarios might have the same longest path. For instance, Fig. 3(a) shows a p-DAG with the longest path of $\{v_1, v_2, ..., v_{14}\}$ regardless of whether v_{10} or v_{11} is executed.

This section presents a method that computes Λ^* based on the lower bound on the length of all the longest path candidates of a p-DAG. To achieve this, a function $\Delta(\tau)$ is first constructed that produces the lower bound of the longest path candidates in a p-DAG (Sec. III-A). Then, we show that $\Delta(\cdot)$ can be effectively applied on τ and its sub-structures to identify Λ^*, effectively eliminating the need for enumeration over every $\mathcal{G}_u \in \mathcal{G}$ (Sec. III-B). The notations introduced in this section are summarised in Tab. II.

TABLE II: Notations Introduced in Sec. III.

Notation	Description
$\Delta(\tau)$	Lower bound on the length of paths in Λ^*.
$\mathcal{G}^\diamond$	The minimal graph yielded by τ.
$\mathcal{V}^\diamond$	The set of nodes associated with $\mathcal{G}^\diamond$.
$\mathcal{E}^\diamond$	The set of edges associated with $\mathcal{G}^\diamond$.
$\theta_x^\diamond$	The branch executed in θ_x under $\mathcal{G}^\diamond$.
$\tau_a = \{\mathcal{V}_a, \mathcal{E}_a, \Theta_a\}$	A sub-structure of τ that contains path λ_a, with $\mathcal{V}_a$, $\mathcal{E}_a$ and Θ_a denoting the set of nodes, edges and probabilistic structures in τ_a, respectively.



(a) The path with v_5 is the longest path regardless of v_{10} or v_{11} is executed.
(b) A scenario that every path in θ_1 is longer than paths in θ_2.

Fig. 3: The illustrative examples used in Sec. III.

A. Determining the Lower Bound of Λ^* for a p-DAG

To compute $\Delta(\tau)$, we first define the *minimal graph* of τ (denoted by $\mathcal{G}^\diamond$), as shown in Def. 2. Among all graphs of $\mathcal{G}$, $\mathcal{G}^\diamond$ is the one that has the minimum longest path, *i.e.*, $len(\mathcal{G}^\diamond) \leq len(\mathcal{G}_u), \forall \mathcal{G}_u \in \mathcal{G}$.

Definition 2. *The minimal graph $\mathcal{G}^\diamond$ is a non-conditional graph yield by τ with $\Delta(\tau) = len(\mathcal{G}^\diamond) \leq len(\mathcal{G}_u), \forall \mathcal{G}_u \in \mathcal{G}$.*

Thus, computing the lower bound on the length of the longest path candidates in Λ^* (*i.e.*, $\Delta(\tau)$) fundamentally requires the identification of $\mathcal{G}^\diamond$ and computing its longest path length $len(\mathcal{G}^\diamond)$. Building upon this insight, Lemma 1 provides the conditions to identify $\mathcal{G}^\diamond$ of a given τ.

Lemma 1. *Let $\theta_x^\diamond$ denote the branch executed in θ_x under $\mathcal{G}^\diamond$, it follows $len(\theta_x^\diamond) \leq len(\theta_x^k), \forall \theta_x^k \in \theta_x, \forall \theta_x \in \Theta$.*

The proof of Lemma 1 is straightforward, and hence, is omitted. With Lemma 1, $\Delta(\tau)$ can be computed as $\Delta(\tau) = len(\mathcal{G}^\diamond)$ by Def. 2. We note that Lemma 1 is a sufficient condition for $\mathcal{G}^\diamond$. In the case where τ only has one longest path candidate (*e.g.*, a non-conditional path that dominates all other paths in τ), it follows that $\Delta(\tau) = len(\mathcal{G}^\diamond) = len(\mathcal{G}_u), \forall \mathcal{G}_u \in \mathcal{G}$ regardless of the branch being executed in each θ_x. However, as described below, this would not undermine the computations of $\Delta(\tau)$ based on $\mathcal{G}^\diamond$.

Algorithm 1: Computation of $\Delta(\tau)$ for a p-DAG τ.

```
Input: $\mathcal{V}$, $\mathcal{E}$, $\Theta$
1 $\mathcal{V}^\diamond = \{v_i \mid v_i \in \mathcal{V} \wedge v_i \notin \theta_x, \forall \theta_x \in \Theta\}$;
2 /* Identify nodes in $\mathcal{G}^\diamond$ by Lemma 1. */
3 for $\theta_x \in \Theta$ do
4     $\theta_x^\diamond = argmin_{\theta_x^k}\{len(\theta_x^k) \mid \theta_x^k \in \theta_x\}$;
5     $\mathcal{V}^\diamond = \mathcal{V}^\diamond \cup \theta_x^\diamond$;
6 end
7 $\mathcal{E}^\diamond = \{e_{i,j} \mid v_i, v_j \in \mathcal{V}^\diamond \wedge e_{i,j} \in \mathcal{E}\}$;
8 return $len(\mathcal{G}^\diamond = \{\mathcal{V}^\diamond, \mathcal{E}^\diamond\})$;
```

With Lemma 1, Alg. 1 computes $\Delta(\tau)$ by constructing $\mathcal{G}^\diamond$ based on $\{\mathcal{V}, \mathcal{E}, \Theta\}$ of τ. The algorithm first initialises $\mathcal{V}^\diamond$ with all the non-conditional nodes in $\mathcal{V}$, which are executed under any graph of τ (line 1). Then, for each θ_x in Θ, the branch $\theta_x^\diamond$ is identified by $len(\theta_x^k), \forall \theta_x^k \in \theta_x$, and the corresponding nodes are added to $\mathcal{V}^\diamond$ (lines 3-6). Based on $\mathcal{V}^\diamond$, the set of edges that connect these nodes is obtained as $\mathcal{E}^\diamond$ (line 7). Finally, with $\mathcal{G}^\diamond = \{\mathcal{V}^\diamond, \mathcal{E}^\diamond\}$ constructed, $\Delta(\tau)$ is computed at line 8 using the Depth-First Search in linear complexity [3]. It is worth noting that the minimal graph $\mathcal{G}^\diamond$ of τ might not be unique, as a probabilistic structure of τ may contain different branches of the same length. In this case, any of the branches can be taken when constructing $\mathcal{G}^\diamond$. However, $\Delta(\tau)$ is a unique value, which gives the length of the longest path of $\mathcal{G}^\diamond$.

Example 1. For the p-DAG in Fig. 1(a), its $\mathcal{G}^\diamond$ is obtained when v_6 and v_{11} are executed based on Lemma 1, *i.e.*, the branch with the minimal $len(\cdot)$. Then, the length of its longest path $\{v_1, v_2, v_6, ..., v_{14}\}$ can be obtained from the $\mathcal{G}^\diamond$, with $len(\mathcal{G}^\diamond) = \Delta(\tau) = 15$ returned by Alg. 1.

B. Identifying Λ^* of a p-DAG based on $\Delta(\cdot)$

With function $\Delta(\cdot)$, Alg. 2 is constructed to compute the Λ^* of τ. Essentially, the algorithm takes the $\mathcal{V}$, $\mathcal{E}$, Θ and Λ of τ as the input, and obtains Λ^* by removing the paths that are always dominated by a longer one. First, the algorithm identifies the $\mathcal{G}^\diamond$ of τ and calculates $len(\mathcal{G}^\diamond)$ by function $\Delta(\tau)$ (line 2). Then, it initialises Λ^* by removing paths in Λ that are shorter than $len(\mathcal{G}^\diamond)$ (line 3) based on the Lemma 2 below. This effectively speeds up the analysis, as only the paths exceeding the lower bound will be examined in later computations.

Lemma 2. *For any $\lambda_h \in \Lambda^*$, it follows that $len(\lambda_h) \geq \Delta(\tau)$.*

Proof. This follows directly from Lemma 1 and Alg. 1. □

With Λ^* initialised, Alg. 2 further determines whether a path is always dominated by another in all scenarios (lines 4-18). For two paths λ_a and λ_b with $a < b$ (where a and b denote the unique indices of paths in Λ), their execution relationships can be categorised into the following three cases.

C-1. λ_a and λ_b are never executed in the same $\mathcal{G}_u$, *i.e.*, a $\theta_x \in S(\lambda_a) \cap S(\lambda_b)$ exists such that $\theta_x^\alpha \in H(\lambda_a) \wedge \theta_x^\beta \in H(\lambda_b)$. This indicates λ_a and λ_b choose different branches for the same probabilistic structure.

Algorithm 2: Calculation of Λ^*

```
Input: $\mathcal{V}$, $\mathcal{E}$, $\Theta$, $\Lambda$
1 /* Initialise the $\Lambda^*$ of $\tau$ by Lemma 2*/
2 $len(\mathcal{G}^\diamond) = \Delta(\tau)$ by Alg. 1;
3 $\Lambda^* = \{\lambda_h \mid \lambda_h \in \Lambda \wedge len(\lambda_h) \geq len(\mathcal{G}^\diamond)\}$;
4 for $\lambda_a, \lambda_b \in \Lambda^*$ with $a < b$ do
5     /* Find dominant path of C-2 by Lemma 3 */
6     if $H(\lambda_a) = H(\lambda_b) \wedge len(\lambda_a) \geq len(\lambda_b)$ then
7         $\Lambda^* = \Lambda^* \setminus \lambda_b$;
8     end
9     /* Find dominant path of C-3 by Lemma 4 */
10    if Lemma 4 holds for $\lambda_a$ and $\lambda_b$ then
11        $\Theta_a = \{\theta_x \mid \theta_x \in S(\lambda_a) \wedge \theta_x \notin S(\lambda_b)\}$;
12        $\mathcal{V}_a = \lambda_a \cup \Theta_a$;
13        $\mathcal{E}_a = \{e_{i,j} \mid v_i, v_j \in \mathcal{V}_a \wedge e_{i,j} \in \mathcal{E}\}$;
14        if $\Delta(\tau_a = \{\mathcal{V}_a, \mathcal{E}_a, \Theta_a\}) \geq len(\lambda_b)$ then
15            $\Lambda^* = \Lambda^* \setminus \lambda_b$;
16        end
17    end
18 end
19 return $\Lambda^*$;
```

C-2. λ_a and λ_b are always executed or not simultaneously in any $\mathcal{G}_u \in \mathcal{G}$, *i.e.*, $H(\lambda_a) = H(\lambda_b)$. This indicates that the paths λ_a and λ_b pass through the same set of probabilistic structures, and choose the same branch in each probabilistic structure. Note that $H(\lambda_a) = H(\lambda_b)$ implies $S(\lambda_a) = S(\lambda_b)$.

C-3. λ_a and λ_b can be executed in the same or in different graphs, *i.e.*, $S(\lambda_a) \neq S(\lambda_b)$ and $\forall \theta_x \in S(\lambda_a) \cap S(\lambda_b)$, $\theta_x^k \in H(\lambda_a) \cap H(\lambda_b)$. In this case, whether λ_a and λ_b can appear in a graph depends on the branches being executed within the probabilistic structures that do not belong to $S(\lambda_a) \cap S(\lambda_b)$.

For C-1, there exists no dominance relationship between λ_a and λ_b as they are not executed simultaneously in any $\mathcal{G}_u$. Below we focus on determining whether λ_b is always dominated under C-2 and C-3.

Forλ_a and λ_b (with $a < b$) that satisfy C-2 (*i.e.*, $H(\lambda_a) = H(\lambda_b)$), it indicates that they pass through the exactly same set of branches. Hence, the two paths are either both executed or not being executed in any $\mathcal{G}_u \in \mathcal{G}$. In such cases, λ_b is not the longest in any $\mathcal{G}_u$ if $len(\lambda_a) \geq len(\lambda_b)$, as described in Lemma 3. Following this, the algorithm removes λ_b from Λ^* if the conditions in Lemma 3 hold (lines 6-8). In addition, if (λ_a, λ_b) is examined, (λ_b, λ_a) will not be considered by Alg. 2.

Lemma 3. *For paths λ_a and λ_b with $a < b$ and $H(\lambda_a) = H(\lambda_b)$, $\lambda_b \notin \Lambda^*$ if $len(\lambda_a) \geq len(\lambda_b)$.*

Proof. Suppose there exists a $\mathcal{G}_u \in \mathcal{G}$ in which λ_b is executed, then every branch $\theta_x^k \in H(\lambda_b)$ must be executed in $\mathcal{G}_u$. Given that $H(\lambda_a) = H(\lambda_b)$, λ_a is also be executed in $\mathcal{G}_u$. In this case, if $len(\lambda_a) \geq len(\lambda_b)$, λ_b is always dominated by λ_a and should be removed from Λ^*. Therefore, the lemma holds. □

For C-3, it is challenging to directly determine the dominance relationship between λ_a and λ_b, as they can be executed in different graphs. However, if λ_a is not executed, an alternative path with the same $S(\lambda_a)$ will be executed, as one branch of each $\theta_x \in \Theta$ must be executed in a graph. Thus, if λ_b is shorter than any of such paths, it is not the longest path in any graph. For instance, Fig. 3(b) presents a p-DAG in which the shortest path that goes through θ_1 is longer than that of θ_2. Hence, for this example, a path that goes through θ_2 is always dominated by one that goes through θ_1.

To determine whether λ_b is dominated under C-3, a sub-structure of τ is constructed based on λ_a and its alternative paths, denoted by $\tau_a = \{\mathcal{V}_a, \mathcal{E}_a, \Theta_a\}$ (lines 11-13). First, Θ_a is constructed by the unique probabilistic structures of λ_a *i.e.*, $\theta_x \in S(\lambda_a) \wedge \theta_x \notin S(\lambda_b)$. Then, $\mathcal{V}_a$ and $\mathcal{E}_a$ are computed by nodes in λ_a and Θ_a. Based on $\Delta(\tau_a)$, Lemma 4 describes the case where λ_b is dominated under C-3, and hence, is removed from Λ^* by the algorithm (lines 14-16).

Lemma 4. *For λ_a and λ_b with $a < b$ that satisfy the conditions in C-3, it follows that $\lambda_b \notin \Lambda^*$ if $\Delta(\tau_a) \geq len(\lambda_b)$.*

Proof. Suppose there exists a path $\lambda_b \in \Lambda^*$ with $\Delta(\tau_a) \geq len(\lambda_b)$. In this case, there always exists a path in τ_a that is equal to or higher than $\Delta(\tau_a)$, *i.e.*, this path no shorter than λ_b. Hence, λ_b is not the longest path under any execution scenario of τ_a, which contradicts with the assumption that $\lambda_b \in \Lambda^*$. Therefore, the lemma holds. □

After every two candidate paths are examined, the algorithm terminates with Λ^* of τ returned (line 19). Example 2 illustrates the computation process for obtaining the Λ^* of τ.

Example 2. For the p-DAG in Fig. 1(a), Λ^* is initialised as $\Lambda^* = \{\lambda_1 = \{v_1, v_2, v_5, v_9, ...\}, \lambda_2 = \{v_1, v_2, v_6, v_9, ...\}, \lambda_3 = \{v_1, v_4, v_8, v_{10}, ...\}\}$ with $len(\mathcal{G}^\diamond) = 15$, $len(\lambda_1) = 20$, $len(\lambda_2) = 15$ and $len(\lambda_3) = 16$. However, there exists no dominance relationship between the paths. Although λ_1 and λ_3 follow C-3, they do not meet the condition in Lemma 4 as $\Delta(\tau_1) = 15 < len(\lambda_3) = 16$. Therefore, $\Lambda^* = \{\lambda_1, \lambda_2, \lambda_3\}$.

Most importantly, by utilising $\Delta(\cdot)$ and the relationship of occurrence between the paths in a p-DAG, Alg. 2 identifies the exact set of longest path candidates of the p-DAG without the need for enumerating through every $\mathcal{G}_u \in \mathcal{G}$, as shown in Theorem 1. This provides the foundation for the constructed analysis of p-DAGs and serves as the key to improving the efficiency of the analysis.

Theorem 1. *Alg. 2 produces the exact Λ^* of a p-DAG τ.*

Proof. First, for a path $\lambda_b \notin \Lambda^*$, there always exists a path $\lambda_a \in \Lambda^*$ with a length equal to or higher than the length of λ_b that is executed together with λ_b. This is guaranteed by Lemmas 2, 3 and 4, which removes λ_b from Λ^* if it is lower than $\Delta(\tau)$, λ_a or $\Delta(\tau_a)$, respectively. Second, for any path $\lambda_a \in \Lambda^*$, there exists a graph $\mathcal{G}_u \in \mathcal{G}$ in which λ_a is the longest. Assume that $\lambda_a \in \Lambda^*$ is not the longest in any graph, λ_a is always dominated by a path $\lambda_b \in \Lambda^*$ or its alternative paths in τ_b, and hence, will be removed based on the lemmas. Therefore, the theorem holds. □

TABLE III: Notations introduced in Sec. IV.

Event	**Definition**
λ_l / λ_s	A path that is longer / shorter than path λ_h.
E_h	The event that λ_h is executed.
$\overline{E_h}$	The event that λ_h is not being executed.
E^*_h	The event that λ_h is executed as the longest path.
$P(\cdot)$	The probability of a given event.
$\mathbb{P}(E^*_h)$	The cumulative probability that the length of the longest path is equal to or higher than $len(\lambda_h)$.
$\mathbb{P}^\S(E^*_h)$	The ground truth of $\mathbb{P}(E^*_h)$.
δ_h	Deviation of estimated $P(E^*_h)$ against the ground truth.

The time complexity for computing Λ^* is quartic. First, Alg. 1 examines each θ^k_x of every $\theta_x \in \Theta$, leading to a complexity of $\mathcal{O}(|\Theta|\cdot|\theta|)$, where $|\theta|$ denotes the maximum number of branches of the probabilistic structures in τ. For Alg. 2, at most $|\Lambda|^2$ iterations are performed to examine the paths, where each iteration can invoke Alg. 1 once. Hence, the time complexity of Alg. 2 is $\mathcal{O}(|\Lambda|^2\cdot|\Theta|\cdot|\theta|)$. In addition, we note that a number of optimisations can be conducted to speed up the computations, *e.g.*, λ_a can be removed directly if $len(\lambda_a) \leq len(\lambda_b)$ at lines 6-8. However, such optimisations are omitted to ease the presentation.

IV. Probabilistic Analysis of the Longest Path

This section analyses the probability where a path $\lambda_h \in \Lambda^*$ is executed as the longest path. To facilitate the computations, we first define the following events of a $\lambda_h \in \Lambda^*$.

- E_h: an event that path λ_h is executed, *i.e.* every $\theta^k_x \in H(\lambda_h)$ is executed;
- $\overline{E_h}$: an event that path λ_h is not executed, *i.e.*, at least one $\theta^k_x \in H(\lambda_h)$ is not executed;
- E^*_h: an event that λ_h is executed and is the longest path. This indicates (i) the event E_h occurs and (ii) the event $\overline{E_l}$ occurs for every $\lambda_l \in \Lambda^*$ with $l \in [1, h-1]$ (*i.e.*, all the paths before λ_h in Λ^* are not executed).

Let $P(\cdot)$ denote the probability of a given event during the execution of τ, the computation of $P(E^*_h)$ consists of two parts. The first part is calculated as $P(E_h) = \prod_{\theta^k_x \in H(\lambda_h)} F(\theta^k_x)$, *i.e.*, the probability of every branch θ^k_x in $H(\lambda_h)$ is executed. However, it is challenging to obtain the probability of the second part, in which a path λ_l is not executed if any $\theta^k_x \in H(\lambda_l)$ is not executed. Hence, the computation of such a probability can become extremely complex when multiple λ_l are considered, especially when these paths share certain θ^k_x.

To address this, we develop an analysis that estimates $P(E^*_h), \forall \lambda_h \in \Lambda^*$ by leveraging the relationships between the events of $\lambda_h \in \Lambda^*$ (Sec. IV-A). In addition, although deviations may exist in the estimations of $P(E^*_h)$, we demonstrate such deviations would not impose significant pessimism along with the computation of $P(E^*_h)$ for every $\lambda_h \in \Lambda^*$, and prove the correctness of the constructed analysis (Sec. IV-B). Notations introduced in this section are summarised in Tab. III.

A. Computation of $P(E_h^*)$

The computation of $P(E_h^*)$ is established based on the following relationship between $P(E_h^*), \forall \lambda_h \in \Lambda^*$. First, given that Λ^* provides the exact set of longest path candidates of τ (see Theorem 1), it follows that $\sum_{\lambda_h \in \Lambda^*} P(E_h^*) = 1$. In addition, as only one path is the longest path in any $\mathcal{G}_u \in \mathcal{G}$, the probability of both λ_a and λ_b being executed as the longest in one graph is $P(E_a^* \wedge E_b^*) = 0$.

Considering the above, $P(E_h^*)$ can be determined by utilising the relationship between the probabilities of all other longest path candidates in Λ^*, as shown in Eq. (1). The paths in Λ^* are sorted in a non-increasing order by their lengths, in which a smaller index indicates a path with a higher length in general, *i.e.*, $len(\lambda_l) \geq len(\lambda_h) \geq len(\lambda_s)$ with $1 \leq l < h < s \leq |\Lambda^*|$. Notation λ_s (resp. λ_l) denote a path in Λ^* that is shorter (resp. longer) than λ_h.

$$P(E_h^*) = 1 - \sum_{l=1}^{h-1} P(E_l^*) - \sum_{s=h+1}^{|\Lambda^*|} P(E_s^*) \tag{1}$$

Following Eq. (1), Fig. 4 illustrates the computation process for $P(E_h^*)$ of every $\lambda_h \in \Lambda^*$. Starting from the first (*i.e.*, longest) path in Λ^*, we can obtain $P(E_h^*)$ by determining the following two values.

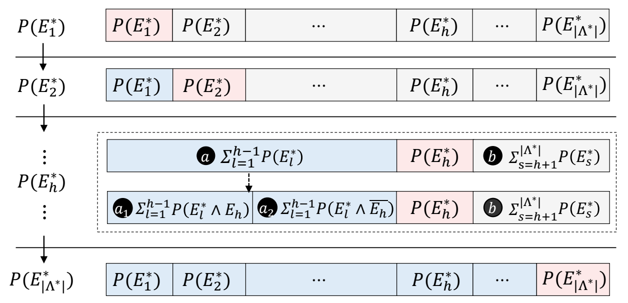
- $\sum_{l=1}^{h-1} P(E_l^*)$: sum of $P(E_l^*)$ with $1 \leq l < h$ (Fig. 4$\textcircled{a}$). This indicates the sum of probability in which a longer candidate path $\lambda_l \in \Lambda^*$ is executed as the longest path;
- $\sum_{s=h+1}^{|\Lambda^*|} P(E_s^*)$: sum of $P(E_s^*)$ in which $h < s \leq |\Lambda^*|$ (Fig. 4$\textcircled{b}$). This indicates the sum of probability where a shorter candidate λ_s is executed as the longest path.

For $\sum_{l=1}^{h-1} P(E_l^*)$, it can be obtained directly because the paths in Λ^* are examined from longest to shortest, with λ_1 being the longest candidate. Hence, when computing $P(E_h^*)$, all the $P(E_l^*)$ with $1 \leq l < h$ have already been calculated in the previous steps, as shown in Fig. 4. As for $\sum_{s=h+1}^{|\Lambda^*|} P(E_s^*)$, it is not determined when $P(E_h^*)$ is under computation, where any $P(E_s^*)$ with $h < s \leq |\Lambda^*|$ is not examined yet.

However, we notice that $\sum_{s=h+1}^{|\Lambda^*|} P(E_s^*)$ is involved in the computation of the probability where λ_h is not executed (*i.e.*, $P(\overline{E_h})$), as shown in Eq. (2). Among all the events in Fig. 4, λ_h is not executed in the events $E_s^*, h < s \leq |\Lambda^*|$ as $len(\lambda_h) \geq len(\lambda_s)$ (Fig. 4$\textcircled{b}$). As for $E_l^*, 1 \leq l < h$ in Fig. 4$\textcircled{a}$, it can be further divided into two cases: (i) a λ_l is executed as the longest path while λ_h is also executed ($E_l^* \wedge E_h$ in Fig. 4$\textcircled{a_1}$), and (ii) a λ_l is executed as the longest path but λ_h is not executed ($E_l^* \wedge \overline{E_h}$ in Fig. 4$\textcircled{a_2}$). Based on the above, $P(\overline{E_h})$ can be computed by the probability of $E_l^* \wedge \overline{E_h}$ with $1 \leq l < h$ (Fig. 4$\textcircled{a_2}$) and E_s^* with $h < s < |\Lambda^*|$ (Fig. 4$\textcircled{b}$).

$$P(\overline{E_h}) = \sum_{l=1}^{h-1} P(E_l^* \wedge \overline{E_h}) + \sum_{s=h+1}^{|\Lambda^*|} P(E_s^*) \tag{2}$$

In addition, given that the probability of λ_h being executed is $\prod_{\theta_x^k \in H(\lambda_h)} F(\theta_x^k)$ (*i.e.*, the probability where every branch



$P(E_1^*)$
$P(E_1^*)$ | $P(E_2^*)$ | ··· | $P(E_h^*)$ | ··· | $P(E_{|\Lambda^*|}^*)$
$P(E_2^*)$
$P(E_1^*)$ | $P(E_2^*)$ | ··· | $P(E_h^*)$ | ··· | $P(E_{|\Lambda^*|}^*)$
$P(E_h^*)$
$\textcircled{a}$ $\Sigma_{l=1}^{h-1} P(E_l^*)$ | $P(E_h^*)$ | $\textcircled{b}$ $\Sigma_{s=h+1}^{|\Lambda^*|} P(E_s^*)$
$\textcircled{a_1}$ $\Sigma_{l=1}^{h-1} P(E_l^* \wedge E_h)$ | $\textcircled{a_2}$ $\Sigma_{l=1}^{h-1} P(E_l^* \wedge \overline{E_h})$ | $P(E_h^*)$ | $\textcircled{b}$ $\Sigma_{s=h+1}^{|\Lambda^*|} P(E_s^*)$
$P(E_{|\Lambda^*|}^*)$
$P(E_1^*)$ | $P(E_2^*)$ | ··· | $P(E_h^*)$ | ··· | $P(E_{|\Lambda^*|}^*)$

Fig. 4: Computation of $P(E_h^*)$ for every $\lambda_h \in \Lambda^*$ ordered in non-increasing path length *(blue: paths that are examined; red: the path under computation; grey: paths to be examined).*

θ_x^k in $H(\lambda_h)$ is executed), the $P(\overline{E_h})$ can also be obtained based on Eq. (3).

$$P(\overline{E_h}) = 1 - \prod_{\theta_x^k \in H(\lambda_h)} F(\theta_x^k) \tag{3}$$

Combining both Eq. (2) and (3) for $P(\overline{E_h})$, we have $\sum_{l=1}^{h-1} P(E_l^* \wedge \overline{E_h}) + \sum_{s=h+1}^{|\Lambda^*|} P(E_s^*) = 1 - \prod_{\theta_x^k \in H(\lambda_h)} F(\theta_x^k)$. Accordingly, $\sum_{s=h+1}^{|\Lambda^*|} P(E_s^*)$ can be computed as Eq. (4).

$$\sum_{s=h+1}^{|\Lambda^*|} P(E_s^*) = 1 - \prod_{\theta_x^k \in H(\lambda_h)} F(\theta_x^k) - \sum_{l=1}^{h-1} P(E_l^* \wedge \overline{E_h}) \tag{4}$$

Based on the above, $\sum_{s=h+1}^{|\Lambda^*|} P(E_s^*)$ can be obtained if $\sum_{l=1}^{h-1} P(E_l^* \wedge \overline{E_h})$ is computed. However, it is still not straightforward to compute $P(E_l^* \wedge \overline{E_h})$, which implies that all candidates longer than λ_l are not executed. To bound $P(E_l^* \wedge \overline{E_h})$, we simplify the computation to only consider the case where λ_l is executed while λ_h is not, as shown in Eq. (5). As all branches in $H(\lambda_l)$ must be executed when λ_l is executed, the branches $\theta_x^k \in H(\lambda_l) \cap H(\lambda_h)$ are not included in the computation of $P(\overline{E_h})$.

$$P(E_l^* \wedge \overline{E_h}) \leq \prod_{\theta_x^k \in H(\lambda_l)} F(\theta_x^k) \times \Big(1 - \prod_{\theta_x^k \in H(\lambda_h) \setminus H(\lambda_l)} F(\theta_x^k)\Big) \tag{5}$$

To this end, $P(E_l^* \wedge \overline{E_h})$, $\sum_{s=h+1}^{|\Lambda^*|} P(E_s^*)$, and eventually, the $P(E_h^*)$ can be obtained by Eq. (5), (4), and (1), respectively. The analysis starts the longest candidate in Λ^* and produces the $P(E_h^*), \forall \lambda_h \in \Lambda^*$ with $|\Lambda^*|$ iterations. For the p-DAG in Fig. 1(a), Example 3 illustrates the computation of $P(E_h^*)$ of every $\lambda_h \in \Lambda^*$ obtain in Example 2.

Example 3. First, for the $\lambda_1 = \{v_1, v_2, v_5, v_9, ...\}$ obtained in Example 2, $P(E_1^*)$ is calculated directly as 0.3. Then, for the $\lambda_2 = \{v_1, v_2, v_6, v_9, ...\}$, $P(E_2^*)$ is computed as $P(E_2^*) = 1 - P(E_1^*) - P(E_3^*)$ based on Eq. (1). Similarly, the $P(E_3^*)$ for $\lambda_3 = \{v_1, v_4, v_8, v_{10}, ...\}$ can be calculated by $P(E_3^*) = 1 - \prod_{\theta_x^k \in H(\lambda_2)} F(\theta_x^k) - P(E_1^* \wedge \overline{E_2})$ based on Eq. (4), where $P(E_1^* \wedge \overline{E_2})$ is estimated as $0.3 \times (1 - 0.6) = 0.12$ based on Eq. (5). Therefore, we have $P(E_1^*) = 0.3$, $P(E_2^*) = 1 - 0.3 - (1 - 0.6 - 0.12) = 0.42$, and $P(E_3^*) = 0.28$ as the results.

Finally, it is worth noting that as fewer paths are considered when computing $P(E_l^* \wedge \overline{E_h})$ in Eq. (5), an upper bound is provided for $P(E_l^* \wedge \overline{E_h})$ according to the *inclusion-exclusion principle*, instead of the exact value. This leads to lower estimations of $\sum_{s=h+1}^{|\Lambda^*|} P(E_s^*)$ in Eq. (4). Hence, deviations can exist in $P(E_h^*)$ as it depends on both $\sum_{l=l}^{h-1} P(E_l^*)$ and $\sum_{s=h+1}^{|\Lambda^*|} P(E_s^*)$. As a result, the probabilities of candidates of longer paths in Λ^* could be overestimated, and subsequently, lead to a lower probability for the shorter ones. Considering such deviations, the following constraints are enforced on $P(E_h^*)$ of every $\lambda_h \in \Lambda^*$ for the correctness of the analysis.

- $P(E_h^*) \geq 0, \forall \lambda_h \in \Lambda^*$. This guarantees no negative values are produced for the estimations of $P(E_h^*)$;
- $\sum_{l=1}^{h-1} P(E_l^*) + P(E_h^*) \leq 1$. If $\sum_{l=1}^{h-1} P(E_l^*) + P(E_h^*) > 1$ when analysing a $\lambda_h \in \Lambda^*$, the computation terminates directly with $P(E_h^*) = 1 - \sum_{l=1}^{h-1} P(E_l^*)$ and $P(E_s^*) = 0,\ h < s \leq |\Lambda^*|$. This guarantees $\sum_{\lambda_h \in \Lambda^*} P(E_h^*) = 1$.

B. Discussions of Deviation and Correctness

As described above, deviations can exist in the estimation of $P(E_l^* \wedge \overline{E_h})$, and hence, the computation of $P(E_h^*)$. The deviation indicates the difference between $P(E_h^*)$ computed by Eq. (1) and the actual value obtained by the enumeration-based approach. As described above, such deviations can occur as the computation of $P(E_l^* \wedge \overline{E_h})$ is simplified in Eq. (5) to only consider the case where λ_l is executed while λ_h is not. In fact, if λ_l is executed as the longest path, all the candidate paths prior to λ_l are not executed. For instance, given $\Lambda^* = \{\lambda_1, \lambda_2, \lambda_3\}$, $P(E_2^* \wedge \overline{E_3}) = P(E_2 \wedge \overline{E_1} \wedge \overline{E_3}) \leq P(E_2 \wedge \overline{E_3})$ (*i.e.*, the estimated value in Eq. (5)). This would overestimate $P(E_2^*)$ while underestimating $P(E_3^*)$ based on Eq. (1).

In this section, we first demonstrate that the deviation of one $P(E_h^*)$ would not propagate along with the computation of every $\lambda_h \in \Lambda^*$, and then prove the correctness of the constructed analysis. Based on the analysis constructed in Sec. IV-A, the deviations of $P(E_h^*)$ of the currently-examined λ_h (denoted as δ_h) can arise from the following two aspects.

- δ_h^{dir}: the *direct deviation* from $P(E_l^* \wedge \overline{E_h})$ in Eq. (5);
- δ_h^{ind}: the *indirect deviation* from the deviations of $P(E_l^*), 1 \leq l < h$ in Eq. (1).

For the δ_h^{dir}, it is not caused by deviations of any $P(E_l^*), 1 \leq l < h$ in the previous computations. Below we focus on δ_h^{ind} to illustrate the propagation of deviations. Eq. (6) and (7) shows the computations of δ_h and δ_h^{ind}, respectively. For a given λ_h, its δ_h is determined by the subtraction of δ_h^{dir} and δ_h^{ind} according to the computation of $P(E_h^*)$ based on $P(E_l^* \wedge \overline{E_h})$ and $\sum_{s=h+1}^{|\Lambda^*|} P(E_s^*)$ (see Sec. IV-A).

$$\delta_h = \delta_h^{dir} - \delta_h^{ind} \tag{6}$$

$$\delta_h^{ind} = \sum_{l=1}^{h-1} \delta_l \tag{7}$$

With Eq. (6) and (7), δ_h^{ind} can be computed by applying both equations recursively, as shown in Eq. (8). First, δ_h^{ind} is computed as $\sum_{l=1}^{h-1} \delta_l^{dir} - \sum_{l=1}^{h-1} \delta_l^{ind}$ by applying Eq. (6) and (7), in which the latter term $\sum_{l=1}^{h-1} \delta_l^{ind}$ can be split into $\sum_{l=1}^{h-2} \delta_l^{ind} + \delta_{h-1}^{ind}$. Then, δ_{h-1}^{ind} is further computed as $\sum_{l=1}^{h-2} \delta_l = \sum_{l=1}^{h-2} \delta_l^{dir} - \sum_{l=1}^{h-2} \delta_l^{ind}$. To this end, δ_h^{ind} is computed as $\delta_h^{ind} = \sum_{l=1}^{h-1} \delta_l^{dir} - \sum_{l=1}^{h-2} \delta_l^{dir}$, which is simplified to $\delta_h^{ind} = \delta_{h-1}^{dir}$.

$$\begin{aligned}
\delta_h^{ind} &= \sum_{l=1}^{h-1} \delta_l = \sum_{l=1}^{h-1} \delta_l^{dir} - \sum_{l=1}^{h-1} \delta_l^{ind} \\
&= \sum_{l=1}^{h-1} \delta_l^{dir} - (\sum_{l=1}^{h-2} \delta_l^{ind} + \delta_{h-1}^{ind}) \\
&= \sum_{l=1}^{h-1} \delta_l^{dir} - (\sum_{l=1}^{h-2} \delta_l^{ind} + \sum_{l=1}^{h-2} \delta_l) \\
&= \sum_{l=1}^{h-1} \delta_l^{dir} - \Big(\sum_{l=1}^{h-2} \delta_l^{ind} + (\sum_{l=1}^{h-2} \delta_l^{dir} - \sum_{l=1}^{h-2} \delta_l^{ind})\Big) \\
&= \sum_{l=1}^{h-1} \delta_l^{dir} - \sum_{l=1}^{h-2} \delta_l^{dir} = \delta_{h-1}^{dir}
\end{aligned} \tag{8}$$

With δ_h^{dir} obtained, $\delta_h = \delta_h^{dir} - \delta_h^{ind} = \delta_h^{dir} - \delta_{h-1}^{dir}$ based on Eq. (6). From the computations, it can be observed that $P(E_h^*)$ is only affected by the deviation from the computations for λ_h and λ_{h-1}. This demonstrates the analysis effectively manages the pessimism by preventing the propagation of deviations.

In addition, such deviations would not undermine the correctness of the analysis. Let $\mathbb{P}(E_h^*) = \sum_{1 \leq l < h} P(E_l^*) + P(E_h^*)$ denote the cumulative probability where the length of the longest path being executed is equal to or higher than $len(\lambda_h)$. Theorem 2 shows that the constructed probabilistic analysis provides an upper bound on $\mathbb{P}(E_h^*)$ for every $\lambda_h \in \Lambda^*$.

Theorem 2. *Let $\mathbb{P}^{\S}(E_h^*)$ denote the exact probability of $\mathbb{P}(E_h^*)$ obtained by the enumeration-based method, it follows that $\mathbb{P}(E_h^*) \geq \mathbb{P}^{\S}(E_h^*), \forall \lambda_h \in \Lambda^*$.*

Proof. Based on Eq. (6) and (8), $P(E_h^*) = P^{\S}(E_h^*) + \delta_h^{dir} - \delta_{h-1}^{dir}$ with derivations considered. In addition, given that $\mathbb{P}(E_h^*) = \sum_{l=1}^{h} P(E_l^*)$, in which $P(E_l^*)$ is further computed as $P^{\S}(E_l^*) + \delta_l^{dir} - \delta_{l-1}^{dir}$. Therefore, $\mathbb{P}(E_h^*)$ and $\mathbb{P}^{\S}(E_h^*)$ follows that $\mathbb{P}(E_h^*) = \mathbb{P}^{\S}(E_h^*) + \delta_h^{dir}$. As δ_h^{dir} is a non-negative value due to the *inclusion-exclusion* principle (see Eq. (5)), $\mathbb{P}(E_h^*) \geq \mathbb{P}^{\S}(E_h^*)$ holds for all $\lambda_h \in \Lambda^*$. □

V. Probabilistic Timing Distribution of p-DAGs

With $P(E_h^*), \forall \lambda_h \in \Lambda^*$ obtained, this section first presents the complete probabilistic response-time analysis for a single periodic p-DAG (Sec.V-A). Then, we extend the analysis to support systems with multiple p-DAGs (Sec.V-B). Notations introduced in this section are summarised in Tab. IV.

A. Probabilistic Timing Analysis of a p-DAG

We first present the computations of the probabilistic response time distribution of a single p-DAG. A p-DAG τ can demonstrate various response time when a different candidate in Λ^* is executed as the longest path. Following the classic

TABLE IV: Notations introduced in Sec. V.

Notation	Description
R_h	The worst-case response time of τ when λ_h is executed as the longest path.
$R_h^\S$	The ground truth value of R_h.
$\mathbb{P}(R_h)$	The cumulative probability where the response time of τ is equal to or higher than R_h.
$\mathbb{P}^\S(R_h)$	The ground-truth value of $\mathbb{P}(R_h)$.
$R_h^\dagger$	The worst-case response time of τ when λ_h is executed as the longest path in a system with under multiple p-DAGs.
$hp(\tau)$	The set of p-DAGs that have a higher priority than τ.

Graham's bound [8] described in Sec. II, for a $\lambda_h \in \Lambda^*$ being executed as the longest path, the worst-case response time (denoted as R_h) under any work conserving schedule is computed by Eq. (9). The resulting value (*i.e.*, the right-hand side of the equation) represents the upper bound on response time. Notation m is the number of cores and $vol(\tau) - len(\lambda_h)$ gives the worst-case interfering workload of λ_h.

$$R_h \leq len(\lambda_h) + \frac{1}{m} \times (vol(\tau) - len(\lambda_h)) \tag{9}$$

The $vol(\tau)$ is the bounded by Eq. (10), which consists of (i) the workload from the non-conditional nodes that are executed in any scenario of τ (*i.e.*, $v_i \notin \Theta$); and (ii) the worst-case workload of each probabilistic structure of τ (*i.e.*, $\forall \theta_x \in \Theta$), where the branch with the maximum workload of each probabilistic structure is taken into account. In addition, although the deviations can occur during the computation of $P(E_h^*)$ (see Sec. IV-B), the resulting probabilistic response time distribution is safe. This is because for any $\lambda_a, \lambda_b \in \Lambda^*$ with $len(\lambda_a) \geq len(\lambda_b)$, the corresponding response times satisfy $R_a \geq R_b$ since $vol(\tau)$ is fixed given by Eq. (10). Therefore, even if $P(E_a^*)$ is overestimated and $P(E_b^*)$ is underestimated, the analysis still bounds the response time due to the conservative ordering of $R_a \geq R_b$.

$$vol(\tau) = \sum_{v_i \notin \Theta} C_i + \sum_{\theta_x \in \Theta} max\{\sum_{v_i \in \theta_x^k} C_i \mid \theta_x^k \in \theta_x\} \tag{10}$$

We note that finer computations on the interfering workload is possible. However, tightening the interference workload of a $\lambda_h \in \Lambda^*$ is non-trivial, which requires identifying different interfering scenarios with various probabilities. Hence, we decide to halt here to avoid over-complicated computations. The correctness of the analysis is supported by Graham's bound [8], and is applied in [6] for p-DAGs. Essentially, the analysis assumes a worst-case execution model where the parallel workload gets no more than its fair share of processor time, *i.e.*, the total response time will not exceed the sum of the longest path and the equally distributed remaining workload across processors. A complete proof can be found in [8].

Combing with the $P(E_h^*)$ computed in Sec. IV, the probabilistic response time distribution of τ can be obtained based on $\mathbb{P}(R_h) = \sum_{1 \leq l < h} P(E_l^*) + P(E_h^*)$ of the $R_h, \forall \lambda_h \in \Lambda^*$, in which $\mathbb{P}(R_h)$ indicates the probability where the response time of τ is equal to or higher than R_h. Example 4 presents the complete probabilistic response time distribution of the p-DAG in Fig. 1(a) on a dual-core system (*i.e.*, $m = 2$).

Example 4. First, given the $\Lambda^* = \{\lambda_1, \lambda_2, \lambda_3\}$ identified in Example 2 and $vol(\tau) = 33$ with $m = 2$, we have $R_1 = 26.5$, $R_2 = 24.5$, and $R_3 = 24$. Second, as $P(E_1^*) = 0.3$, $P(E_2^*) = 0.42$, $P(E_3^*) = 0.28$ computed in Example 3, we have $\mathbb{P}(R_1) = 0.3$, $\mathbb{P}(R_2) = 0.72$, and $\mathbb{P}(R_3) = 1.0$ for the p-DAG in Fig. 1(a).

Theorem 3 justifies the correctness of the complete probabilistic timing analysis for p-DAGs constructed in this work.

Theorem 3. *Let $R_h^\S$ and $\mathbb{P}^\S(R_h)$ denote the exact values of R_h and $\mathbb{P}(R_h)$, it follows $R_h \geq R_h^\S \wedge \mathbb{P}(R_h) \geq \mathbb{P}^\S(R_h), \forall \lambda_h \in \Lambda^*$.*

Proof. We first prove that $R_h \geq R_h^\S$ for a given $\lambda_h \in \Lambda^*$. Based on Eq. (10), $vol(\tau)$ is computed by the branch with the maximum workload in each $\theta_x \in \Theta$. As only one $\theta_x^k \in \theta_x$ is executed, this bounds the worst-case volume of all possible scenarios; and hence, $R_h \geq R_h^\S$ follows. As for $\mathbb{P}(R_h) \geq \mathbb{P}^\S(R_h)$ for a given R_h, this is proved in Theorem 2 where $\mathbb{P}(R_h) = \mathbb{P}(\lambda_h) \geq \mathbb{P}^\S(\lambda_h) = \mathbb{P}^\S(R_h)$. Therefore, this theorem holds for all $\lambda_h \in \Lambda^*$. □

This concludes the constructed timing analysis for a p-DAG. By exploiting the set of the longest path candidates in τ (*i.e.*, Λ^*), the analysis produces a probabilistic response time distribution of τ (*e.g.*, the one in Fig. 1(b)) without the need for enumerating through every $\mathcal{G}_u \in \mathcal{G}$. In addition, we acknowledge that the proposed analysis is not exact (as depicted in Fig. 2), where pessimism can exist in (i) the estimation of $P(E_h^*), \lambda_h \in \Lambda^*$ and (ii) the computation of the interference workload of $\lambda_h \in \Lambda^*$. However, as shown in Sec. VI, such pessimism does not affect the effectiveness of the proposed analysis, which can effectively scale to large p-DAG while not significantly sacrificing the accuracy. With a specified time limit defined for τ, the analysis can provide the probability where the system misses its deadline, offering valuable insights that effectively empower optimised system design solutions, as shown by Tab. V in Sec. VI.

B. Probabilistic Timing Analysis for Multiple p-DAGs

For systems comprising multiple periodic p-DAGs scheduled under a work-conserving fixed-priority scheme, Eq. (9) can be applied through minor modifications to enable the analysis of p-DAGs within such systems. This section first presents the analysis under a simplified setting, where all p-DAGs in the system are released synchronously with harmonic periods and constrained deadlines. Specifically, for any two tasks, the longer period is an integer multiple of the shorter one, and each relative deadline does not exceed its corresponding period. We then relax this assumption to extend the analysis to support more general system configurations.

For systems in which all the p-DAGs are released synchronously with harmonic period and constrained deadlines, Eq. (11) presents the worst-case response time of τ when λ_h is executed as the longest path under multiple p-DAGs (denoted as $R_h^\dagger$), in which $\tau_w \in hp(\tau)$ is a p-DAG with a higher priority than τ. In such a system, the execution of λ_h in τ can incur delays from (i) the interfering workload from nodes in τ (*i.e.*, $vol(\tau) - len(\lambda_h)$); and (ii) the interference workload from the high-priority p-DAGs (*i.e.*, $\sum_{\tau_w \in hp(\tau)} \lceil \frac{R_h^\dagger}{T_{\tau_w}} \rceil vol(\tau_w)$). In addition, as all tasks are released in a synchronous fashion with harmonic periods, no carry-in job can occur if the system is schedulable [24]. Therefore, similar to the analysis constructed in [1] (more specifically, Eq. (14)), the notation $\lceil \frac{R_h^\dagger}{T_{\tau_w}} \rceil$ effectively bounds the maximum number of releases that an interfering high-priority task $\tau_w \in hp(\tau)$ can issue during the release of the currently-examined task τ.

$$R_h^\dagger \leq len(\lambda_h) + \frac{1}{m} \times \Big(vol(\tau) - len(\lambda_h) \Big) + \frac{1}{m} \times \left(\sum_{\tau_w \in hp(\tau)} \left\lceil \frac{R_h^\dagger}{T_{\tau_w}} \right\rceil vol(\tau_w) \right) \tag{11}$$

In addition, the interference imposed from both the intra-task nodes (*i.e.*, nodes in τ) and higher-priority p-DAGs can be safely bounded by distributing the interfering workload across the m processors. The rationale is that under any work-conserving scheduler, a node on the longest path of τ can only experience interference if all m processors are simultaneously busy. At such a point in time, the processors must either be executing other nodes of the same p-DAG or executing nodes belonging to higher-priority p-DAGs. This is proved by the Lemma 4.2 in [23]. Therefore, the total interfering workload can be amortized over m processors without underestimating the interference experienced by τ.

To compute the worst-case response time $R_h^\dagger$, the analysis is framed as solving a fixed-point equation. Specifically, the process starts by initializing $R_h^\dagger$ to the response time obtained when only intra-task interference is considered, as given in Eq. (9). The value of $R_h^\dagger$ is then iteratively updated using Eq. (11), until convergence is reached or the computed value exceeds the task's deadline, in which case the task is deemed unschedulable. The formal proof for computing the response time via fixed-point iteration has been well established in prior work, including Theorem 24 in [4] and Theorem 5 in [21].

Finally, the analysis constructed above can be further extended to support p-DAGs with different periods and constrained deadlines. In such cases, when analysing a set of periodic tasks, some jobs may be released before τ but have execution segments that overlap with the busy window of τ [25], thereby partially contributing to the interference. To account for this effect, an additional carry-in job of each high-priority p-DAG is included in the interference of τ to provide the upper bound, as shown in Eq. (12). As stated in [25], for tasks with constrained deadlines, at most one carry-in job of every high-priority task can interfere with the execution of τ.

$$R_h^\dagger \leq len(\lambda_h) + \frac{1}{m} \times \Big(vol(\tau) - len(\lambda_h) \Big) + \frac{1}{m} \times \left(\sum_{\tau_w \in hp(\tau)} \Big(vol(\tau_w) + \left\lceil \frac{R_h^\dagger}{T_{\tau_w}} \right\rceil vol(\tau_w) \Big) \right) \tag{12}$$

VI. Evaluation

This section evaluates the proposed analysis against the existing approaches [6], [8] in terms of the pessimism of analytical results (Sec. VI-B), the computation cost (Sec. VI-C) and the impact on the system design with p-DAGs (Sec. VI-D).

A. Experimental Setup

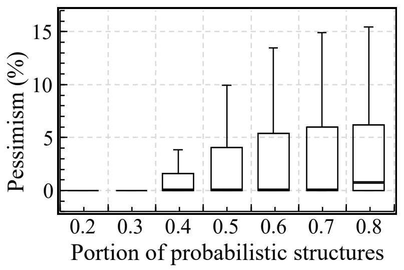
The experiments are conducted on a set of randomly generated p-DAGs with four symmetric cores (*i.e.*, $m = 4$). Specifically, the generation of a p-DAG starts by constructing the DAG structure. As with [1], [3], [9], the number of layers is randomly chosen in $[5, 8]$ and the number of nodes in each layer is decided in $[2, p]$ (with $p = 6$ by default). Each node has a 20% likelihood of being connected to a node in the previous layer. As with [26], the period T is randomly generated in $[1, 1400]$ units of time with $D = T$. The workload is calculated by $T \times 50\%$, given a total utilisation of 50%. The WCET of each node is uniformly generated based on the total workload of the p-DAG task.

Then, similar with [6], a number of probabilistic structures (*i.e.*, $|\Theta|$) are generated by replacing the non-conditional nodes in the generated DAG, where $|\Theta|$ ranges from 2 to 10 with $|\Theta| = 3$ by default. Each $\theta_x \in \Theta$ contains three probabilistic branches. By this setting, a p-DAG under evaluation contains $3^3 = 27$ scenarios with $|\Theta| = 3$, and can scale up to $3^{10} = 59,049$ scenarios when $|\Theta| = 10$. A $\theta_x^k \in \theta_x$ is a non-conditional sub-graph generated using the same approach, with the number of layers and nodes in each layer determined in $[2, 4]$. A parameter *probabilistic structure ratio* (psr) is used to control the volume of the probabilistic structures in τ, *e.g.*, $psr = 0.4$ means the workload of probabilistic structures is 40% of the total workload of τ. The $F(\theta_x^k) \in [0, 1]$ of each θ_x^k is assigned with a randomly probability, with $\sum_{\theta_x^k \in \theta_x} F(\theta_x^k) = 1$ enforced for all θ_x^k in every θ_x. All parameters within a specific range are randomly generated from a uniform distribution. For each system setting, 500 p-DAGs are generated to evaluate the competing methods.

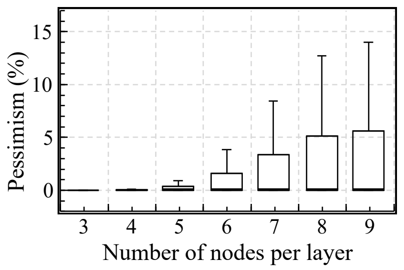
The *Non-Overlapping Area Ratio* (NOAR) [27] is applied to compare the probabilistic distributions produced by the proposed analysis and the enumeration-based analysis applied in [6] analysis. The NOAR is computed as the *non-overlapping area* between the two distributions divided by the area of the distribution obtained by the enumeration-based approach. The *area* of a probabilistic distribution is quantified as the space enclosed by its distribution curve and the x-axis from the lowest to the highest response time. The *non-overlapping area* of two distributions is the space covered exclusively by either distribution. A lower value of NOAR indicates less pessimism in the timing distribution derived from the proposed analysis.

B. Evaluating the Pessimism of the Proposed Analysis

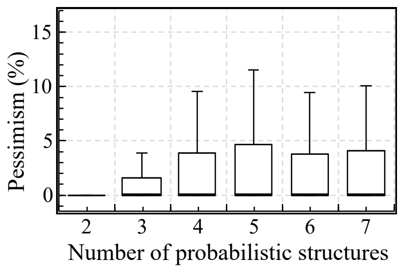
This section evaluates the pessimism of the proposed analysis by comparing it with the enumeration-based analysis, demonstrating that the deviations in the proposed analysis would not impose a significant pessimism on the analytical results. Fig. 5 presents the deviations between the proposed



(a) $p = 6$, $|\Theta| = 3$ and varied psr.



(b) $psr = 0.4$, $|\Theta| = 3$ and varied p.



(c) $psr = 0.4$, $p = 6$ and varied $|\Theta|$.

Fig. 5: The pessimism of the proposed analysis compared to the enumeration-based analysis under varied psr, p and $|\Theta|$.

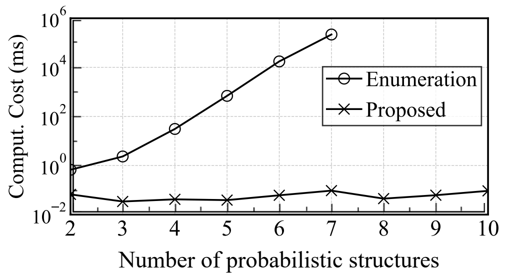


Fig. 6: Comparison of the computation cost under varied $|\Theta|$ *(the y-axis is in log-scale).*

analysis and the enumeration-based approach in terms of NOAR, with p-DAGs generated under varied psr (Fig. 5(a)), p (Fig. 5(b)) and $|\Theta|$ (Fig. 5(c)), respectively.

Obs 1. The proposed analysis achieves an average deviation of only 1.04% compared to the enumeration-based analysis, and the deviations remain below 5% in most cases.

This observation is obtained from Fig. 5, in which the deviation between our and the enumeration-based analysis is 1.45%, 0.73% and 0.71% on average with varied psr, p and $|\Theta|$, respectively. Notably, our method shows negligible deviations (0.23% on average) for p-DAGs with a relatively simple structure, *e.g.*, $psr \leq 0.4$, $p \leq 6$ or $|\Theta| \leq 3$. In such cases, the number of longest path candidates identified is relatively low, hence, leading to low deviations from Eq. (5) when computing $P(E_h^*)$ of $\lambda_h \in \Lambda^*$ (see Sec. IV-A).

However, as the structure of p-DAGs becomes more complex, a slight increase in the deviation is observed between the two analysis, *e.g.*, 3.01% and 1.81% on average when $psr = 0.7$ in Fig. 5(a) and $p = 8$ in Fig. 5(b), respectively. For large and complex p-DAGs with a high $|\Lambda^*|$, deviations can exist in multiple $P(E_h^*), \forall \lambda_h \in \Lambda^*$ values, and thereby leading to an increased NOAR between two analysis. However, as observed, the deviations are below 5% for most cases across all experiments. This supports the claim in Sec. IV-B that the proposed analysis effectively manages the propagation of deviations during the computation of each $\lambda_h \in \Lambda^*$, justifying the effectiveness of the analysis.

C. Comparison of the Computation Costs

Fig. 6 shows the computation cost (in milliseconds) of the proposed analysis and the enumeration-based approach, under p-DAGs generated under varied $|\Theta|$. For each value of $|\Theta|$, 500 p-DAGs are generated and evaluated, and the average computation cost is reported in the figure. In addition, it reveals a possible stability issue if the implementation of the enumeration-based approach is not carefully optimised. The results are measured on a desktop with an Intel i5-13400 processor running at 2.50GHz and a memory of 24GB.

Obs 2. The proposed analysis significantly reduces the computation cost compared to the enumeration-based approach and demonstrates high efficiency for large p-DAGs.

As observed in Fig. 6, the computation cost of the enumeration-based approach grows exponentially as $|\Theta|$ increases. In addition, we observe that this analysis fails to provide any results when $|\Theta| > 7$, which encounters an out-of-memory error on the experimental machine. This is because the enumeration-based analysis requires exhaustive exploration of all execution scenarios by recursively traversing the p-DAG, where each scenario contains extensive information describing both the structural and temporal characteristics. Thus, the enumeration-based approach demands substantial memory, particularly for large p-DAGs, *e.g.*, when $|\Theta| \geq 8$ with over 6,561 scenarios. Although various optimisations can be employed to reduce the footprint of each execution scenario, this can involve non-trivial efforts that can be fully avoided with the proposed analysis. By eliminating the need for enumeration, our analysis achieves significantly lower computation costs and memory usages, *i.e.*, lower than 10 milliseconds and 320 MB across all the evaluated p-DAGs, including ones with $|\Theta| > 7$.

Combining Obs.1 (Sec. VI-B), the proposed analysis maintains a low pessimism for relatively small p-DAGs while demonstrating high efficiency for large p-DAGs, providing an effective and efficient analysis solution for analysing p-DAGs.

D. Impact on System Design Solutions

This section compares the design solutions produced by the proposed and existing analyses. Tab. V shows the average number of cores required to achieve a given acceptance ratio, decided by the competing methods, *e.g.*, "Enumeration-80%" means that the enumeration-based analysis is applied with an acceptance ratio of 80%.

Obs 3. The proposed analysis empowers effective system design solutions and demonstrates negligible differences in

TABLE V: Comparison of the average number of cores required under different acceptance ratios and varied $|\Theta|$.

$\|\Theta\|$	3	4	5	6	7	8	9
Enumeration-70%	8.66	8.18	8.07	7.88	7.76	-	-
Proposed-70%	8.66	8.19	8.09	7.89	7.78	7.20	7.07
Enumeration-80%	12.78	11.31	10.67	9.93	9.94	-	-
Proposed-80%	12.78	11.31	10.67	9.93	9.94	9.05	8.30
Enumeration-90%	12.78	11.32	10.70	10.01	10.11	-	-
Proposed-90%	12.78	11.32	10.70	10.01	10.11	9.33	8.63
Enumeration-100%	13.67	12.38	12.07	11.96	12.22	-	-
Proposed-100%	13.67	12.38	12.07	11.96	12.22	12.16	12.10
Graham-100%	13.67	12.38	12.07	11.96	12.22	12.16	12.10

reducing the number of cores required compared to the enumeration-based analysis.

As shown in the table, both probabilistic analysis outperform Graham's bound in a general case by leveraging the probabilistic timing behaviours of p-DAGs. For instance, with the acceptance ratio of 70%, our analysis reduces the number of cores by 36.33% compared to Graham's bound when $|\Theta|= 7$. More importantly, negligible differences are observed between our analysis and the enumeration-based approach for $|\Theta|\leq 7$, whereas our analysis remains effective as $|\Theta|$ continues to increase. This justifies the effectiveness of the constructed analysis and its benefits in improving design solutions by exploiting probabilistic timing behaviours of p-DAGs. In addition, we observe that fewer cores are needed as $|\Theta|$ increases. This is expected as the construction of the probabilistic structures may increase task parallelism with a pre-defined workload (as described in the experimental setup in Sec. VI-A), which would lead to p-DAGs that are more likely to be schedulable within a given limit.

VII. Conclusion

This paper presents a probabilistic response time analysis for a p-DAG by exploiting its longest path candidates. We first identify the complete set of longest path candidates of the p-DAG, and calculate the probability of each being executed as the longest path. Then, the worst-case interfering workload of each candidate is computed to produce the complete probabilistic response time distribution. Experiments show that compared to existing approaches, our analysis significantly enhances efficiency for large p-DAG while maintaining low pessimism. The constructed analysis provides an effective analytical solution for systems with p-DAGs, empowering optimised system design that fully leverages the probabilistic timing behaviours.

In future work, we will focus on tightening the bounds on the interference that each longest path candidate can incur by taking the various interference scenarios and their execution probabilities into account. This will enable a more accurate characterisation of timing behaviour under the execution uncertainties of p-DAGs. In addition, a new analysis for systems with multiple p-DAGs will be constructed by exploiting the execution uncertainties of each interfering p-DAG task, instead of applying a single conservative bound.

References

[1] S. Zhao, X. Dai, I. Bate, A. Burns, and W. Chang, "DAG scheduling and analysis on multiprocessor systems: Exploitation of parallelism and dependency," in *2020 IEEE Real-Time Systems Symposium (RTSS)*. IEEE, 2020, pp. 128–140.

[2] S. Zhao, X. Dai, B. Lesage, and I. Bate, "Cache-aware allocation of parallel jobs on multi-cores based on learned recency," in *Proceedings of the 31st International Conference on Real-Time Networks and Systems*, 2023, pp. 177–187.

[3] Q. He, N. Guan, Z. Guo *et al.*, "Intra-task priority assignment in real-time scheduling of DAG tasks on multi-cores," *IEEE Transactions on Parallel and Distributed Systems*, vol. 30, no. 10, pp. 2283–2295, 2019.

[4] Q. He, M. Lv, and N. Guan, "Response time bounds for DAG tasks with arbitrary intra-task priority assignment," in *33rd Euromicro Conference on Real-Time Systems (ECRTS 2021)*, 2021.

[5] S. Baruah, "Feasibility analysis of conditional DAG tasks is co-npnp-hard," in *Proceedings of the 29th International Conference on Real-Time Networks and Systems*, 2021, pp. 165–172.

[6] N. Ueter, M. Günzel, and J.-J. Chen, "Response-time analysis and optimization for probabilistic conditional parallel DAG tasks," in *2021 IEEE Real-Time Systems Symposium (RTSS)*. IEEE, 2021, pp. 380–392.

[7] A. Melani, M. Bertogna, V. Bonifaci, A. Marchetti-Spaccamela, and G. C. Buttazzo, "Response-time analysis of conditional DAG tasks in multiprocessor systems," in *2015 27th Euromicro Conference on Real-Time Systems*. IEEE, 2015, pp. 211–221.

[8] R. L. Graham, "Bounds on multiprocessing timing anomalies," *SIAM journal on Applied Mathematics*, vol. 17, no. 2, pp. 416–429, 1969.

[9] S. Zhao, X. Dai, and I. Bate, "DAG scheduling and analysis on multi-core systems by modelling parallelism and dependency," *IEEE transactions on parallel and distributed systems*, vol. 33, no. 12, pp. 4019–4038, 2022.

[10] R. I. Davis and L. Cucu-Grosjean, "A survey of probabilistic timing analysis techniques for real-time systems," *LITES: Leibniz Transactions on Embedded Systems*, pp. 1–60, 2019.

[11] J. Abella, D. Hardy, I. Puaut, E. Quinones, and F. J. Cazorla, "On the comparison of deterministic and probabilistic WCET estimation techniques," in *2014 26th Euromicro Conference on Real-Time Systems*. IEEE, 2014, pp. 266–275.

[12] G. Bernat, A. Colin, and S. M. Petters, "WCET analysis of probabilistic hard real-time systems," in *23rd IEEE Real-Time Systems Symposium, 2002. RTSS 2002.* IEEE, 2002, pp. 279–288.

[13] S. Bozhko, F. Marković, G. von der Brüggen, and B. B. Brandenburg, "What really is pWCET? a rigorous axiomatic proposal," in *2023 IEEE Real-Time Systems Symposium (RTSS).* IEEE, 2023, pp. 13–26.

[14] Z. Houssam-Eddine, N. Capodieci, R. Cavicchioli, G. Lipari, and M. Bertogna, "The HPC-DAG task model for heterogeneous real-time systems," *IEEE Transactions on Computers*, 2020.

[15] J. Zhao, D. Du, X. Yu, and H. Li, "Risk scenario generation for autonomous driving systems based on causal bayesian networks," *arXiv preprint arXiv:2405.16063*, 2024.

[16] R. Maier, L. Grabinger, D. Urlhart, and J. Mottok, "Causal models to support scenario-based testing of ADAS," *IEEE Transactions on Intelligent Transportation Systems*, 2023.

[17] T. Han and K. Kim, "Minimizing probabilistic end-to-end latencies of autonomous driving systems," in *2023 IEEE 29th Real-Time and Embedded Technology and Applications Symposium (RTAS)*. IEEE, 2023, pp. 27–39.

[18] V. Nélis, P. M. Yomsi, and L. M. Pinho, "The variability of application execution times on a multi-core platform," in *16th International Workshop on Worst-Case Execution Time Analysis (WCET 2016)*. Schloss-Dagstuhl-Leibniz Zentrum für Informatik, 2016.

[19] F. J. Cazorla, E. Quiñones, T. Vardanega, L. Cucu, B. Triquet, G. Bernat, E. Berger, J. Abella, F. Wartel, M. Houston *et al.*, "Proartis: Probabilistically analyzable real-time systems," *ACM Transactions on Embedded Computing Systems (TECS)*, vol. 12, no. 2s, pp. 1–26, 2013.

[20] M. Ziccardi, E. Mezzetti, T. Vardanega, J. Abella, and F. J. Cazorla, "EPC: extended path coverage for measurement-based probabilistic timing analysis," in *2015 IEEE Real-Time Systems Symposium*. IEEE, 2015, pp. 338–349.

[21] Q. He, J. Sun, N. Guan, M. Lv, and Z. Sun, "Real-time scheduling of conditional DAG tasks with intra-task priority assignment," *IEEE Transactions on computer-aided design of integrated circuits and systems*, 2023.

[22] H. Yi, J. Liu, M. Yang, Z. Chen, and X. Jiang, “Improved convolution-based analysis for worst-case probability response time of CAN,” 2024. [Online]. Available: https://arxiv.org/abs/2411.05835

[23] A. Melani, M. Bertogna, V. Bonifaci, A. Marchetti-Spaccamela, and G. Buttazzo, “Schedulability analysis of conditional parallel task graphs in multicore systems,” *IEEE Transactions on Computers*, 2016.

[24] R. I. Davis and A. Burns, “A survey of hard real-time scheduling for multiprocessor systems,” *ACM computing surveys (CSUR)*, vol. 43, no. 4, pp. 1–44, 2011.

[25] Y. Sun and G. Lipari, “Response time analysis with limited carry-in for global earliest deadline first scheduling,” in *2015 IEEE Real-Time Systems Symposium*. IEEE, 2015, pp. 130–140.

[26] Z. Jiang, S. Zhao, R. Wei, Y. Gao, and J. Li, “A cache/algorithm co-design for parallel real-time systems with data dependency on multi/many-core system-on-chips,” in *Proceedings of the 61st ACM/IEEE Design Automation Conference*, 2024, pp. 1–6.

[27] N. Ye, J. P. Walker, and C. Rüdiger, “A cumulative distribution function method for normalizing variable-angle microwave observations,” *IEEE Transactions on Geoscience and Remote Sensing*, 2015.