Workload-Aware Harmonic Partitioned Scheduling for Probabilistic Real-Time Systems

Jiankang Ren[1], Ran Bi[1], Xiaoyan Su[2], Qian Liu[1], Guowei Wu[3], and Guozhen Tan[1]
[1]School of Computer Science and Technology, Dalian University of Technology, China
[2]Faculty of Management and Economics, Dalian University of Technology, China
[3]School of Software Technology, Dalian University of Technology, China

***Abstract*—Multiprocessor platforms, widely adopted to realize real-time systems nowadays, bring the probabilistic characteristic to such systems because of the performance variations of complex chips. In this paper, we present a harmonic partitioned scheduling scheme with workload awareness for periodic probabilistic real-time tasks on multiprocessors under the fixed-priority preemptive scheduling policy. The key idea of this research is to improve the overall schedulability by strategically arranging the workload among processors based on the exploration of the harmonic relationship among probabilistic real-time tasks. In particular, we define a harmonic index to quantify the harmonicity among probabilistic real-time tasks. This index can be obtained via the harmonic period transformation and probabilistic cumulative worst case utilization calculation of these tasks. The proposed scheduling scheme first sorts tasks with respect to the workload, then packs them to processors one by one aiming at minimizing the increase of harmonic index caused by the task assignment. Experiments with randomly generated task sets show significant performance improvement of our proposed approach over the existing harmonic partitioned scheduling algorithm for probabilistic real-time systems.**

I. Introduction

With the rapid development of chip technology, complex multiprocessor platforms are more and more widely applied in safety-critical domains to accommodate the increasingly high demand from computationally-intensive workloads. Although they can provide better average execution time for every task, extremely large worst case execution time (WCET) may be experienced by some tasks due to the complexity of architecture, leading to high over-provisioning to meet real-time requirements in the real-world implementation. To alleviate the over-provisioning while satisfying real-time constraints, probabilistic approaches were proposed to handle the uncertainties in WCETs [1]. Furthermore, safety-critical systems in avionics, automotive and industrial control tend to integrate multiple functionalities with different safety requirements on a shared platform, where the safety is usually certificated based on the failure probability [2]–[4]. Therefore, it is important to provide an efficient multiprocessor scheduling strategy for real-time systems with probabilistic WCETs, such that probabilistic safety requirements can be guaranteed while achieving efficient resource utilization.

Edgar and Burns [5] presented the pioneer work on the probabilistic occurrence of the WCET of a real-time task in 2001. Since then, a rich literature has developed for probabilistic real-time scheduling in uniprocessor platforms (e.g., [6]–[13]). Recently, efforts on the probabilistic real-time scheduling have shifted towards multiprocessor platforms. For probabilistic real-time tasks on multiprocessors, Wang et al. [14], [15] introduced a harmonicity aware task partitioned scheduling algorithm. Its key idea is to explore the harmonic relationship among tasks, and allocate tasks with harmonic periods to one processor in order to improve the system usage. Experimental results showed that the system schedulability was greatly improved compared to the traditional bin-packing approaches. However, this existing solution has three major drawbacks:

- It cannot fully capture the harmonic relationship of tasks on multiprocessor platforms. This approach quantifies the harmonic relationship among all unassigned tasks with respect to reference tasks. However, the harmonicity of a subset of tasks deployed on one processor only depends on the tasks assigned to this processor, rather than other tasks.
- The performance degrades when serving heavy (high-utilization) tasks. This is because it performs resource allocation on the basis of task groups, but does not consider the workload characteristic of each task, which leads to the "fragmentation" problem (see Section III-A for details), — some tasks cannot be allocated even though there remains relatively large total capacity on all processors.
- It requires a significant number of feasibility tests when determining candidate harmonic sub-task sets. This is highly undesirable because the feasibility test for probabilistic real-time systems is exponentially complex with respect to the number of tasks and the size of probabilistic WCETs [8] (which may involve thousands of parameters for complex real-world applications [16]).

In this paper, we propose a novel probabilistic real-time scheduling scheme to overcome all the above problems. In particular, we develop a workload-aware harmonic partitioning algorithm (named HWAP) which integrates both harmonicity and workload characteristic into the proposed scheduling scheme to solve the fragmentation problem and improve the system schedulability. One major contribution of this paper lies in the design of a harmonic index to quantify the harmonic relationship among tasks based on their probabilistic cumulative worst case utilization. Before the task assignment, all tasks are sorted with respect to the workload in a decreasing order to reduce the fragmentation. Then the sorted tasks are

978-3-9819263-0-9/DATE18/©2018 EDAA

packed to processors one by one, which can achieve a minimal increase on the harmonic index caused by the deployment of tasks. HWAP can enable the efficient resource optimization with relatively low computational cost. This is because (i) the fragmentation problem is effectively alleviated; (ii) when computing the harmonic relationship of unassigned tasks, in contrast to the existing approach [14], [15], we will not consider all tasks in the unassigned task set as reference tasks, but only select tasks that are potentially assigned to the same processor. Furthermore, we evaluate the proposed scheduling scheme on randomly generated task systems, and simulation results show that the schedulability of our proposed scheme outperforms the state-of-the-art harmonic probabilistic real-time scheduling algorithm in multiprocessor systems.

II. Probabilistic Real-Time System Model

The probabilistic real-time system considered in this paper consists of n independent periodic probabilistic real-time tasks, identified by $\Gamma = \{\tau_1,...,\tau_n\}$, and scheduled on a multiprocessor platform with m identical, unit-capacity processors, denoted as $\Pi = \{\pi_1,...,\pi_m\}$, based on Rate Monotonic (RM) scheduling policy. We follow the task model in [7]: each task $\tau_i \in \Gamma$ is characterized by a tuple $\tau_i = (\mathcal{C}_i, T_i, D_i, \zeta_i)$, where

- $\mathcal{C}_i$ is a random variable and denotes the probabilistic worst case execution time (pWCET) of task τ_i;
- T_i is the period of task τ_i;
- D_i is the relative deadline of task τ_i;
- ζ_i is a non-negative real number and denotes the deadline miss probability requirement of task τ_i.

We assume that all tasks are implicit-deadline tasks (i.e., $D_i = T_i$), and they are initially released simultaneously.

The pWCET $\mathcal{C}_i$ of each task $\tau_i \in \Gamma$ has a known probability function $f_{\mathcal{C}_i}(C) = P(\mathcal{C}_i = C)$, which gives the probability that τ_i has a WCET equal to C. We assume that all possible values of $\mathcal{C}_i$ belong to an interval $[C_i^{\mathsf{min}}, C_i^{\mathsf{max}}]$, and thus $\mathcal{C}_i$ can be written as follows:

$$\mathcal{C}_i = \begin{pmatrix} C_i^1 = C_i^{\mathsf{min}} & C_i^2 & ... & C_i^k = C_i^{\mathsf{max}} \\ f_{\mathcal{C}_i}(C_i^{\mathsf{min}}) & f_{\mathcal{C}_i}(C_i^2) & ... & f_{\mathcal{C}_i}(C_i^{\mathsf{max}}) \end{pmatrix}. \quad (1)$$

where k is the number of possible WCET values and $\sum_{j=1}^{k} f_{\mathcal{C}_i}(C_i^j) = 1$. If C is a possible WCET value of task τ_i, we denote it as $C \in \mathcal{C}_i$.

To describe the probabilistic workload of a task τ_i, we can calculate its probabilistic worst case utilization $\mathcal{U}_i$ based on its pWCET and period, i.e.,

$$\mathcal{U}_i = \begin{pmatrix} U_i^1 = \frac{C_i^1}{T_i} & U_i^2 = \frac{C_i^2}{T_i} & ... & U_i^k = \frac{C_i^k}{T_i} \\ f_{\mathcal{U}_i}(U_i^1) & f_{\mathcal{U}_i}(U_i^2) & ... & f_{\mathcal{U}_i}(U_i^k) \end{pmatrix}. \quad (2)$$

where $f_{\mathcal{U}_i}(U_i^j) = f_{\mathcal{C}_i}(C_i^j)$ for each $C_i^j \in \mathcal{C}_i$. Note that $\mathcal{U}_i$ can be specified by its cumulative distribution function (CDF) $F_{\mathcal{U}_i}(x) = \sum_{z=U_i^1}^{x} f_{\mathcal{U}_i}(z)$.

Based on the probabilistic worst case utilization $\mathcal{U}_i$ of each task $\tau_i \in \Gamma$, the probabilistic cumulative worst case utilization of task set Γ is defined as follows:

$$\mathcal{U}_\Gamma = \bigotimes_{\tau_i \in \Gamma} \mathcal{U}_i. \quad (3)$$

where $\otimes$ denotes the convolution of two random variables, for $\mathcal{Z} = \mathcal{X} \otimes \mathcal{Y}$, $P\{\mathcal{Z} = z\} = \sum_{k=-\infty}^{k=+\infty} P\{\mathcal{X} = k\} P\{\mathcal{Y} = z-k\}$.

Probabilistic schedulability: A probabilistic real-time task set Γ is considered *schedulable* under a scheduling policy if the deadline miss probability DMP_i of each task $\tau_i \in \Gamma$ is not larger than its deadline miss probability requirement ζ_i.

In this paper, we assume that all tasks in the task set have the same deadline miss probability requirement, denoted by ζ (i.e., $\zeta_i = \zeta$ for each task $\tau_i \in \Gamma$). Therefore, if the deadline miss probability $\mathsf{DMP}_\Gamma = \max_{\tau_i \in \Gamma}\{\mathsf{DMP}_i\}$ for task set Γ is not larger than ζ, the task set Γ is considered schedulable.

Problem: The main objective of this work is to design an efficient task partitioning strategy to partition a given set of periodic, independent, probabilistic real-time tasks with pWCETs and relative deadlines equal to periods, such that the set of tasks allocated to each processor is schedulable. In this paper, we discuss the limitations of the existing harmonicity aware task partitioning strategy. We also present a feasible harmonic partitioning strategy with workload awareness.

III. Harmonic Partitioned Probabilistic Real-Time Scheduling with Workload-Aware Strategy

A. Motivating the Workload-Aware Strategy

Before introducing our proposed harmonic partitioning algorithm with workload awareness, we first give an example to discuss the limitations of the existing harmonicity aware multiprocessor scheduling and motivate the workload-aware strategy used in this paper. In [14], [15], a harmonicity aware task partitioning strategy is presented for fixed priority scheduling of probabilistic real-time tasks on multiprocessor platforms (called HAP for short). HAP assigns tasks group by group to allocate as many tasks with closer harmonic relationship to a reference task as possible to the same processor. The basic idea of HAP can be briefly described as below:

- Among all unassigned tasks, a reference task is selected from the first task till the last task.
- For each reference task, all other tasks are sorted according to the harmonic index values, and selected from high value to low to form a candidate task subset until the probabilistic schedulability test fails.
- A suitable subset is selected from all candidate subsets based on the mean utilization or a utilization threshold.
- The tasks in the chosen subset are allocated to an empty processor.

The above process is repeated until all tasks are assigned. It has been demonstrated that HAP can improve the schedulability of probabilistic real-time tasks compared to deterministic approaches and traditional bin-packing approaches. Nevertheless, this harmonic-aware strategy fails to improve the schedulability of some task sets. The reason for this is as follows. The advantage of HAP is its capability to group harmonic tasks on the same processor. However, it neglects the fragmentation effect on the system schedulability, which leads to poor performance for some task systems, especially

TABLE I: An example task set: task parameters

Task	$\mathcal{C}_i$	T_i	D_i	ζ_i
τ_1	$\begin{pmatrix} 5 & 6 \\ 0.9 & 0.1 \end{pmatrix}$	9	9	0.1
τ_2	$\begin{pmatrix} 2 & 3 \\ 0.9 & 0.1 \end{pmatrix}$	8	8	0.1
τ_3	$\begin{pmatrix} 3 & 4 \\ 0.9 & 0.1 \end{pmatrix}$	8	8	0.1
τ_4	$\begin{pmatrix} 2.5 & 3 \\ 0.9 & 0.1 \end{pmatrix}$	11	11	0.1
τ_5	$\begin{pmatrix} 2 & 3 \\ 0.9 & 0.1 \end{pmatrix}$	10	10	0.1

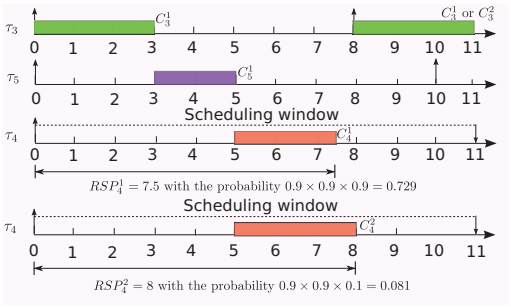

Fig. 1: Simulation of a SAS for $\{\tau_3, \tau_4, \tau_5\}$.

the ones containing heavy tasks. In the following, we illustrate the fragmentation problem with an example.

Example 1. Consider a task set $\Gamma = \{\tau_1, \tau_2, \tau_3, \tau_4, \tau_5\}$ depicted in Table I to be scheduled on two processors $\Pi = \{\pi_1, \pi_2\}$ under RM policy. By HAP, all tasks are divided into three groups $\{\tau_1, \tau_5\}$, $\{\tau_2, \tau_3\}$,$\{\tau_4\}$ one by one. This implies that three processors are required for HAP to schedule this task set. Due to the fragmentation problem, the partitioning of Γ fails with HAP for two processors. If we use First-Fit (FF) strategy to partition this task set, τ_1 and τ_2 will be allocated to processor π_1, after which τ_3, τ_4 and τ_5 are allocated. The minimum utilization sum of τ_1 and τ_2 is $5/9 + 2/8 > 0.8$. For τ_3, τ_4 and τ_5, their minimum utilizations are not less than 0.2. Hence, no task in $\{\tau_3,\tau_4,\tau_5\}$ can be allocated to processor π_1. Now, we consider that τ_3, τ_4 and τ_5 are allocated to π_2. As a simulation of a synchronous arrival sequence (SAS) for $\{\tau_3, \tau_4, \tau_5\}$ in time interval [0, 11] in Fig. 1, the response time distribution of τ_4 is $\begin{pmatrix} 7.5 & 8 & D_4^+ \\ 0.729 & 0.081 & 0.19 \end{pmatrix}$, where D_4^+ denotes the response time values larger than deadline D_4. Note that only the deadline meeting scenarios for τ_4 are given in the simulation of SAS. We can find that the deadline miss probability of τ_4 is $0.19 > \zeta_4$, so it is also infeasible to assign τ_3, τ_4 and τ_5 to π_2. Hence, tasks in this task set are failed to be allocated two processors with FF packing strategy. However, there exists a feasible allocation for this task set. From the simulation of SAS for $\{\tau_1, \tau_4\}$ and $\{\tau_2, \tau_3, \tau_5\}$ in Figs. 2 and 3, we can see that the response time distribution of τ_4 is $\begin{pmatrix} 7.5 & 8 & 8.5 & 9 \\ 0.81 & 0.09 & 0.09 & 0.01 \end{pmatrix}$, and the response time distribution of τ_5 is $\begin{pmatrix} 7 & 8 & D_5^+ \\ 0.729 & 0.243 & 0.028 \end{pmatrix}$. Therefore, it is feasible to allocate τ_1 and τ_4 to π_1, and to assign τ_2, τ_3 and τ_5 to π_2. In this paper, we will give a feasible harmonic

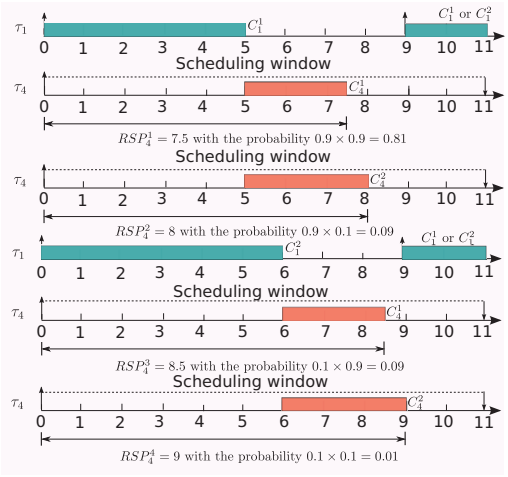

Fig. 2: Simulation of a SAS for $\{\tau_1, \tau_4\}$.

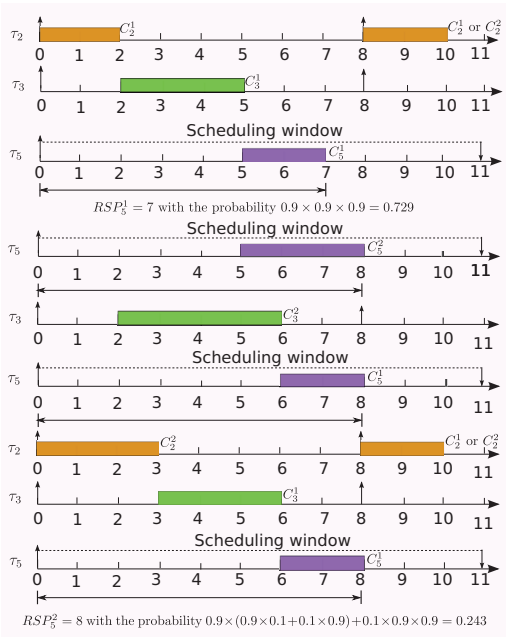

Fig. 3: Simulation of a SAS for $\{\tau_2, \tau_3, \tau_5\}$.

partitioning strategy with workload awareness for Γ, which can take advantage of harmonic relationship exploration while tackling the fragmentation problem (see Example 4).

B. Workload Awareness

It has been shown that the fragmentation problem can be alleviated by ordering tasks first based on certain criteria to achieve workload awareness, and then processing them in that order. In this paper, we consider the Decreasing Utilization (DU) for ordering tasks prior to the application of our partitioning heuristics. Since a probabilistic real-time task is associated with multiple utilization values, we consider the following two options to sort tasks by identifying a single utilization value for each task to characterize its workload.

- **Decreasing Expected Utilization (DEU)**: All tasks in task set Γ are sorted in the decreasing order of their expected utilizations. Given a task $\tau_i \in \Gamma$, its expected

TABLE II: Expected utilization and nominal utilization of task

Task	$\mathcal{U}_i$	$\mathsf{U}_i^{\mathsf{e}}$	$\mathsf{U}_i^{\mathsf{n}}$
τ_1	$\begin{pmatrix} 5/9 & 2/3 \\ 0.9 & 0.1 \end{pmatrix}$	17/30	5/9
τ_2	$\begin{pmatrix} 0.25 & 0.375 \\ 0.9 & 0.1 \end{pmatrix}$	0.2625	0.25
τ_3	$\begin{pmatrix} 0.375 & 0.5 \\ 0.9 & 0.1 \end{pmatrix}$	0.3875	0.375
τ_4	$\begin{pmatrix} 5/22 & 3/11 \\ 0.9 & 0.1 \end{pmatrix}$	51/220	5/22
τ_5	$\begin{pmatrix} 0.2 & 0.3 \\ 0.9 & 0.1 \end{pmatrix}$	0.21	0.2

utilization $\mathsf{U}_i^{\mathsf{e}}$ is calculated as the ratio between its expected WCET C_i^{e} and its period T_i, i.e.,

$$\mathsf{U}_i^{\mathsf{e}} = \frac{C_i^{\mathsf{e}}}{T_i} = \frac{\sum_{j=1}^{k} C_i^j \times f_{\mathcal{C}_i}(C_i^j)}{T_i}. \quad (4)$$

- **Decreasing Nominal Utilization (DNU)**: All tasks in task set Γ are ordered in the decreasing order of their nominal utilizations. Given a task $\tau_i \in \Gamma$, its nominal utilization $\mathsf{U}_i^{\mathsf{n}}$ is calculated as the ratio between its nominal WCET C_i^{n} and its period T_i, i.e.,

$$\mathsf{U}_i^{\mathsf{n}} = \frac{C_i^{\mathsf{n}}}{T_i} = \frac{\min\limits_{C_i^j \in \mathcal{C}_i, F_{\mathcal{C}_i}(C_i^j) \geq 1-\zeta_i} \{C_i^j\}}{T_i}. \quad (5)$$

Example 2. Consider the task set $\Gamma = \{\tau_1, \tau_2, \tau_3, \tau_4, \tau_5\}$ depicted in Table I. As shown in Table II, we can get the expected utilization and the nominal utilization for each task in Γ based on its pWCET, period and deadline miss probability requirement.

C. Harmonic Index of Task Set

Before quantifying the harmonic property of the probabilistic real-time task set, we first give some related definitions based on the counter for deterministic real-time systems [17].

Definition III.1. *Given a probabilistic real-time task set $\Gamma = \{\tau_1, \tau_2, ..., \tau_n\}$ where $\tau_i = (\mathcal{C}_i, T_i, D_i, \zeta_i)$, let $\Gamma' = \{\tau'_1, \tau'_2, ..., \tau'_n\}$ where $\tau'_i = (\mathcal{C}_i, T'_i, D_i, \zeta_i)$, $T'_i \leq T_i$, and T'_j is an integer multiple of T'_i for tasks τ'_i and τ'_j in Γ' if $T'_j \geq T'_i$. Then Γ' is called a* **sub-harmonic task set** *of Γ.*

Definition III.2. *Let Γ' be a sub-harmonic task set of Γ. Then Γ' is called a* **primary harmonic task set** *of Γ if there exists no other sub-harmonic task set Γ'' such that $T'_i \leq T''_i$ for every task $\tau_i \in \Gamma$.*

All primary harmonic task sets can be obtained for a task set by the harmonic period transformation using DCT (Distance-Constrained Tasks) algorithm in [18]. Now, we define the harmonic index for a probabilistic real-time task set by quantifying the "distance" of this task set to the corresponding harmonic task sets in terms of the cumulative distribution of its total utilization.

Definition III.3. *Given a probabilistic real-time task set Γ, let $\Omega(\Gamma)$ represent all primary harmonic task sets of Γ. Then the* **harmonic index** *of Γ, denoted as $\mathcal{H}(\Gamma)$, is defined as*

$$\mathcal{H}(\Gamma) = \min_{\Gamma' \in \Omega(\Gamma)} \Delta\mathcal{U}(\Gamma'). \quad (6)$$

TABLE III: Task parameters of primary harmonic task sets

Task	T'_i	$\mathcal{U}'_i$	T''_i	$\mathcal{U}''_i$
τ_2	8	$\begin{pmatrix} 0.25 & 0.375 \\ 0.9 & 0.1 \end{pmatrix}$	5	$\begin{pmatrix} 0.4 & 0.6 \\ 0.9 & 0.1 \end{pmatrix}$
τ_3	8	$\begin{pmatrix} 0.375 & 0.5 \\ 0.9 & 0.1 \end{pmatrix}$	5	$\begin{pmatrix} 0.6 & 0.8 \\ 0.9 & 0.1 \end{pmatrix}$
τ_5	8	$\begin{pmatrix} 0.25 & 0.375 \\ 0.9 & 0.1 \end{pmatrix}$	10	$\begin{pmatrix} 0.2 & 0.3 \\ 0.9 & 0.1 \end{pmatrix}$

where $\Delta\mathcal{U}(\Gamma') = \|F_{\mathcal{U}_\Gamma}(\mathbf{u}) - F_{\mathcal{U}_{\Gamma'}}(\mathbf{u})\|$, which is used to quantify the "distance" of a task set Γ to its harmonic task set Γ' in terms of the probabilistic cumulative worst case utilization based on $\ell^2 - norm$ operation. $\mathbf{u}$ represents a sequence of all possible utilization values of $\mathcal{U}_\Gamma$ and $\mathcal{U}_{\Gamma'}$.

Note that, $\mathcal{U}_\Gamma$ and $\mathcal{U}_{\Gamma'}$ may have different possible utilizations, and thus $F_{\mathcal{U}_\Gamma}(\mathbf{u})$ and $F_{\mathcal{U}_{\Gamma'}}(\mathbf{u})$ are obtained by enumerating all possible utilizations of Γ and Γ'.

Example 3. Consider a task set $\Gamma_{\mathsf{sub}} = \{\tau_2, \tau_3, \tau_5\}$ that is a subset of task set Γ depicted in Table I. As shown in Table III, it gives task parameters of primary harmonic task sets Γ'_{sub} and Γ''_{sub} for Γ_{sub}, where T'_i, $\mathcal{U}'_i$, T''_i and $\mathcal{U}''_i$ are parameters of corresponding tasks in Γ'_{sub} and Γ''_{sub}. We can get $\Delta\mathcal{U}(\Gamma'_{\mathsf{sub}}) = 0.2446$, and $\Delta\mathcal{U}(\Gamma''_{\mathsf{sub}}) = 0.6596$. Therefore, the harmonic index of Γ_{sub} is $\min\{\Delta\mathcal{U}(\Gamma'_{\mathsf{sub}}), \Delta\mathcal{U}(\Gamma''_{\mathsf{sub}})\} = 0.2446$.

D. Partitioning Algorithm

In this section, we present a partitioning algorithm for assigning a set of probabilistic real-time tasks Γ to m identical, unit-capacity processors Π. Our algorithm aims to find a feasible processor for each task, while exploring harmonic relationship by minimizing the increase of harmonic index caused by a task assignment, such that a task can be executed along with harmonic tasks.

Algorithm 1 HWAP partitioning algorithm.

```
1: Γ(π_j) ← ∅, for all j = 1, ..., m.
2: Sort all tasks in Γ in decreasing order of expected utilization (i.e., DEU)
   or nominal utilization (i.e., DNU) and then in increasing order of period.
3: while Γ ≠ ∅ do
4:     τ_first ← The first task in Γ
5:     Γ ← Γ \ {τ_first}
6:     π_selected ← NIL
7:     Δ_min ← ∞
8:     for each processor π_j ∈ Π do
9:         if Γ(π_j) = ∅ then
10:            if π_selected = NIL then
11:                π_selected ← π_j
12:            break
13:        else
14:            if DMP_{Γ(π_j)∪{τ_first}} ≤ ζ then
15:                Δ ← H(Γ(π_j) ∪ {τ_first}) − H(Γ(π_j))
16:                if Δ < Δ_min then
17:                    Δ_min ← Δ
18:                    π_selected ← π_j
19:    if π_selected ≠ NIL then
20:        Γ(π_selected) ← Γ(π_selected) ∪ {τ_first}
21:    else
22:        return FAILURE
23: return SUCCESS
```

Algorithm 1 is our partitioning algorithm, written in pseudocode. In this algorithm, the set of tasks allocated to the processor π_j is identified by $\Gamma(\pi_j)$. The algorithm starts by

initializing $\Gamma(\pi_j)$ to null (Line 1) and by sorting all tasks according to certain criteria (Line 2). The algorithm then tries to select a suitable processor for each task in the task set (Lines 4–18). The first task τ_{first} in Γ is selected (Line 4), and it is removed from the task set (Line 5). Here, variables π_{selected} and Δ_{min} are used to record the selected processor for task τ_{first} and the minimum harmonic index increase due to the deployment of task τ_{first}. The iterative processor test is performed until an empty processor (Lines 9–12). If no feasible processor is available for task τ_{first} (i.e., $\pi_{\mathsf{selected}} = \mathsf{NIL}$) and there is no task deployed on the current processor π_j under test (i.e., $\Gamma(\pi_j) = \emptyset$), this processor is selected as the host of task τ_{first} (Lines 10–11). At each processor test, the algorithm selects a feasible processor for task τ_{first} such that the harmonic index increase caused by the task deployment is minimal (Line 14–18). For the feasible checking, the schedulability analysis can be performed based on the worst case response time distribution computation approach proposed in [8]. If a feasible processor π_{selected} is obtained, task τ_{first} is assigned to π_{selected} (Lines 19–20). Otherwise, the algorithm aborts with a failure (Line 22). If every task is successfully allocated to a processor, the algorithm reports success (Line 23).

Consider any processor $\pi_j \in \Pi$, and the task set $\Gamma(\pi_j)$ assigned to π_j. According to lines 10 and 14 in Algorithm 1, the deadline miss probability of $\Gamma(\pi_j)$ is not larger than the requirement, and thus $\Gamma(\pi_j)$ is schedulable on processor π_j. Therefore, the resulting task allocation obtained by Algorithm 1 always ensures that the tasks on each processor can be successfully scheduled.

Note that, different from the existing harmonicity aware solution HAP that tries to assign tasks group by group based on the harmonic relationship to reference tasks, HWAP allocates tasks one by one that are sorted with respect to their workload, and it explores harmonic relationship by minimizing the harmonic index increase caused by each task assignment. Hence, HWAP can take the advantage of both harmonic relationship exploration and task ordering with respect to the workload to improve the system usage. For the computational complexity, $\mathcal{O}(mn)$ feasibility tests are required by HWAP, while $\mathcal{O}(n^3)$ for HAP. For each feasibility test, HAP and HWAP use the same schedulability analysis approach. Therefore, the complexity of HWAP is far less than HAP with respect to the increase of the task number n.

Example 4. Consider the task set $\Gamma = \{\tau_1, \tau_2, \tau_3, \tau_4, \tau_5\}$ depicted in Table I again, we show that it is feasible to allocate this task set to two processors $\Pi = \{\pi_1, \pi_2\}$ by HWAP. Suppose all tasks are sorted based on DEU, then the order of tasks is τ_1, τ_3, τ_2, τ_4 and τ_5. Task τ_1 will be allocated to processor π_1, and task τ_3 will be allocated to processor π_2 since it is unschedulable for $\{\tau_1, \tau_3\}$ on the same processor. Then, task τ_2 will be allocated to processor π_2 although it is schedulable for $\{\tau_1, \tau_2\}$ on processor π_1. This is because it is more harmonic for τ_2 to be scheduled along with τ_3. After that, since it is not feasible for τ_4 to be scheduled along with τ_2 and τ_3 on processor π_2, τ_4 will be allocated to processor π_1. At last, τ_5 will be allocated to processor π_2. Until now, τ_1 and τ_4 are allocated to processor π_1, and τ_2, τ_3 and τ_5 are allocated to processor π_2. According to the analysis given in Example 1, this is a feasible partition. Note that the FF strategy with DEU or DNU still cannot partition task set Γ for two processors, because it does not take the task period relationship in consideration. The analysis for the FF strategy with DEU or DNU is omitted here, since it is similar with the analysis for the pure FF strategy given in Example 1.

IV. Experimental Evaluation

In this section, we experimentally compare the performance of our scheme, HWAP, with the existing harmonic partitioned scheduling algorithm HAP (which is based on the utilization sum based harmonic index) from [14], [15]. We have two main goals for our evaluation: (1) to compare the performance of HWAP in terms of the schedulability and the required number of processors to that of HAP; (2) to evaluate the computational complexity of HWAP.

For our experiments, we used randomly generated task sets, which we created using the UUniFast approach from [19]. All tasks are implicit-deadline probabilistic real-time tasks, and they have the same deadline miss probability requirement, which is set to 0.1. For each task set, the task number is 12, the value number of pWCETs is 8. The possible WCETs of a task are integers drawn at random from the interval [1, 100], and their probability are also randomly generated such that the sum of all probability is equal to one. Then, the period of each task τ_i in the task set is calculated as $\lceil C_i^{\mathsf{e}}/U_i^{\mathsf{e}} \rceil$ based on its expected WCET C_i^{e} and expected utilization U_i^{e} generated by the UUniFast approach. We define the normalized expected utilization $\mathsf{U}_{\mathsf{nor}}(\Gamma)$ of a task set Γ to be:

$$\mathsf{U}_{\mathsf{nor}}(\Gamma) = \frac{\mathsf{U}^{\mathsf{e}}(\Gamma)}{m}. \tag{7}$$

where $\mathsf{U}^{\mathsf{e}}(\Gamma)$ is the expected cumulative utilization of Γ, and m is the number of processors. Using the above parameters, we generated the tasks one at a time until the following condition on system utilization was satisfied: $\mathsf{U}^*_{\mathsf{nor}} - 0.005 \leq \mathsf{U}_{\mathsf{nor}}(\Gamma) \leq \mathsf{U}^*_{\mathsf{nor}} + 0.005$, where $\mathsf{U}^*_{\mathsf{nor}} \in \{0.5, 0.55, 0.6, 0.65, 0.7, 0.75, 0.8, 0.85, 0.9, 0.95, 0.975\}$ and $m = 4$. For each $\mathsf{U}^*_{\mathsf{nor}}$, we generated 100 task sets.

Probabilistic schedulability. Fig. 4 shows the fraction of schedulable task sets versus the normalized expected utilization of different algorithms. We can find that HWAP *consistently outperforms* the existing harmonic partitioning algorithm HAP, and as much as an extra 10% of task sets can be scheduled by HWAP. This is because, with HWAP, the system schedulability is improved not only through harmonic relationship exploration, but also via workload awareness. The fragmentation problem can be effectively alleviated by the task ordering with respect to the workload. The performance gap tends to widen as the expected utilization increases; the reason is that, as the expected utilization rises, heavy tasks are more likely to be included in a task set, and thus there are more opportunities for HWAP to improve the system schedulability

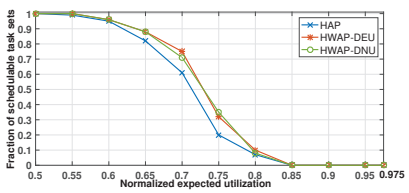

Fig. 4: Fraction of schedulable task sets.

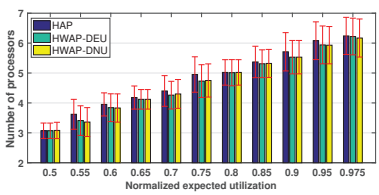

Fig. 5: Number of processors per task set.

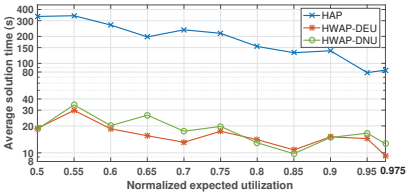

Fig. 6: Average solution time per task set.

by alleviating the fragmentation problem. We can also see that HWAP-DEU has a better performance than HWAP-DNU in most cases. DEU can capture the probabilistic characteristics of a task accurately, whereas the strength of DNU is the capability of taking the probabilistic timing requirement into account to quantify the task workload. For this reason, HWAP-DEU does not always outperform HWAP-DNU.

Number of processors. Fig. 5 illustrates the number of processors to schedule all tasks in a task set versus the normalized expected utilization. Not surprisingly, HWAP outperforms HAP in most test cases; the reason is, again, HWAP enhances the performance by harmonic relationship exploration and workload awareness. Additionally, from the standard deviation values given by the error bar, we can see that the performance of HWAP is more stable than HAP in most test cases owing to the preprocessing of sorting all tasks.

Computation complexity. Fig. 6 shows the time needed per task set to assign all tasks; we report the average across all task sets. We can find that HWAP also *consistently outperforms* HAP. Note that a base 10 logarithmic scale is used for y-axis in Fig. 6. The computation of HWAP-DEU and HWAP-DNU can be completed in tens of seconds, which is significantly less than the solution time of HAP. The main reason for this is that HWAP requires fewer feasibility tests than HAP. Furthermore, HWAP sorts all tasks in decreasing order of workload before the task allocation, and thus for a task being assigned, its feasibility test tends to be performed along with fewer tasks on a processor. Therefore, the solution time of the feasibility test is reduced for HWAP. We also can see that the solution time is decreasing with respect to utilization. This is because the task number is constant in our setting, fewer tasks can be deployed on a processor for task sets with higher utilizations, and hence the cost on the feasibility test is reduced.

V. Conclusions

We have proposed HWAP, a harmonic partitioned scheduling technique for multiprocessor probabilistic real-time systems that is based on the harmonic relationship exploration and the decreasing workload task ordering. HWAP works by associating a probabilistic real-time task set with a harmonic index to quantify its harmonic property. In order to alleviate the fragmentation problem, all tasks are sorted with respect to their workload first. Then the sorted tasks are assigned to processors one by one to explore the harmonic relationship by minimizing the harmonic index increase caused by each task deployment. This strategy not only provides harmonic relationship exploration to improve the system schedulability, it also enables workload awareness to alleviate the fragmentation problem. Our evaluation shows that HWAP outperforms the existing harmonic partitioning algorithm for probabilistic real-time systems in terms of system schedulability with lower computational cost.

Acknowledgment

This work is supported in part by the National Natural Science Foundation of China under Grant No.: 61602080, 61602084, 61601080, 61672129, 61772112, 61772113 and 61402078, the National Key R&D Program of China under Grant No.: 2017YFC0704200, the National Science Foundation for Post-doctoral Scientists of China under Grant No.: 2016M591431 and the Fundamental Research Funds for the Central Universities under Grant No.: DUT15RC(3)126.

References

[1] J. Abella, D. Hardy, I. Puaut *et al.*, "On the comparison of deterministic and probabilistic wcet estimation techniques," in *ECRTS*, 2014.
[2] J. Ren and L. T. X. Phan, "Mixed-criticality scheduling on multiprocessors using task grouping," in *ECRTS*, 2015.
[3] A. Masrur, "A probabilistic scheduling framework for mixed-criticality systems," in *DAC*, 2016.
[4] Y. Abdeddaïm and D. Maxim, "Probabilistic schedulability analysis for fixed priority mixed criticality real-time systems," in *DATE*, 2017.
[5] S. Edgar and A. Burns, "Statistical analysis of wcet for scheduling," in *RTSS*, 2001.
[6] D. Maxim, O. Buffet, L. Santinelli, L. Cucu-Grosjean, and R. I. Davis, "Optimal priority assignment algorithms for probabilistic real-time systems," in *RTNS*, 2011.
[7] D. Maxim, M. Houston, L. Santinelli, G. Bernat *et al.*, "Re-sampling for statistical timing analysis of real-time systems," in *RTNS*, 2012.
[8] D. Maxim and L. Cucu-Grosjean, "Response time analysis for fixed-priority tasks with multiple probabilistic parameters," in *RTSS*, 2013.
[9] L. Santinelli and L. Cucu-Grosjean, "A probabilistic calculus for probabilistic real-time systems," *ACM TECS*, 2015.
[10] S. Ben-Amor, D. Maxim, and L. Cucu-Grosjean, "Schedulability analysis of dependent probabilistic real-time tasks," in *RTNS*, 2016.
[11] P. Axer and R. Ernst, "Stochastic response-time guarantee for non-preemptive, fixed-priority scheduling under errors," in *DAC*, 2013.
[12] B. Tanasa, U. D. Bordoloi, P. Eles, and Z. Peng, "Probabilistic response time and joint analysis of periodic tasks," in *ECRTS*, 2015.
[13] L. Santinelli, Z. Guo, and L. George, "Fault-aware sensitivity analysis for probabilistic real-time systems," in *DFT*, 2016.
[14] T. Wang, L. Niu, S. Ren *et al.*, "Multi-core fixed-priority scheduling of real-time tasks with statistical deadline guarantee," in *DATE*, 2015.
[15] T. Wang, S. Homsi, L. Niu, S. Ren, O. Bai, G. Quan, and M. Qiu, "Harmonicity-aware task partitioning for fixed priority scheduling of probabilistic real-time tasks on multi-core platforms," *ACM TECS*, 2017.
[16] L. Cucu-Grosjean, L. Santinelli *et al.*, "Measurement-based probabilistic timing analysis for multi-path programs," in *ECRTS*, 2012.
[17] M. Fan and G. Quan, "Harmonic-aware multi-core scheduling for fixed-priority real-time systems," *IEEE TPDS*, 2014.
[18] C.-C. Han and H.-Y. Tyan, "A better polynomial-time schedulability test for real-time fixed-priority scheduling algorithms," in *RTSS*, 1997.
[19] E. Bini and G. C. Buttazzo, "Measuring the performance of schedulability tests," *Real-Time Systems*, 2005.