Static Approximation Algorithms for Regularity-based Resource Partitioning *

Yu Li, Albert M. K. Cheng

Department of Computer Science
University of Houston
Houston, TX, 77204, USA
http://www.cs.uh.edu

Technical Report Number UH-CS-12-03

May 16, 2012

Keywords: Hierarchical Real-time Scheduling, Resource Partitioning, Multiresource, Multiprocessor, Schedulability Bound, Regularity Bound, Resource Utilization

Abstract

As a hierarchical real-time system framework, the Regularity-based Resource Partition Model allocates physical resources in time intervals determined by integral numbers of a time unit to tasks in different applications. A Regularity-based Resource Partition is characterized by its regularity and availability factor. An important problem is how to schedule a group of resource partitions with the specification of their regularities and availability factors. Mok and Feng have provided the AAF-Single algorithm which works only on a single-resource platform. Li and Cheng have recently extended AAF-Single to AAF-Multi for multiresource platforms without violating the schedulability bound given by Feng. However, the resource utilization of these two algorithms could be significantly improved. This paper is dedicated to developing optimized resource partitioning algorithms for the Regularity-base Resource Partition Model. We first decompose the resource partitioning problem into two subproblems, and invoke the Pfair algorithm to solve the first subproblem. Then we introduce a category of approximation algorithms called Static Approximation Algorithms (SAA) to solve the second subproblem. An SAA adjusts the availability factors of the resource partitions with a specific boundary sequence. We prove that the schedulability bound of any feasible SAA is at most 0.5. Furthermore, we develop an optimal SAA called Magic7. Simulation shows that Magic7-enhanced algorithms improve the resource utilization by 10% or more.

*Supported in part by the National Science Foundation under Award No.0720856.

Static Approximation Algorithms for Regularity-based Resource Partitioning *

Yu Li, Albert M. K. Cheng

Abstract

As a hierarchical real-time system framework, the Regularity-based Resource Partition Model allocates physical resources in time intervals determined by integral numbers of a time unit to tasks in different applications. A Regularity-based Resource Partition is characterized by its regularity and availability factor. An important problem is how to schedule a group of resource partitions with the specification of their regularities and availability factors. Mok and Feng have provided the AAF-Single algorithm which works only on a single-resource platform. Li and Cheng have recently extended AAF-Single to AAF-Multi for multiresource platforms without violating the schedulability bound given by Feng. However, the resource utilization of these two algorithms could be significantly improved. This paper is dedicated to developing optimized resource partitioning algorithms for the Regularity-base Resource Partition Model. We first decompose the resource partitioning problem into two subproblems, and invoke the Pfair algorithm to solve the first subproblem. Then we introduce a category of approximation algorithms called Static Approximation Algorithms (SAA) to solve the second subproblem. An SAA adjusts the availability factors of the resource partitions with a specific boundary sequence. We prove that the schedulability bound of any feasible SAA is at most 0.5. Furthermore, we develop an optimal SAA called Magic7. Simulation shows that Magic7-enhanced algorithms improve the resource utilization by 10% or more.

Keywords

Hierarchical Real-time Scheduling, Resource Partitioning, Multiresource, Multiprocessor, Schedulability Bound, Regularity Bound, Resource Utilization

I. Introduction

With the rapid development of both software and hardware, large-scale, open real-time systems have become increasingly popular. To improve resource utilization, these systems intend to deploy a group of real-time applications on one physical platform. Instead of running on a dedicated physical resource, the real-time tasks belonging to one application run on a real-time virtual resource [2], which can only occupy a temporal partition of the physical resource. Thus a new type of real-time system called the hierarchical real-time system emerges.

A resource partition [2] describes how to assign the time intervals on a physical resource (e.g. a multicore CPU) to a real-time virtual resource. Similar to the period and the execution time for a periodic real-time task, a resource partition also has its own specification. A brief and precise definition of this specification can make it easier to schedule the resource partitions (resource-level scheduling) on the physical resource, as well as to schedule the real-time tasks (task-level scheduling) on the resource partitions. We call the formalization of this specification as the scheduling-interface-definition problem in a hierarchical real-time system. Once a scheduling interface is identified, both the resource-level scheduler and the task-level scheduler can start to work together.

An important factor to consider in the scheduling-interface-definition problem is the type of real-time task models used at the task level. The Regularity-based Resource Partitioning Model [1] is an excellent resource allocation method that works well in tandem with the following widely-adopted task model:

A task T is defined as (c, p), where integer c is the worst case execution time requirement, and integer p is the period and relative deadline of the task. A task can be interrupted or resumed only at integral time instants.

In this task model, the physical resource is allocated in time intervals determined by integral numbers of a time unit, or $quantum$ size. Thus, the Regularity-based Resource Partitioning Model considers the scheduling interface in the domain of integers and possesses distinctively interesting properties.

*Supported in part by the National Science Foundation under Award No.0720856.

A Regularity-based Resource Partition is specified by its regularity and availability factor. Mok and Feng [1] present an algorithm (called AAF-Single in [4]) for scheduling such defined Regularity-based Resource Partitions on a single-resource platform. Then Li and Cheng [4] extend AAF-Single to AAF-Multi for a uniform multiresource platform without violating the previous schedulability bound. In this paper, we develop new regularity-based resource partitioning algorithms which can significantly achieve higher resource utilization than that of AAF-Single and AAF-Multi either on a uniform multiresource platform or on a single-resource one.

The rest of this paper is organized as follows. In Section II we review the Regularity-based Partition Model and its extension to multiresource platforms. Section III shows how to improve the current regularity-based resource partitioning algorithms by incorporating the Pfair approach [5, 6]. An optimal Static Approximation Algorithm called Magic7 is then introduced in Section IV. We show the simulation results in Section V. Finally, we overview the related work and draw the conclusion.

II. The Regularity-Based Resource Partition Model

In this section we first review the concepts of the Regularity-Based Partition Model given in [1, 3] (Several of these concepts originated from Shigero et al [7]). Then we review its extension to multiresource platforms introduced in [4]. We only focus on the scheduling strategies at the resource level, and skip those at the task level because this paper is devoted to investigating resource-level scheduling algorithms.

In the Regularity-Based Resource Partition Model, a resource partition describes how to assign the time intervals by quantum from a physical resource to a real-time virtual resource periodically.

Definition 2.1 *A resource partition* Π *is a tuple* $(\mathcal{S}, p)$*, where* $\mathcal{S} = \langle s_1, s_2, ..., s_n : 0 \leq s_1 < s_2 < ... < s_n < p \rangle$ *is the time-slice sequence, and* p *is the partition period.*

Definition 2.2 $|\mathcal{S}|$ *represents the number of items in the sequence* $\mathcal{S}$*.*

Definition 2.3 *The Availability Factor of a resource partition* $\Pi = (\mathcal{S}, p)$ *is* $\alpha(\Pi) = \frac{|\mathcal{S}|}{p} \in (0, 1]$*.*

Definition 2.4 *The Supply Function* $S(t)$ *of a partition* Π *is the total number of time-slices that is available in* Π *from time 0 to t.*

Most of the timing parameters in the Regularity-based Resource Partition Model are integers, and the resource partitions working in the integer domain show interesting properties. The Regularity-based Partitions, both regular and irregular, are defined by bounding their Supply Regularities, which represent the deviation ranges of their actual resource utilization.

Definition 2.5 *The Instant Regularity* $Ir(t)$ *at time* t *on partition* Π *is given by* $S(t) - t \cdot \alpha(\Pi)$*.*

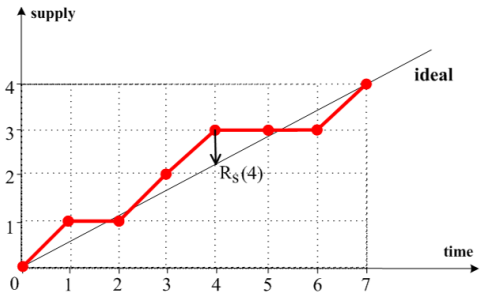


Fig. 1. Instant Regularity at Time 4

Definition 2.6 *Letting* a, b, k *be non-negative integers, the Supply Regularity* $R_s(\Pi)$ *of Partition* Π *is equal to the minimum value of* k *such that* $\forall a, b$*,* $|Ir(b) - Ir(a)| < k$*.*

Definition 2.7 *A Regular Partition is a partition with supply regularity of* 1*.*

Definition 2.8 *A k Supply-Irregular Partition is a partition with supply regularity of k, where $k > 1$.*

The Regular Partition and the Irregular Partition, especially the former, can provide wonderful real-time support to the lower-level real-time tasks working in the integer domain. We shall not mention them here since we are only concerned with the resource partitioning problem in this paper. The problem we want to solve here is:

Problem I *Given a sequence $\langle(\alpha_i, k_i) : 1 \leq i \leq n\rangle$ as the availability factors and supply regularities of n resource partitions, schedule them on a physical (single or multiple) resource.*

Next, we review the algorithms in [3] showing how to solve Problem I on a single-resource platform. The main idea is to convert all resource partitions into a group of regular partitions with availability factors of powers of one-half whose schedulability can be easily checked on a single-resource platform.

Lemma 2.1 *When k regular partitions are combined together, they form a partition with supply regularity of k.*

Definition 2.9 *Given a Partition Π with availability factor α and supply regularity Rs, the Adjusted Availability Factor $\mathcal{AAF}(\alpha,\ Rs)$ is the total of the availability factors of partitions that are used to compose Π.*

The process of computing $\mathcal{AAF}$ of a resource partition can be regarded as an approximating function. The basic idea is to find a set of powers of one-half with a cardinality that does not exceed the supply regularity of the partition. Simultaneously, we attempt to keep the sum of these numbers as small as possible. Feng [3] has developed a algorithm for computing $\mathcal{AAF}$; we present a different recursive algorithm that makes more sense:

$$\mathcal{AAF}(\alpha, k) = \begin{cases} 0 & if\ \alpha = 0; \\ \frac{1}{2^n},\ where\ n = \left\lfloor \log_{\frac{1}{2}} \alpha \right\rfloor & if\ \alpha > 0\ and\ k = 1; \\ \mathcal{AAF}\left(\alpha - \mathcal{L}(\alpha), k-1\right) + \mathcal{L}(\alpha) & otherwise, \end{cases}$$

$where\ \mathcal{L}(\alpha) = \frac{1}{2^n},\ where\ n = \left\lceil \log_{\frac{1}{2}} \alpha \right\rceil.$

For better understandability, we give several examples showing how to compute the $\mathcal{AAF}$ of a partition:

$\mathcal{AAF}(0.17,\ 1) = 0.25,$
$\mathcal{AAF}(0.67,\ 2) = 0.5 + 0.25 = 0.75,$
$\mathcal{AAF}(0.67,\ 3) = 0.5 + 0.125 + 0.0625 = 0.6875,$
$\mathcal{AAF}(0.75,\ 3) = \mathcal{AAF}(0.75,\ 2) = 0.5 + 0.25 = 0.75.$

Theorem 2.1 provides the schedulability bound for Problem I given by Mok and Feng. Partitions are schedulable on a single-resource platform if the sum of their $\mathcal{AAF}$ does not exceed 1. The pseudo code of AAF-Single follows. Time slices are assigned to the regular partitions from heavy (with a higher availability factor) to light.

Theorem 2.1 *Given a sequence $\langle(\alpha_i, k_i) : i = 1, 2, ..., n\rangle$ where α_i is the availability factor and k_i is the supply regularity of a partition P_i, these partitions are schedulable on a single resource if $\sum_{i=1}^{n} \mathcal{AAF}(\alpha_i, k_i) \leq 1$.*

ALGORITHM AAF-SINGLE

(0) $S := \{P_i : i = 1, 2, ..., n\};$
(1) $A_i := \mathcal{AAF}(\alpha_i, k_i)\ for\ all\ i;$
(2) $l := 1;$
(3) **while** $S \neq \phi$ **do**
(4) $\quad w := 1\ /\ 2^l;$
(5) $\quad$ **foreach** $P_j \in S$ **do**
(6) $\quad\quad$ **if** $A_i\ \geq\ w$ **then**
(7) $\quad\quad\quad assign\ time\text{-}slices\ to\ P_j\ at\ level\text{-}l;$
(8) $\quad\quad\quad A_i := A_i - w;$
(9) $\quad\quad\quad$ **if** $A_i = 0$ **then** $S = S - \{P_j\}$ **fi**

(10) **fi**
(11) **od**
(12) l++
(13) **od**

Figure 2 shows an example of the outcome from the algorithm AAF-Single for a partition group $\{(0.375,\ 2),\ (0.3125,\ 2),\ (0.$ Each irregular partition is composed of a number of regular partitions which does not exceed the supply regularity of this irregular partition. Notice that on a regular partition with availability factor of a power of one-half, the numbers of time slices form an arithmetic sequence.

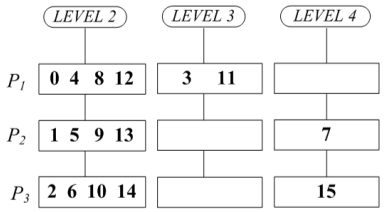


Fig. 2. AAF-Single

We recently [4] present a new algorithm called AAF-Multi which extends the schedulability bound given in Theorem 2.1 for a single-resource platform to one for a multiresource platform. Theorem 2.2 shows this new schedulability bound.

Theorem 2.2 *Given a sequence* $\langle(\alpha_i, k_i) : i = 1, 2, ..., n\rangle$ *where* α_i *is the availability factor and* k_i *is the supply regularity of a partition* P_i*, these partitions are schedulable on* m *resources if* $\sum_{i=1}^{n} \mathcal{AAF}(\alpha_i, k_i) \leq m$.

Suffering from the time-slice overlapping problem, AAF-Single cannot be directly applied onto a multiresource platform. Consider the example in Figure 3, where $\{(0.75,\ 2),\ (0.625,\ 2),\ (0.625,\ 2)\}$ are $\mathcal{AAF}$ and supply regularity of P_1, P_2, and P_3. As shown in the top-left part of Figure 3, these three partitions cannot be scheduled by AAF-Single because the overlap occurrence on P_3 is unavoidable.

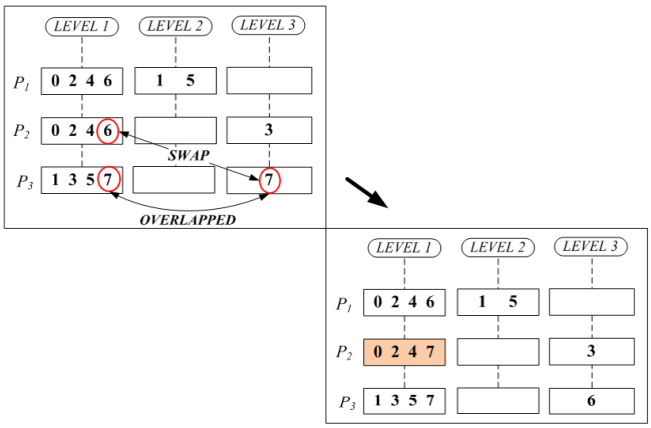


Fig. 3. From AAF-Single to AAF-Multi

To solve the overlapping problem, we [4] have introduced a new type of partition – sub-regular partition. A regular partition with an available factor of a power of one-half is transformed into a sub-regular partition if a subset of its time slices are right-shifted by 1. The first level of P_2 in the bottom-right part of Figure 3 is an example of a sub-regular partition, where the time slice 6 is right-shifted to 7. Instead of regular partitions with available factors of powers of one-half, irregular partitions in the partition group are converted into sub-regular partitions in AAF-Multi. We have proven that the supply regularity of a partition composed of k time sub-regular

partitions is not greater than k. We then develop a deductive methodology which can schedule these sub-regular partitions together with the regular partitions in the partition group on a multiresource platform.

However, the resource utilization provided by AAF-Single and AAF-Multi is not sufficiently high. In the next two sections, we present new approaches to improve it.

III. AAF-Pfair-Mixed: Improvement by Incorporating Pfair

The Pfair scheduling algorithm [5, 6] shares a lot of similarities with the Regularity-based Resource Partitioning Algorithms if we regard resource partitions as periodic real-time tasks. In this section, we shall review some key ideas of Pfair first, and then incorporate Pfair into our resource partitioning algorithms to improve their resource utilization.

A. Introduction to Pfair

The Pfair algorithm solves the real-time task scheduling problem on a multiprocessor platform by observing the following constraints or rules:

- There are n periodic tasks $\{(e_x, p_x) : 1 \leq x \leq n\}$ to be scheduled over m identical processors, where integer e_x is the worst case execution time, and integer p_x is the period and relative deadline of Task x.
- Tasks are interrupted and resumed only at integral time instants.
- No task can run on more than one processor simultaneously.
- Tasks make progress in a steady rate. Task i receives either $\lfloor w_x \cdot t \rfloor$ or $\lceil w_x \cdot t \rceil$ serving time in the time interval $[0, t]$, where $w_x = e_x/p_x$.

The regularity-based resource partitioning algorithms work in a remarkably similar way as Pfair if we consider regularity-based resource partitions as real-time tasks in Pfair. First, they both work in the domain of integers. All input parameters of tasks and resource partitions are integers. Second, neither allows a task or a resource partition to run concurrently. Last but no least, both consider the difference between the actual resource supply and the ideal resource availability. To explain the last similarity, we cite the following conclusion from [5].

A Schedule S is Pfair if and only if

$\forall x, t : 1 \leq x \leq n, t \in N : -1 < lag(S, x, t) < 1,$

where $lag(S, x, t) = w_x \cdot t - \sum_{x=0}^{t-1} S(x, i)$.

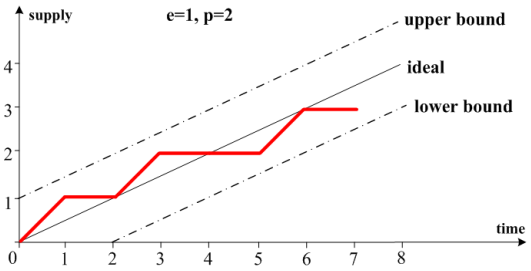


Fig. 4. A Pfair Task

The value of $\sum_{x=0}^{t-1} S(x, i)$ indicates how many time slices Task x has been assigned to by Schedule S at time instant t, which is the same as the Supply Function in Definition 2.3. Furthermore, the definition of function lag shows that its value equals to the minus value of Instant Regularity in Definition 2.4. Because Pfair restricts the value of lag in (-1, 1), if we schedule resource partitions using Pfair, it can be guaranteed that the supply regularity of each partition is 2. This is enough for any irregular partition, but not for any regular partition. Baruah et al [5] show the schedulability bound of Pfair is 1, which means there is no utilization loss in Pfair. If all of the resource partitions are irregular, we can schedule them with Pfair with 100% resource utilization.

B. AAF-Pfair-Mixed Algorithm

The investigation on the Pfair algorithm inspires us to improve the current AAF-based resource partitioning algorithms. Our strategy is relatively straightforward – separating the irregular partitions from the regular ones, as shown in Figure 5.

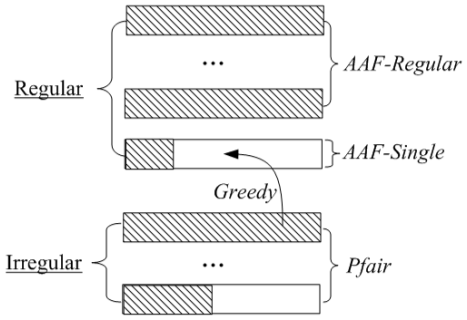


Fig. 5. AAF-Pfair-Mixed

The resource partitions are grouped into two categories: regular and irregular, which are scheduled on separate resources. A hybrid approach is applied, combining global and partitioned scheduling strategies. We schedule regular partitions with AAF-Regular and irregular ones with Pfair. This new hybrid algorithm is called AAF-Pfair-Mixed. Next we present the pseudo code of AAF-Regular which schedules a regular-partition set $\{P_i : i = 1, 2, ..., n\}$ on a multi-resource, where the availability factor of P_i is α_i.

ALGORITHM AAF-REGULAR

(0) $S := \{P_i : i = 1, 2, ..., n\}$;
(1) $A_i := \mathcal{AAF}(\alpha_i, 1)\ for\ all\ i$;
(2) $w := 1; m := 1; r := 1$;
(3) **while** $(S \neq \phi)$ **do**
(4) **foreach** $P_j \in S$ **do**
(5) **if** $(A_i = w)$ **then**
(6) $schedule\ P_i\ to\ the\ m\text{-}th\ resource$;
(7) $r := r - w$;
(8) **if** $(r = 0)$ **then**
(9) m++;
(10) $r := 1$;
(11) **fi**
(12) $S = S - \{P_j\}$;
(13) **fi**
(14) **od**;
(15) $w := w/2$;
(16) **od**

By incorporating Pfair, the resource utilization overhead due to the adjusting of the availability factors of the irregular partitions is removed. However, a new type of overhead emerges as a result of the partitioned scheduling strategy we have used. As shown in Figure 5, partitions in both regular and irregular categories cannot fully utilize the resources assigned to them. In the worst case, nearly 2 resources could be wasted. The overall resource utilization would be drastically impacted especially when the resource number is small. An easy enhancement is migrating some irregular partitions to the last resource belonging to the regular category when it is not fully utilized (AAF-Regular guarantees that the other resources in the regular category are fully utilized). We can use a simple greedy strategy when choosing irregular partitions for migration. Each time we select the fittest one which can utilize the most capacity of the last resource in the regular category. Notice that we have to adjust the availability factor of a

migrating irregular partition; otherwise, it cannot be scheduled together with the regular partitions sharing the same resource. As a result, AAF-Single can work for the last resource in the regular category.

Another advantage of AAF-Pfair-Mixed is that it can restrict the number of migrating partitions. On one hand, there is no partition migrations in AAF-Regular at all. On the other hand, we can use the enhanced Pfair approach in [7], which guarantees that the number of migrating partitions is less than the number of resources. Although this approach does not necessarily reduce the number of migrations, we can still keep those more important partitions fixed to specific resources, which might be desirable sometimes.

IV. MAGIC7: AN OPTIMAL STATIC APPROXIMATION ALGORITHM

Since Pfair cannot schedule regular partitions, we use AAF-Regular to schedule them in the previous section. Are there better solutions than AAF-Regular? We will consider the following problem in this section:

Problem II *Given a sequence* $\langle \alpha_i : 1 \leq i \leq n \rangle$ *as the availability factors of* n *regular partitions, schedule them on a physical (single or multiple) resource.*

Problem II is a special case of Problem I when all the resource partitions are regular. Mok and Feng [1] have announced that this is an NP-hard problem, so it is hard to find an optimal scheduling algorithm for all the combinations of availability factors. However, we can still try to find a generally "good" algorithm.

A. Static Approximation Algorithm

AAF-Regular is an approximation methodology. It adjusts the availability factor of each regular partition to the closest greater or equal value in the sequence $\left\langle 1, \frac{1}{2}, \frac{1}{4}, \frac{1}{8}, ... \right\rangle$. We can also choose other sequences like $\left\langle 1, \frac{1}{3}, \frac{1}{9}, \frac{1}{27}, ... \right\rangle$ or a finite one $\left\langle 1, \frac{4}{5}, \frac{3}{5}, \frac{2}{5}, \frac{1}{5} \right\rangle$. Excluding the fixed element 1, we call such a sequence an Approximating Boundary Sequence (ABS). An ABS satisfies the following:

- *All elements are in the range of* $(0, 1)$.
- *These elements must be strictly decreasing.*
- *If the sequence is infinite, its limit is 0.*

In general, a Static Approximation Algorithm (SAA) adjusts each availability factor to the closest greater or equal value in a specific ABS. This is tantamount to saying that such an algorithm mostly depends on its ABS. There are some other approximation algorithms depending on more than one ABS, or a dynamic ABS, which might be more efficient but much more complicated. Those algorithms are outside the scope of this paper.

Definition 4.1 *An Approximating Boundary Sequence (ABS) is a real number sequence* $\langle b_1, b_2, b_3, ... \rangle$*, where* $\forall i > 0,\ 0 < b_{i+1} < b_i < 1$*;* $\lim b_n = 0$ *if* $n \to \infty$.

Definition 4.2 *The Approximating Function of an ABS* $\mathcal{B} = \langle b_1, b_2, b_3, ... \rangle$ *for* $\alpha \in (0, 1)$ *is*

$\mathcal{R}(\mathcal{B}, \alpha) = b_{i-1}$ *when* $b_i < \alpha \leq b_{i-1}$,

where $b_0 = 1$*;* $b_{|\mathcal{B}|+1} = 0$ *if* $\mathcal{B}$ *is finite.*

Definition 4.3 *An ABS* $\mathcal{B}$ *is feasible if and only if*
$\forall \langle \alpha_1, \alpha_2, ..., \alpha_n \rangle$ *as the availability factors of* n *time regular partitions, they are schedulable on* $\lceil \sum_{i=1}^{n} \mathcal{R}(\mathcal{B}, \alpha_i) \rceil$ *time resources after being adjusted by the Approximating Function of* $\mathcal{B}$*. For convenience, we derive directly that* $\langle \alpha_1, \alpha_2, ..., \alpha_N \rangle$ *is (un)schedulable under* $\mathcal{B}$.

An SAA is feasible if and only if its ABS is feasible. From now on, we will focus on how to find an optimal ABS. There are three factors that we need to consider. The first is the schedulability bound of such an algorithm. The second is the average resource utilization, which indicates the performance of the algorithm. The third is the feasibility of the ABS we find. We will discuss these three factors one by one in this section.

B. Schedulability Bound

The schedulability bound of a feasible SAA is the maximum worst-case utilization for which a regular-partition set can be scheduled by this SAA. Mok and Feng [1] have shown that the schedulability bound of the AAF-Regular Algorithm is 0.5. We will investigate the schedulability bound of general SAAs here. Because the feasibility of an SAA entirely depends on the feasibility of its ABS, we derive the schedulability bound of an ABS directly for convenience. We use $\Upsilon(\mathcal{B})$ to represent the schedulability bound of a feasible ABS $\mathcal{B}$.

Theorem 4.1 *If a feasible ABS $\mathcal{B}$ is finite, $\Upsilon(\mathcal{B}) = 0$.*

Proof: Suppose $\mathcal{B} = \langle b_1, b_2, ..., b_n\rangle$, and $\Upsilon(\mathcal{B}) = \epsilon \in (0, 1]$. Let $N = \lfloor \frac{1}{b} \rfloor + 1 > \frac{1}{b}$, where $b = b_n$ or $b = 1$ if $\mathcal{B}$ is empty. $\langle \alpha_i = \frac{b\cdot\epsilon}{N} : i = 1, 2, ..., N\rangle$ is unschedulable under $\mathcal{B}$ on a single-resource because $\sum_{i=1}^{N} \mathcal{R}(\mathcal{B}, \alpha_i) = N \cdot b > 1$. However, its overall utilization equals to $N \cdot \frac{b\cdot\epsilon}{N} \le \epsilon$, which contradicts $\Upsilon(\mathcal{B}) = \epsilon$. ■

Theorem 4.2 *If a feasible ABS $\mathcal{B} = \langle b_1, b_2, b_3, ...\rangle$ is infinite, $\Upsilon(\mathcal{B}) = \min\left\{\frac{b_{i+1}}{b_i} : i \ge 0\right\}$, where $b_0 = 1$.*

Proof: Let $\epsilon = \min\left\{\frac{b_{i+1}}{b_i} : i \ge 0\right\}$.

• Suppose $\langle \alpha_i : i \in [1, n]\rangle$ is the worst-case availability factor combination to be scheduled on m time resources. Then $\forall \rho \in (0, 1), \sum_{i=1}^{n} \mathcal{R}(\mathcal{B}, \alpha_i) > \rho \cdot m$ since $\mathcal{B}$ is infinite.

Suppose $\alpha_i \in (b_{k_i+1}, b_{k_i}]$, then

$\alpha_i > b_{k_i+1} \ge b_{k_i} \cdot \epsilon = \mathcal{R}(\mathcal{B}, \alpha_i) \cdot \epsilon$.

So $\forall \rho \in (0, 1), \sum_{i=1}^{n} \alpha_i = \sum_{i=1}^{n} \mathcal{R}(\mathcal{B}, \alpha_i) > \rho \cdot \epsilon \cdot m$, which concludes $\Upsilon(\mathcal{B}) \ge \epsilon$.

• If $\Upsilon(\mathcal{B}) > \epsilon$, suppose $\Upsilon(\mathcal{B}) = (1 + 2\delta) \cdot \epsilon$ where $\delta \in \left(0, \frac{1-\epsilon}{2\epsilon}\right]$.

$\exists k, \frac{b_{k+1}}{b_k} < \sqrt{1 + \delta} \cdot \epsilon$.

Let $\alpha = \sqrt{1 + \delta} \cdot b_{k+1} < (1 + \delta) \cdot \epsilon \cdot b_k \le (1 + \frac{1-\epsilon}{2\epsilon}) \cdot \epsilon \cdot b_k < b_k$.

Let $m = \left\lfloor \frac{b_k \cdot (1+\delta)}{\delta} \right\rfloor + 1 > \frac{b_k \cdot (1+\delta)}{\delta}$.

Let $n = \left\lfloor \frac{m}{b_k} \right\rfloor + 1 \in \left(\frac{m}{b_k}, \frac{m}{b_k} + 1\right]$.

The sequence $\langle \alpha_i = \alpha : i = 1, 2, ..., n\rangle$ is unschedulable by $\mathcal{B}$ on m time resources because $n \cdot \mathcal{R}(\mathcal{B}, \alpha) = n \cdot b_k > \frac{m}{b_k} \cdot b_k = m$.

However, its overall utilization is

$$
\begin{aligned}
& n \cdot \sqrt{1 + \delta} \cdot b_{k+1} \\
< & (\tfrac{m}{b_k} + 1) \cdot (1 + \delta) \cdot \epsilon \cdot b_k \\
= & (m + b_k) \cdot (1 + \delta) \cdot \epsilon \\
= & m \cdot (1 + \delta) \cdot \epsilon + b_k \cdot (1 + \delta) \cdot \epsilon \\
< & m \cdot (1 + \delta) \cdot \epsilon + m \cdot \delta \cdot \epsilon \\
= & m \cdot (1 + 2\delta) \cdot \epsilon,
\end{aligned}
$$

which contradicts $\Upsilon(\mathcal{B}) = (1 + 2\delta) \cdot \epsilon$. ■

C. Average Resource Utilization

Besides the schedulability bound, the average resource utilization is another important aspect of a scheduling algorithm. The approximating process leads to some utilization overhead, the ratio of which determines the performance of an SAA. As shown in Figure 6, the polygonal line with a shape of stairs indicates the Approximating Function of an ABS $\mathcal{B} = \langle b_1, b_2, b_3, ...\rangle$, and the overall area of the slash filled triangles indicates the utilization overhead of the approximating process by $\mathcal{B}$.

Definition 4.4 *The Utilization Overhead of an ABS $\mathcal{B} = \langle b_1, b_2, b_3, ...\rangle$ is*

$\mathcal{O}(\mathcal{B}) = \frac{1}{2} \sum_{i\ge0} (b_i - b_{i+1})^2 \in \left(0, \frac{1}{2}\right]$,

where $b_0 = 1$; $b_{|\mathcal{B}|+1} = 0$ if $\mathcal{B}$ is finite.

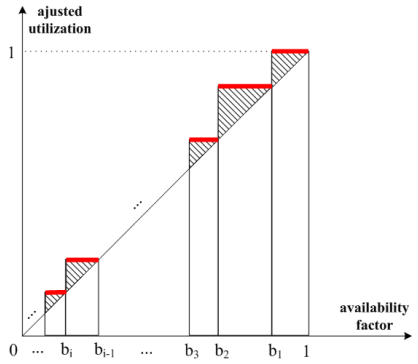


Fig. 6. The Approximating Function and its Overhead

Finally, the average resource utilization of $\mathcal{B}$ equals to $\frac{0.5}{\mathcal{O}(\mathcal{B})+0.5} \in \left[\frac{1}{2}, 1\right)$, which is decreasing when $\mathcal{O}(\mathcal{B})$ is increasing. The average resource utilization equals to the minimum value 0.5 when $\mathcal{B}$ is empty, where a dedicated resource is assigned to each regular partition.

D. Standard Regular Partition

Feasibility is the most difficult part of the research on SAA since resource time slices are assigned to regular partitions in a very strict way. It can be concluded from Definitions 2.5 and 2.6 that the variation range of the instant regularities of a regular partition is smaller than one. There are two possible choices for a regular partition at a specific time instant, either preempting a resource during the next time-slice interval or not. The difference between the instant regularities after one time unit with these two choices equals to one *. This leads to only one feasible choice. Furthermore, Shigero et al draw the following conclusion in [7].

Lemma 4.1 [7] *A regular partition is uniquely determined by its availability factor except for the offset.*

The time-slice sequence on a regular partition is determined once it preempts a resource for the first time, so a regular partition will entirely coincide with another one having the same availability factor after being shifted by a certain offset. We can define a standard regular partition for each specific availability factor, and the rest of the regular partitions with the same availability factor can be easily obtained from it.

Definition 4.5 *A Regular Partition preempting a resource during the first time-slice interval is a Standard Regular Partition.*

Figure 7 shows the time-slice sequence of a standard regular partition with the availability factor of $\frac{4}{7}$. Its time-slice sequence is $\langle 0, 1, 3, 5\rangle$.

Definition 4.6 $\mathcal{T}_0(p, q)$ *represents the time-slice sequence of a standard regular partition, where* p *is the period and* $\frac{q}{p}$ *is the availability factor.*

Definition 4.7 $\mathcal{T}(p, q, \delta)$ *represents the time-slice sequence right-shifted from* $\mathcal{T}_0(p, q)$ *by* δ *time units, where a modulus operation is applied when a time slice is out of* $[0, p)$.

For example, $\mathcal{T}_0(7, 4) = \langle 0, 1, 3, 5\rangle$, $\mathcal{T}(7, 4, 2) = \langle 0, 2, 3, 5\rangle$.

Corollary 4.1 $(\mathcal{S}, p)$ *is a regular partition if and only if* $\exists\delta \in [0, p), \mathcal{S} = \mathcal{T}(p, |\mathcal{S}|, \delta)$.

* Suppose R is a resource partition with the availability factor of α. The instant regularity of R at time $t+1$ can be obtained from the one at time t as follows:

$$Ir(R,\ t+1) = \begin{cases} Ir(R,\ t) + 1 - \alpha & if\ preempting\ a\ resource, \\ Ir(R,\ t) - \alpha & otherwise. \end{cases}$$

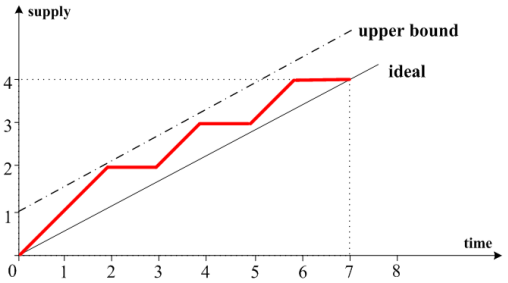


Fig. 7. Standard Regular Partition with the Availability Factor of $\frac{4}{7}$

Lemma 4.2 $\mathcal{T}_0(p,q) = \left\langle 0, \left\lfloor \frac{p}{q} \right\rfloor, \left\lfloor \frac{2p}{q} \right\rfloor, ..., \left\lfloor \frac{(q-1)p}{q} \right\rfloor \right\rangle$.

Proof: $\forall n \in (0,q), \left\lfloor \frac{n \cdot p}{q} \right\rfloor \cdot \frac{q}{p} \leq n, \left(\left\lfloor \frac{n \cdot p}{q} \right\rfloor + 1 \right) \cdot \frac{q}{p} > n.$

Let $n = 1$, then the second time slice must be $\left\lfloor \frac{p}{q} \right\rfloor$;

Let $n = 2$, then the third time slice must be $\left\lfloor \frac{2p}{q} \right\rfloor$;

... ...

Let $n = q-1$, then the q-th time slice must be $\left\lfloor \frac{(q-1)p}{q} \right\rfloor$. ■

Lemma 4.3 *The difference between two neighboring numbers in* $\mathcal{T}(p,q) \cup \langle p \rangle$ *is either* $\left\lfloor \frac{p}{q} \right\rfloor$ *or* $\left\lfloor \frac{p}{q} \right\rfloor + 1$.

Proof: $\forall n \in (0,q]$, the n-th number in $\mathcal{T}_0(p,q) \cup \langle p \rangle$ is $\left\lfloor \frac{(n-1) \cdot p}{q} \right\rfloor$, and the ($n$+1)-th one is $\left\lfloor \frac{n \cdot p}{q} \right\rfloor$, where

$\left\lfloor \frac{(n-1) \cdot p}{q} \right\rfloor + \left\lfloor \frac{p}{q} \right\rfloor \leq \left\lfloor \frac{n \cdot p}{q} \right\rfloor \leq \left\lfloor \frac{(n-1) \cdot p}{q} \right\rfloor + \left\lfloor \frac{p}{q} \right\rfloor + 1.$ ■

Corollary 4.2 *The distance between two neighboring time slices on a regular partition is either* $\left\lfloor \frac{p}{q} \right\rfloor$ *or* $\left\lfloor \frac{p}{q} \right\rfloor + 1$, *where* $\frac{q}{p}$ *is the availability factor of this partition.*

Lemma 4.4 $\mathcal{T}(p,r,\delta) \subset \mathcal{T}_0(p,q)$, *where* $q + r < p$ *and* $r < q < 2r$, *then*

$\exists n > 0$, where $\frac{q}{p} = \frac{2n+1}{4n+3}; \frac{r}{p} = \frac{n+1}{4n+3}$.

Proof: Let $d = \left\lfloor \frac{p}{q} \right\rfloor$, then the distance of any neighbor-pair in $\mathcal{T}_0(p,q)$ is either d or $d+1$.

• Case 1: $d \geq 3$

$d \leq \left\lfloor \frac{p}{r} \right\rfloor \leq \left\lfloor \frac{2p}{q} \right\rfloor \leq 2d+1$

Because $\mathcal{T}(p,r,\delta)$ is a sub-sequence of $\mathcal{T}_0(p,q)$, the distance of any neighbor-pair in it can only be chosen from either $\{d, d+1\}$ or $\{2d, 2d+1\}$ (since $2d > (d+1)+1$). The first case is impossible because there are a neighbor-pair separated by an element in $\mathcal{T}_0(p,q) - \mathcal{T}(p,r,\delta)$, the distance of which is not less than $2d$. In the second case, the elements in $\mathcal{T}_0(p,q)$ must be assigned to $\mathcal{T}(p,r,\delta)$ and $\mathcal{T}_0(p,q) - \mathcal{T}(p,r,\delta)$ alternately, which indicates $r = q - r$. Thus, $d \geq 3$ is impossible.

• Case 2: $d = 1$

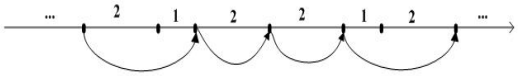
Suppose there are x (/ y) times neighbor-pairs, the distance of each of which is 1 (/ 2). Then $x + 2y = p, x + y = q \Rightarrow x = 2q - p, y = p - q$.

$\left\lfloor \frac{p}{q} \right\rfloor = d = 1; p > q + r; r < q < 2r$

$\Rightarrow \frac{1}{2}p < q < \frac{2}{3}p; \ \frac{1}{4}p < \frac{1}{2}q < r < 1 - q < \frac{1}{2}p$

$\Rightarrow$ there are no three consecutive integers in $\mathcal{T}_0(p,q) \cup \langle p \rangle$.

Case 2-1: If $\left\lfloor \frac{p}{r} \right\rfloor = 2$, the distance of any neighbor-pair in $\mathcal{T}(p,r,\delta) \cup \langle p \rangle$ is 2 or 3. Suppose there are x' (/ y') times neighbor-pairs, the distance of each of which is 3 (/ 2).



Because each 3-distance neighbor-pair in $\mathcal{T}(p,r,\delta)\cup\langle p\rangle$ corresponds to an 1-distance neighbor-pair in $\mathcal{T}_0(p,q)\cup\langle p\rangle$, we have $x'=x, y'=y-x \Rightarrow r=x'+y'=y \Rightarrow q+r=p$.

Case 2-2: If $\left\lfloor\frac{p}{r}\right\rfloor=3$, the distance of any neighbor-pair in $\mathcal{T}(p,r,\delta)\cup\langle p\rangle$ is 3 or 4. Because each neighbor-pair in $\mathcal{T}(p,r,\delta)\cup\langle p\rangle$ corresponds to at least two consecutive neighbor-pairs in $\mathcal{T}_0(p,q)\cup\langle p\rangle$, we have $q\geq 2r$.

Thus, $d=1$ is impossible.

• Case 3: $d=2$

Suppose there are x (/ y) times neighbor-pairs, the distance of each of which is 2 (/ 3). Then $2x+3y=p, x+y=q \Rightarrow x=3q-p, y=p-2q$.

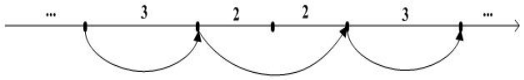
$\left\lfloor\frac{p}{q}\right\rfloor=d=2; p>q+r; r<q<2r$

$\Rightarrow \frac{1}{3}p<q<\frac{1}{2}p;\ \frac{1}{6}p<\frac{1}{2}q<r<q<\frac{1}{2}p$

Case 3-1: $\left\lfloor\frac{p}{r}\right\rfloor=2$ is impossible because there is a neighbor-pair in $\mathcal{T}(p,r,\delta)$ separated by a number in $\mathcal{T}_0(p,q)-\mathcal{T}(p,r,\delta)$, the distance of which is not less than $2d=4$.

Case 3-2: If $\left\lfloor\frac{p}{r}\right\rfloor=4$ or 5, the elements in $\mathcal{T}_0(p,q)$ must be assigned to $\mathcal{T}(p,r,\delta)$ and $\mathcal{T}_0(p,q)-\mathcal{T}(p,r,\delta)$ alternately, which indicates $r=q-r$.

Case 3-3: If $\left\lfloor\frac{p}{r}\right\rfloor=3$, the distance of any neighbor-pair in $\mathcal{T}(p,r,\delta)\cup\langle p\rangle$ is 3 or 4, and each 3-distance (/ 4-distance) neighbor-pair in $\mathcal{T}(p,r,\delta)\cup\langle p\rangle$ corresponds to a 3-distance neighbor-pair (/ two consecutive 2-distance neighbor-pairs) in $\mathcal{T}_0(p,q)\cup\langle p\rangle$. 4-distance neighbor-pair must exist in $\mathcal{T}(p,r,\delta)\cup\langle p\rangle$, otherwise, $r=q=\frac{p}{3}$.



Suppose there is a series of consecutive 2-distance neighbor-pairs between two nearest 3-distance neighbor-pairs in $\mathcal{T}_0(p,q)\cup\langle p\rangle$, whose number is $2n>0$, then $\frac{p}{q}\in\left(2+\frac{1}{2n+2}, 2+\frac{1}{2n}\right)$. Thus, the number of consecutive 2-distance neighbor-pairs between any two nearest 3-distance neighbor-pairs in $\mathcal{T}_0(p,q)\cup\langle p\rangle$ equals to $2n$. Hence, $\frac{q}{p}=\frac{2n+1}{4n+3}; \frac{r}{p}=\frac{n+1}{4n+3}$. ■

Corollary 4.3 *If $p\geq 3q$ and $r<q<2r$, $\mathcal{T}(p,r,\delta)\subset\mathcal{T}_0(p,q)$ is impossible.*

Theorem 4.3 $\Upsilon(\mathcal{B})\leq\frac{1}{2}$*, where $\mathcal{B}$ is any feasible ABS.*

Proof: Suppose $\mathcal{B}=\langle b_1,b_2,b_3,...\rangle$, where $b_1=\frac{q}{p}$ and p,q are co-prime. Then $\exists m,n>0, m\cdot p=n\cdot q+1$.

If $\Upsilon(\mathcal{B})>\frac{1}{2}$, by Theorem 4.2, $\forall i\geq 0, \frac{b_{i+1}}{b_i}>\frac{1}{2}$, where $b_0=1$, then $p\geq 3$ and $\exists j, b_j\in\left(\frac{1}{2p},\frac{1}{p}\right)$.

Suppose $b_j=\frac{q'}{p'}$, where p',q' are co-prime. Let $P=LCM(p,p')$, then $\mathcal{T}(P,\frac{q'\cdot P}{p'},\delta)\subset\mathcal{T}_0(P,\frac{P}{p})$ is impossible by Corollary 4.3. Thus, the sequence $\left\langle\frac{q}{p},\frac{q}{p},...,\frac{q}{p},\frac{q'}{p'}\right\rangle$ with $n+1$ elements is unschedulable under $\mathcal{B}$ on m resources, which means $\mathcal{B}$ is infeasible. ■

E. Feasible Approximating Boundary Sequences

Same as Problem II, the decision problem of the feasibility of an arbitrary ABS is very hard. Hereby we start from investigating several special types of ABSes. The first is the Geometric Approximating Boundary Sequence. The ABS of the AAF-Regular algorithm can belong in this type.

Definition 4.8 *A Geometric Approximating Boundary Sequence $\mathcal{G}(m)$ is specified as the infinite geometric sequence $\left\langle\frac{1}{m},\frac{1}{m^2},\frac{1}{m^3},...\right\rangle$ where integer $m>1$.*

Theorem 4.4 *$\forall m > 1, \mathcal{G}(m)$ is feasible.*

Proof: We use a similar approach as with AAF-Regular. ■

Another interesting type of ABS is the Arithmetic Approximating Boundary Sequence. This is a kind of finite ABS.

Definition 4.9 *An Arithmetic Approximating Boundary Sequence $\mathcal{A}(n)$ is specified as the finite arithmetic sequence $\left\langle \frac{n-1}{n}, \frac{n-2}{n}, ..., \frac{1}{n} \right\rangle$ where integer $n > 1$.*

To find feasible Arithmetic Approximating Boundary Sequences, we introduce the Regularity Magic Number (RMN). Regular partitions can be merged and split if their periods are all equal to a common RMN.

Definition 4.10 *Integer $\mathcal{P}$ is a Regularity Magic Number (RMN) if and only if $\forall q_1, q_2 \in (0, \mathcal{P})$ where $q_1 + q_2 \leq \mathcal{P}$, $\exists \delta_1, \delta_2 \in [0, \mathcal{P})$, where*

$\mathcal{T}_0(\mathcal{P}, q_1 + q_2) = \mathcal{T}(\mathcal{P}, q_1, \delta_1) \cup \mathcal{T}(\mathcal{P}, q_2, \delta_2)$.

Corollary 4.4 *Given a Regularity Magic Number $\mathcal{P}$, $\forall q_1, q_2$, where $0 < q_2 < q_1 \leq \mathcal{P}$, $\exists \delta_1, \delta_2 \in [0, \mathcal{P})$, where*

$\mathcal{T}_0(\mathcal{P}, q_1 - q_2) = \mathcal{T}(\mathcal{P}, q_1, \delta_1) - \mathcal{T}(\mathcal{P}, q_2, \delta_2)$.

Theorem 4.5 *The set of all Regularity Magic Numbers is $\{2, 3, 4, 5, 7\}$.*

Proof: We can check that 2, 3, 4, 5, 7 are RMNs. For example, 4 is an RMN since:

$\mathcal{T}_0(4, 1) = \langle 0 \rangle$;

$\mathcal{T}_0(4, 2) = \langle 0, 2 \rangle = \mathcal{T}_0(4, 1) \cup \mathcal{T}(4, 1, 2)$;

$\mathcal{T}_0(4, 3) = \langle 0, 1, 2 \rangle = \mathcal{T}(4, 1, 1) \cup \mathcal{T}_0(4, 2)$;

$\mathcal{T}_0(4, 4) = \langle 0, 1, 2, 3 \rangle = \mathcal{T}(4, 1, 3) \cup \mathcal{T}_0(4, 3)$;

$\mathcal{T}_0(4, 4) = \mathcal{T}_0(4, 2) \cup \mathcal{T}(4, 2, 1)$.

There is no other RMN since $\forall n \geq 6$ and $n \neq 7, \mathcal{T}(n, 2, \delta) \subset \mathcal{T}_0(n, 3)$ is impossible by Lemma 4.4. ■

Theorem 4.6 *$\mathcal{A}(n)$ is feasible if and only if n is a Regularity Magic Number.*

Proof: First $\mathcal{A}(n)$ is infeasible when n is not an RMN because $\left\langle \frac{n-3}{n}, \frac{2}{n} \right\rangle$ is unschedulable under $\mathcal{A}(n)$ on a single-resource ($\mathcal{T}(n, 2, \delta) \subset \mathcal{T}_0(n, 3)$ is impossible by Lemma 4.4).

Then we use an inductive approach to prove that $\mathcal{A}(n)$ is feasible when n is an RMN. First, $\mathcal{A}(n)$ is feasible for a single-resource by the definition of RMN. Suppose $\mathcal{A}(n)$ is feasible when the number of the resources is less than M.

Given $\left\langle \frac{q_1}{n}, \frac{q_2}{n}, ..., \frac{q_m}{n} \right\rangle$, where $\sum_{i=1}^{m} \frac{q_i}{n} = M$, as the availability factors of regular partitions $P_1, P_2, ..., P_m$. We will show that they are schedulable on M resources.

Find k, where $\sum_{i=1}^{k} \frac{q_i}{n} \leq 1$ and $\sum_{i=1}^{k+1} \frac{q_i}{n} > 1$.

If $\sum_{i=1}^{k} \frac{q_i}{n} = 1$, these partitions can be easily scheduled: $P_1, P_2, ..., P_k$ on one resource and $P_{k+1}, ..., P_m$ on the other $M-1$ resources.

Otherwise $\sum_{i=1}^{k} \frac{q_i}{n} < 1$ and $\sum_{i=1}^{k+1} \frac{q_i}{n} > 1$. Let $\frac{q}{n} = \sum_{i=1}^{k+1} \frac{q_i}{n} - 1 \in \left(0, \frac{q_{k+1}}{n}\right)$, then $\left\langle \frac{q}{n}, \frac{q_{k+2}}{n}, ..., \frac{q_m}{n} \right\rangle$ can be scheduled on $M-1$ resources. Suppose the time-slice sequence of the first partition is $\mathcal{T}(n, q, \delta)$. By Corollary 4.4, $\exists \delta_1, \delta_2$, where

$\mathcal{T}_0(n, q) = \mathcal{T}(n, q_{k+1}, \delta_1) - \mathcal{T}(n, q_{k+1} - q, \delta_2)$,

then

$\mathcal{T}(n, q, \delta) = \mathcal{T}(n, q_{k+1}, \delta_1 + \delta) - \mathcal{T}(n, q_{k+1} - q, \delta_2 + \delta)$.

At last, we just need to schedule $\left\langle \frac{q_1}{n}, \frac{q_2}{n}, ..., \frac{q_{k+1}-q}{n} \right\rangle$ on the left resource and let the last partition have the time-slice sequence $\mathcal{T}(n, q_{k+1} - q, \delta_2 + \delta)$. ■

Since the schedulability bound of an Arithmetic ABS is zero by Theorem 4.1, we define the third type of ABS – the Hybrid Approximating Boundary Sequence, which is infinite but can inherit the feasibility property from the Arithmetic ABS.

Definition 4.11 *A Hybrid Approximating Boundary Sequence $\mathcal{H}(n,m)$ is specified as the sequence $\langle \frac{n-1}{n}, \frac{n-2}{n}, ..., \frac{1}{n}, \frac{1}{n\cdot m}, \frac{1}{n\cdot m^2}, ...$ where integers $n>1,\ m>1$.*

Theorem 4.7 *$\mathcal{H}(n,m)$ is feasible if and only if n is a Regularity Magic Number.*

Proof Sketch: The scheduling has two steps. First schedule the elements in $\left\{1, \frac{n-1}{n}, \frac{n-2}{n}, ..., \frac{1}{n}\right\}$ using the strategy of scheduling $\mathcal{A}(n)$. Then schedule the other elements on the remaining part of the resources using the strategy of scheduling $\mathcal{G}(m)$. ■

$\mathcal{B}$	$\Upsilon(\mathcal{B})$	$\mathcal{O}(\mathcal{B})$	Feasibility
$\mathcal{G}(m)$	$\frac{1}{m}$	$\frac{1}{2}-\frac{1}{m+1}$	feasible
$\mathcal{A}(n)$	0	$\frac{1}{2n}$	feasible iif n is an RMN
$\mathcal{H}(n,m)$	$\frac{1}{m}$	$\frac{1}{2n}-\frac{1}{n^2(m+1)}$	feasible iif n is an RMN

TABLE I
A COMPARISON BETWEEN THREE TYPES OF ABS

Table I shows a comparison of the three types of ABS we have defined in this subsection. We can conclude from it that the smallest value of m and the largest value of n can lead to the best performance. Theorem 4.7 shows the maximum Regularity Magic Number is 7, so we find that the optimal ABS so far is $\mathcal{H}(7,2)$.

F. Extended Approximating Boundary Sequence

To achieve the optimal resource utilization, we extend the concept of Approximating Boundary Sequence.

Definition 4.12 *An Extended Approximating Boundary Sequence (E-ABS) is a tuple $(\mathcal{B},\mathcal{B}')$, where $\mathcal{B}=\langle b_1,b_2,b_3,...\rangle$, $\mathcal{B}'=\langle b'_1,b'_2,b'_3,...\rangle$ are two ABSes; $b_1+b'_1<1$ if neither $\mathcal{B}$ nor $\mathcal{B}'$ is empty.*

The boundary sequence of the upper E-ABS can be regarded as $\langle ...,1-b'_3,1-b'_2,1-b'_1,b_1,b_2,b_3,...\rangle$. An ABS $\mathcal{B}$ corresponds to an E-ABS $(\mathcal{B},\langle\rangle)$.

Definition 4.13 *The Approximating Function of an E-ABS $\mathcal{E}=(\mathcal{B}=\langle b_1,b_2,b_3,...\rangle,\mathcal{B}'=\langle b'_1,b'_2,b'_3,...\rangle)$ for $\alpha\in(0,1)$ is*

$$\mathcal{R}(\mathcal{E},\alpha)=\begin{cases}\mathcal{R}(\mathcal{B},\alpha) & if\ \alpha\le b;\\ 1-b'_{i+1}\ when\ \alpha\in(1-b'_i,1-b'_{i+1}] & otherwise,\end{cases}$$

where $b=b_1:0\ ?\ |\mathcal{B}|>0$; $b'_0=1-b$; $b'_{|\mathcal{B}'|+1}=0$ if $\mathcal{B}'$ is finite.

Same as a feasible ABS, $\Upsilon(\mathcal{E})$ is specified as the Schedulability Bound of a feasible E-ABS $\mathcal{E}$.

Theorem 4.8 *Given a feasible E-ABS $\mathcal{E}=(\mathcal{B}=\langle b_1,b_2,b_3,...\rangle,\mathcal{B}'=\langle b'_1,b'_2,b'_3,...\rangle)$,*

$\Upsilon(\mathcal{E})=\min\left\{\Upsilon(\mathcal{B}),\min\left\{\frac{1-b'_{i-1}}{1-b'_i}:i>0\right\}\right\}$,

where $b'_0=1-b_1:1\ ?\ |\mathcal{B}|>0$; $b'_{|\mathcal{B}'|+1}=0$ if $\mathcal{B}'$ is finite.

Definition 4.14 *The Utilization Overhead of an E-ABS $\mathcal{E}=(\mathcal{B}=\langle b_1,b_2,b_3,...\rangle,\mathcal{B}'=\langle b'_1,b'_2,b'_3,...\rangle)$ is*

$\mathcal{O}(\mathcal{E})=\frac{1}{2}(\sum_{i>0}(b_i-b_{i+1})^2+(1-b_1-b'_1)^2+\sum_{i>0}(b'_i-b'_{i+1})^2)$,

where $b_{|\mathcal{B}|+1}=0$ if $\mathcal{B}$ is finite; $b_{|\mathcal{B}'|+1}=0$ if $\mathcal{B}'$ is finite.

Corollary 4.5 *Given an E-ABS $\mathcal{E}=(\mathcal{B},\mathcal{B}')$ where neither $\mathcal{B}$ nor $\mathcal{B}'$ is non-empty,*

$\mathcal{O}(\mathcal{E})<\min\{\mathcal{O}(\mathcal{B}),\mathcal{O}(\mathcal{B}')\}$.

Theorem 4.9 *$(\mathcal{H}(n,m),\langle\frac{1}{n\cdot m},\frac{1}{n\cdot m^2},\frac{1}{n\cdot m^3},...\rangle)$ is feasible if and only if n is a Regularity Magic Number. $\mathcal{Z}(n,m)$ is specified as such an E-ABS.*

Proof: $\mathcal{Z}(n,m)$ is infeasible if n is not an RMN (by Theorem 4.6).

Then we use an inductive approach to prove that $\mathcal{Z}(n,m)$ is feasible if n is an RMN. First, $\mathcal{Z}(n,m)$ is feasible for a single-resource obviously. Suppose $\mathcal{Z}(n,m)$ is feasible when the number of the resources is less than M.

Given a non-increasing sequence $Q = \langle \alpha_1, \alpha_2, ..., \alpha_K \rangle$, where $\sum_{i=1}^{K} \alpha_i = M$ and $\alpha_i \in \mathcal{Z}(n,m) : i = 1, 2, ..., K$, as the availability factors of regular partitions $P_1, P_2, ..., P_K$, we will show that they are schedulable on M resources.

• if $\alpha_1 \leq \frac{n-1}{n}$, Q is schedulable by Theorem 4.7.

• if $\alpha_1 > \frac{n-1}{n}$ and $\alpha_i = \alpha_1 : i = 2, 3, ..., m$, suppose $\alpha_1 = 1 - \frac{1}{n \cdot m^e}$ where $e > 0$.

Let $\alpha = m \cdot \alpha_1 - (m-1) = 1 - \frac{1}{n \cdot m^{e-1}}$. Then $\langle \alpha, \alpha_{m+1}, \alpha_{m+2}, ..., \alpha_K \rangle$ is schedulable on $M-m+1$ resources. Suppose the time-slice sequence of the first partition is

$\mathcal{T}_0(n \cdot m^{e-1}, n \cdot m^{e-1}) - \mathcal{T}(n \cdot m^{e-1}, 1, \delta)$

At last, we just need to schedule $P_1, P_2, ..., P_m$ like this: for $i = 1, 2, ..., m$, the time-slice sequence of P_i is

$\mathcal{T}_0(n \cdot m^e, n \cdot m^e) - \mathcal{T}(n \cdot m^e, 1, \delta + (i-1) \cdot n \cdot m^{e-1})$.

• Otherwise, because $\sum_{i=1}^{K} \alpha_i = M$, $\exists k_1, k_2, ..., k_J, \sum_{i=1}^{J} \alpha_{k_i} = 1 - \alpha_1$.

Then we can schedule $P_1, P_{k_1}, P_{k_2}, ..., P_{k_J}$ on one resource, and the other partitions on the other $M-1$ resources. ■

Lemma 4.5 *If an RBS $\mathcal{B} = \langle b_1, b_2, b_3, ... \rangle$ is feasible, where $\Upsilon(\mathcal{B}) = \frac{1}{2}; b_k \geq \frac{1}{2}; \forall n > 0, b_k \neq 1 - \frac{2n+1}{4n+3}$, then $1 - b_k \in \mathcal{B}$; $\frac{1-b_k}{2} \in \mathcal{B}$; $\forall i, b_i \notin \left(\frac{1-b_k}{2}, 1 - b_k\right)$.*

Proof: If $\exists i, b_i \in \left(\frac{1-b_k}{2}, 1 - b_k\right)$, by Lemma 4.4, $\langle b_k, b_i \rangle$ is unschedulable under $\mathcal{B}$ on a single resource. Then $1 - b_k \in \mathcal{B}$ and $\frac{1-b_k}{2} \in \mathcal{B}$ because $\Upsilon(\mathcal{B}) = \frac{1}{2}$ (by Theorem 4.2). ■

Lemma 4.6 *If an RBS $\langle b_1, b_2, b_3, ... \rangle$ is feasible, where $\Upsilon(\mathcal{B}) = \frac{1}{2}; b_i > b_j \geq \frac{1}{2}; 1 - b_j < 2(1 - b_i)$, then $\exists n > 0$, $b_i = 1 - \frac{n+1}{4n+3}; b_j = 1 - \frac{2n+1}{4n+3}$.*

Proof:

• Case 1: $1 - b_i \in \mathcal{B}$

Because $1 - b_i < 1 - b_j < 2(1 - b_i)$ and $(1 - b_i) + (1 - b_j) < 1$, by Lemma 4.4, $1 - b_i = \frac{n+1}{4n+3}$ and $1 - b_j = \frac{2n+1}{4n+3}$ (otherwise, $\langle b_j, 1 - b_i \rangle$ is unschedulable under $\mathcal{B}$ on a single resource).

• Case 2: $1 - b_i \notin \mathcal{B}$

By Lemmas 4.4 and 4.5, $\exists m > 0, b_i = 1 - \frac{2m+1}{4m+3}$ and $\frac{m+1}{4m+3} \in \mathcal{B}$.

Because $\frac{m+1}{4m+3} < 1 - b_j \leq \frac{1}{2} < 2 \cdot \frac{m+1}{4m+3}$ and $1 - b_j \neq 1 - b_i = \frac{2m+1}{4m+3}$, by Lemma 4.4, $\langle b_j, \frac{m+1}{4m+3} \rangle$ is unschedulable under $\mathcal{B}$ on a single resource. ■

Lemma 4.7 *If an RBS $\langle b_1, b_2, b_3, ... \rangle$ is feasible, where $\Upsilon(\mathcal{B}) = \frac{1}{2}$, it is impossible that $\exists i, j, \frac{1}{3} \leq b_i < b_j \leq \frac{1}{2}$.*

Proof: Otherwise, $\langle b_i, b_j \rangle$ is unschedulable under $\mathcal{B}$ on a single resource by Lemma 4.4. ■

Definition 4.14 *Given $\alpha \in (0,1)$ and an E-ABS $\mathcal{E} = (\mathcal{B}, \mathcal{B}')$, $\alpha \in \mathcal{E}$ if and only if $\alpha \in \mathcal{B}$ or $1 - \alpha \in \mathcal{B}'$.*

Definition 4.15 *Given two E-ABSes $\mathcal{E}_1$ and $\mathcal{E}_2$, $\mathcal{E}_1 = \mathcal{E}_2$ if and only if $\forall \alpha \in \mathcal{E}_1, \alpha \in \mathcal{E}_2$ and $\forall \alpha \in \mathcal{E}_2, \alpha \in \mathcal{E}_1$.*

Theorem 4.10 *If a feasible E-RBS $\mathcal{E} \neq \mathcal{Z}(7,2)$ and $\Upsilon(\mathcal{E}) = \frac{1}{2}, \mathcal{O}(\mathcal{E}) > \mathcal{O}(\mathcal{Z}(7,2))$.*

Proof: $\mathcal{O}(\mathcal{Z}(7,2)) = \frac{17}{294} < 0.058$.

Suppose $\mathcal{E} = (\mathcal{B} = \langle b_1, b_2, b_3, ... \rangle, \mathcal{B}' = \langle b'_1, b'_2, b'_3, ... \rangle)$.

Since we can transfer elements between $\mathcal{B}$ and $\mathcal{B}'$, without loss of generality, we assume $\forall b_i \in \mathcal{B}, b_i < \frac{1}{2}$; $\forall b'_i \in \mathcal{B}', b'_i \leq \frac{1}{2}$.

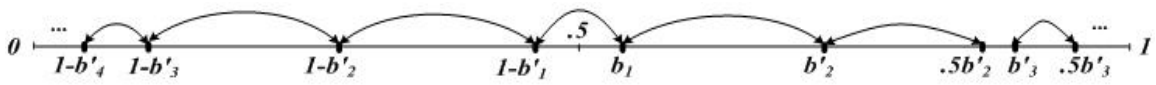
Case 1: $|\mathcal{B}'| \leq 1$

$\mathcal{O}(\mathcal{E}) \geq (\frac{1}{4})^2 > 0.06$.

Case 2: $|\mathcal{B}'| > 1$

• Case 2-1: $\exists i < j$, $b'_i < 2b'_j$

By Lemma 4.6, $b'_i = \frac{2n+1}{4n+3}; b'_j = \frac{n+1}{4n+3}$.
Still by Lemma 4.6, $i = 1; j = 2; \forall k \geq 2, b'_k \geq 2b'_{k+1}$.
By Lemma 4.5, $\forall k \geq 2, b'_k \in \mathcal{B}$; $\frac{b'_k}{2} \in \mathcal{B}$; $\forall l, b_l \notin \left(\frac{b'_k}{2}, b'_k\right)$.
By Lemma 4.4, if $b_l > \frac{n+1}{4n+3}, b_l = \frac{2n+1}{4n+3}$.



Thus, $2\mathcal{O}(\mathcal{E}) \geq \sum_{k\geq 2}(b'_k - b'_{k+1})^2 + (\frac{n}{4n+3})^2 + (\frac{1}{4n+3})^2 + (\frac{n}{4n+3})^2 + \sum_{k\geq 2}(b'_k - \frac{b'_k}{2})^2$.
Let $F(x)$ is the upper bound of $\sum_{k\geq 0}(x_k - x_{k+1})^2 + \frac{1}{4}\sum_{k\geq 0}(x_k)^2$, where $x_0 = x \in \left(0, \frac{1}{2}\right)$ and $\forall k \geq 0, x_{k+1} \in \left(0, \frac{x_k}{2}\right]$.
Then $\forall k \geq 0, F(x_k) = F(x_{k+1}) + (x_k - x_{k+1})^2 + \frac{(x_k)^2}{4}$ and $F(x_{k+1}) = (\frac{x_{k+1}}{x_k})^2 F(x_k)$
$\Rightarrow \frac{4}{x_k} + \frac{F(x_k)}{(x_k)^2} = \frac{9x_k - 8x_{k+1}}{(x_k + x_{k+1})(x_k - x_{k+1})}$, where $x_k \in \left(0, \frac{1}{2}\right)$ and $x_{k+1} \in \left(0, \frac{x_k}{2}\right]$
$\Rightarrow \forall k \geq 0, x_{k+1} = \frac{x_k}{2}$
$\Rightarrow F(x) = \frac{2}{3}x^2$.
Thus, $2\mathcal{O}(\mathcal{E}) \geq F(b'_2) + \frac{2n^2+1}{(4n+3)^2} = \frac{2}{3} \cdot (\frac{n+1}{4n+3})^2 + \frac{2n^2+1}{(4n+3)^2} = \frac{8n^2+4n+5}{3(4n+3)^2}$, where $n \geq 1 \Rightarrow \mathcal{O}(\mathcal{E}) \geq \frac{17}{294}$.
$\mathcal{O}(\mathcal{E})$ equals to $\frac{17}{294}$ only when $n = 1, b'_1 = \frac{3}{7}, b'_2 = \frac{2}{7}$ and $\mathcal{E} = \left(\left\langle \frac{3}{7}, \frac{2}{7}, \frac{1}{7}, \frac{1}{14}, \frac{1}{28}, ... \right\rangle, \left\langle \frac{3}{7}, \frac{2}{7}, \frac{1}{7}, \frac{1}{14}, \frac{1}{28}, ... \right\rangle\right) = \mathcal{Z}(7, 2)$.

• Case 2-2: $\forall i < j$, $b'_i \geq 2b'_j$
By Lemma 4.5, $\forall k \geq 1, b'_k \in \mathcal{B}$; $\frac{b'_k}{2} \in \mathcal{B}$; $\forall l, b_l \notin \left(\frac{b'_k}{2}, b'_k\right)$.

Case 2-2-1: $b'_1 \leq \frac{1}{3}$
By Lemma 4.7, There is at most one element of $\mathcal{B}$ in $\left(\frac{1}{3}, \frac{1}{2}\right)$.
Then: $2\mathcal{O}(\mathcal{E}) > F(b'_1) + (1 - b'_1 - \frac{1}{2})^2 + (\frac{1}{2} - \frac{1}{3})^2 = \frac{2}{3}(b'_1)^2 + (b'_1)^2 - b'_1 + \frac{5}{18} \geq \frac{23}{180} > 0.127$.

Case 2-2-2: $b'_1 > \frac{1}{3}$
By Lemma 4.7, b'_1 is the only element of $\mathcal{B}$ in $\left(\frac{1}{3}, \frac{1}{2}\right]$.
Then: $2\mathcal{O}(\mathcal{E}) \geq F(b'_1) + (1 - b'_1 - b'_1)^2 = \frac{2}{3}(b'_1)^2 + (1 - 2b'_1)^2 \geq \frac{1}{7} > 0.142$. ■

Finally, we derive that $\mathcal{Z}(7, 2)$ is the optimal E-ABS among those which keep the maximum schedulability bound. Magic7 is named for the Static Approximation Algorithm with $\mathcal{Z}(7, 2)$, whose boundary sequence can be regarded as $\left\langle ..., \frac{55}{56}, \frac{27}{28}, \frac{13}{14}, \frac{6}{7}, \frac{5}{7}, \frac{4}{7}, \frac{3}{7}, \frac{2}{7}, \frac{1}{7}, \frac{1}{14}, \frac{1}{28}, \frac{1}{56}, ... \right\rangle$. Table II compares Magic7 and AAF-Regular.

	$\mathcal{B}$	$\Upsilon(\mathcal{B})$	$\mathcal{O}(\mathcal{B})$	Average Utilization
AAF-Regular	$\mathcal{G}(2)$	0.5	0.167	75%
Magic7	$\mathcal{Z}(7, 2)$	0.5	0.058	89.6%

TABLE II
A COMPARISON BETWEEN MAGIC7 AND AAF-REGULAR

G. Magic7-Single and Magic7-Pfair-Mixed

Magic7 can be incorporated into the solutions for Problem I which schedules both regular and irregular partitions. Magic7-Single is an enhanced version of AAF-Single, whose Approximating Function is $\mathcal{AAF}'(\alpha, k)$.

$$\mathcal{AAF}'(\alpha,k)=\begin{cases}0 & if\ \alpha=0;\\ \frac{1}{7\cdot 2^n},\ where\ n=\left\lfloor \log_{\frac{1}{2}} 7\alpha \right\rfloor & if\ k{=}1, \alpha{\in}\left(0,\frac{1}{7}\right);\\ \frac{\lceil 7\alpha \rceil}{7} & if\ k{=}1, \alpha{\in}\left[\frac{1}{7},\frac{6}{7}\right];\\ 1-\frac{1}{7\cdot 2^n},\ where\ n=\left\lceil \log_{\frac{1}{2}} 7(1-\alpha) \right\rceil & if\ k{=}1, \alpha{\in}\left(\frac{6}{7},1\right);\\ \mathcal{AAF}'(\alpha-\mathcal{L}'(\alpha),k-1)+\mathcal{L}'(\alpha) & otherwise,\end{cases}$$

$$where\ \mathcal{L}'(\alpha)=\begin{cases}\frac{1}{7\cdot 2^n},\ where\ n{=}\left\lceil \log_{\frac{1}{2}} 7\alpha \right\rceil & if\ \alpha{\in}\left(0,\frac{1}{7}\right);\\ \frac{\lfloor 7\alpha \rfloor}{7} & if\ \alpha{\in}\left[\frac{1}{7},\frac{6}{7}\right];\\ 1-\frac{1}{7\cdot 2^n},\ where\ n{=}\left\lfloor \log_{\frac{1}{2}} 7(1-\alpha) \right\rfloor & if\ \alpha{\in}\left(\frac{6}{7},1\right);\\ 1 & if\ \alpha=1.\end{cases}$$

Correspondingly, Magic7-Pfair-Mixed is an enhanced version of AAF-Pfair-Mixed, as shown in Figure 8. Besides improving the resource utilization, Magic7-Pfair-Mixed also guarantees that the number of the migrating partitions is less than that of the resources, and each migrating partition preempts time slices on at most two resources.

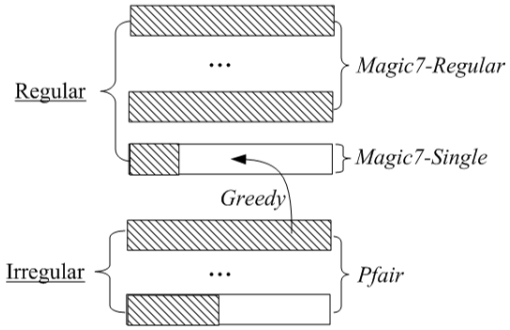


Fig. 8. Magic7-Pfair-Mixed

V. Simulation

We generate regular or irregular partitions whose availability factors are randomly chosen in $(0,1)$, and supply regularities are randomly chosen in $[1, MaxReg]$, where $MaxReg$ is a varying parameter. In Figures 9–11, the horizontal axes represent the utilization of the randomly generated partition sets, and the vertical axes represent the schedulability of these partition sets by different algorithms. From the charts, we can see that the Magic7-enhanced algorithm improves the resource utilization remarkably in each scenario.

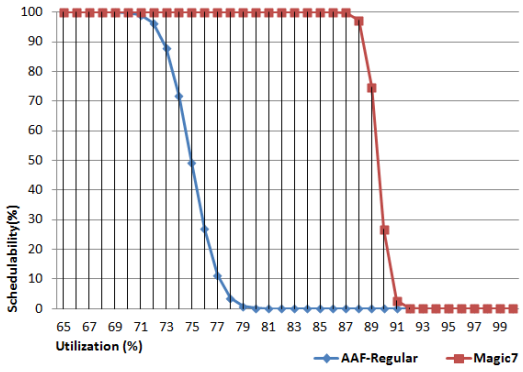


Fig. 9. 64 Resources, MaxReg=1

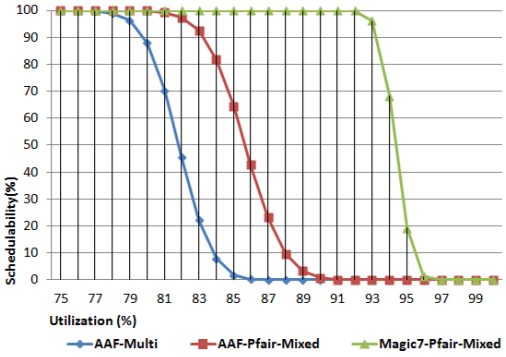


Fig. 10. 64 Resources, MaxReg=2

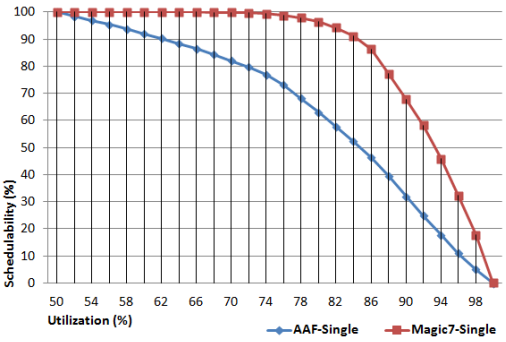


Fig. 11. Single Resource, MaxReg=2

VI. Related work

The concept of regularity originated from Shigero et al. In [7], they investigated the scheduling problem of regular partitions (our Problem II) for the first time, but only two very restricted cases on a single-resource platform were checked. Mok and Feng [1, 3] introduced the irregular partition in their Regularity-based Resource Partition Model. Furthermore, they presented an AAF-based approximation algorithm to schedule regular and irregular partitions on a single-resource platform. Recently, Li and Cheng [4] extended the AAF-based algorithm onto a multi-resource platform. However, all these AAF-based algorithms works only with a specific approximating function. In this paper, we investigate general approximating functions, and derive an optimal approximation algorithm along with a formal correctness proof.

Classic real-time scheduling algorithms (such as Rate Monotonic and Earliest Deadline First in [8]) are always concerned with the execution time, period and deadline specifications of the real-time tasks. They cannot directly be applied to our regularity-based resource partitioning problems since they cannot guarantee the regularity bounds of the resource partitions. The Pfair algorithm [4, 5] is one of the few exceptions. However, it can only generalize irregular partitions.

There is a large collection of multiprocessor scheduling and hierarchical scheduling techniques. Some of them share similarities with our approach more or less. For example, Semi-partitioned scheduling [9, 10] and cluster-based scheduling [11] also combine the use of partitioned and global scheduling to achieve higher resource utilization. However, none of them can be used for scheduling regularity-based partitions because of the tight temporal restrictions on regular partitions.

VII. Conclusion

In this paper, we investigate the resource partitioning problems (Problem I in Sect. II, Problem II in Sect. IV) in the Regularity-based Resource Partition Model. Our contributions include:

• We present a hybrid approach which separates the two types of regularity-based partitions, regular and irregular, for assignment to different resources. With it, Problem II is decomposed into two subproblems. One is the problem of scheduling irregular partitions and the other is Problem I.

• We mitigate the resource wasting problem due to the partitioned strategy.

• We demonstrate that the Pfair algorithm can be used to schedule irregular partitions.

• We introduce a category of resource partitioning algorithms – Static Approximation Algorithms – for solving Problem II. We prove that the schedulability bound of a feasible Static Approximation Algorithm is at most 0.5, and derive the one with the highest resource utilization (called Magic7) while keeping this maximum schedulability bound.

• Since there is no partition-migration problem on a single-resource platform, our Magic7-enhanced algorithms perform perfectly in this scenario.

There are still many open issues to be investigated. For example, Dynamic Approximation Algorithms could be studied in future. Another, the partition-migration overhead could be taken more into account in the multiresource scenarios.

REFERENCES

[1] A. K. Mok and X. Feng. Towards compositionality in real-time resource partitioning based on regularity bounds. RTSS, 2001.
[2] A. K. Mok, X. Feng, and D. Chen. Resource partition for real-time systems. RTAS, 2001.
[3] X. Feng. Design of real-time virtual resource architecture for largescale embedded systems. Ph.D. dissertation, Department of Computer Science, The University of Texas at Austin, 2004.
[4] Y. Li, A. M. K. Cheng, A. K. Mok. Regularity-based Partitioning of Uniform Resources in Real-time Systems. Submmited to RTCSA 2012.
[5] S. Baruah, N. Cohen, G. Plaxton, and D. Varvel. Proportionate progress: A notion of fairness in resource allocation. Algorithmica, 1996.
[6] S. Baruah, J. Gehrke, and G. Plaxton. Fast scheduling of periodic tasks on multiple resources. IPPS, 1995.
[7] S. Shigero, M. Takashi, and H. Kei. On the schedulability conditions on partial time slots. RTCSA, 1999.
[8] C. L. Liu and J. W. Layland. Scheduling algorithms for multiprogramming environment in a hard real-time environment. J. ACM, 20(1):46-61, 1973.
[9] B. Andersson and K. Bletsas. Sporadic multiprocessor scheduling with few preemptions. ECRTS, 2008.
[10] S. Kato and N. Yamasaki. Semi-Partitioned Fixed-Priority Scheduling on Multiprocessors. RTAS, 2009.
[11] I. Shin, A. Easwaran, and I. Lee. Hierarchical scheduling framework for virtual clustering of multiprocessors. ECRTS, 2008.