Schedulability Analysis for the Abort-and-Restart Model

Hing Choi Wong

Doctor of Philosophy

University of York
Computer Science

December 2014

Abstract

In real-time systems, a schedulable task-set guarantees that all tasks complete before their deadlines. In functional programming, atomic execution provides the correctness of the program. Priority-based functional reactive programming (P-FRP) allows the usage of functional programming in the real-time system environment. The abort-and-restart (AR) is a scheme to implement P-FRP but an appropriate scheduling approach does not exist at the moment. Hence, efficient analysis is needed for the AR model.

In this thesis, the schedulability analysis for the AR model is introduced and it shows that finding the critical instant for the AR model with periodic and sporadic tasks is intractable, and a new formulation is derived. Afterwards, a new priority assignment scheme is developed that has the performance close to the exhaustive search method, which is intractable for large systems. The technique of deferred preemption is employed and a new model, *deferred abort* (DA), provides better schedulability and dominates the non-preemptive model. Lastly, a tighter analysis is introduced and the technique of the multi-set approach from the analysis of cache related preemption delay is employed to introduce a new approach, *multi-bag*. The multi-bag approach can apply to both the AR model and the DA model. In the experiments, the schedulability of the AR model is improved at each stage of the research in this thesis.

Contents

List of Figures

List of Tables

List of Algorithms

Acknowledgements

Foremost, I would like to express my sincere gratitude to my supervisor Professor Alan Burns for his patience, motivation, enthusiasm, and immense knowledge. His guidance helped me in all the time of research and writing of this thesis.

I am extremely grateful to my family who have given me encouragement and love; they have also offered me financial support.

I would like to express my thanks to Dr Robert Davis for his insightful comments and discussions.

Last, I would also like to thank all my friends in York; they have given me a wonderful time and encouragement.

Declaration

I declare that the research work presented in this thesis is an independent and original contribution, except where explicitly cited. This work has not previously been presented for an award at this, or any other, University. The chapters of the thesis are based on the following publications.

Chapters 3 and 4 are based on a conference paper, a workshop paper and a technical report.

H.C. Wong and A. Burns. Schedulability Analysis for the Abort-and-Restart (AR) Model. In *Proceedings RTNS*. ACM, 2014

H.C. Wong and A. Burns. Improved Priority Assignment for the Abort-and-Restart (AR) Model. Technical Report YCS-2013-481, University of York, Department of Computer Science, 2013

H.C. Wong and A. Burns. Improved Priority Assignment for the Abort-and-Restart (AR) Model. In *Proceedings JRWRTC (RTNS)*, 2013

Chapter 6 is partially based on a conference paper.

H.C. Wong and A. Burns. Schedulability Analysis for the Abort-and-Restart (AR) Model. In *Proceedings RTNS*. ACM, 2014

Chapter 1

Introduction

In this chapter, the definition of real-time systems, priority-based functional reactive programming and abort-and-restart model are introduced. Motivation, thesis proposition, contributions and thesis structure are given. The focus of the work is on single processor systems.

1.1 Real-Time Systems

Many definitions of real-time systems are given by different authors. In Burns and Wellings' book [18] and the paper given by Stankovic [77], they define that a real-time system has to respond to an environment according to the received information within a certain time. Both the logical result and the delivery time are important. A failure to respond before the deadline is as bad as an incorrect response.

In Krishna's book [51], the author finds it difficult to provide a precise and cogent statement, but he implies that any system where a timely response by the computer to external stimuli is vital is a real-time system.

Lastly, the other definition found in the paper given by Joseph and Pandya [47] is that a number of devices are linked to a real-time system. The system runs a real-time program to read inputs frequently from devices, processes these inputs immediately, and then finally responds to the devices. It is a typical real-time system but not all devices operate in this way.

To summarise, a real-time system is a timely response system which receives inputs and then responds to the devices. Most devices produce inputs frequently; some devices have minimum and maximum times between their inputs, and others generate inputs at unknown times. The system should respond to all inputs before deadlines, otherwise it is a system failure.

1.2 Priority-based Functional Reactive Programming

Priority-based functional reactive programming (P-FRP) has been introduced as a new functional programming scheme [7] for real-time systems. It is an extension of functional reactive programming (FRP), which is a declarative programming paradigm [40, 81] that has two properties, behaviours and events. Behaviours are time-varying reactive values and events are discretely time-ordered. An FRP program [48] consists of repeated behaviours and event definitions. FRP has been used for setting up various reactive systems in the field of computer animation [34, 35, 72], computer vision [71], graphical user interface design [25] and robotics [65, 66].

Before P-FRP, the event-driven FRP (E-FRP) [82] was developed for real-time systems. Any E-FRP program needs to reply to every event by the operation of its handler, finish the operation of each handler, and operate in limited space and time. The problem is that each handler must finish the current execution before another event comes up. Even when a more urgent event occurs, it still needs to wait until the job of the current handler is completed. In the paper given by Kaiabachev et al. [48], an improved compilation strategy for E-FRP was developed and they named it P-FRP, which gives programmers more control over the execution strategy of events.

Therefore P-FRP aims to improve the programming of reactive real-time systems. It has the properties of lock-free shared resources and the priority policy. Lock-free shared resources means that each handler executes atomically and no resource is blocked by another handler. Programmers are able to assign higher priorities for urgent events. According to the characteristic of P-FRP, a new implementation scheme is required to solve this problem.

1.3 Abort-and-Restart Model

Abort-and-restart (AR) is a scheme [48] to support P-FRP. To achieve the properties of P-FRP, higher priority tasks can preempt lower priority tasks, and the lower priority tasks are aborted and restarted after the higher priority tasks have finished execution. In the classical preemptive model, the lower priority tasks continue their execution but it is different for P-FRP; the lower priority tasks restart as new. AR is the key operation for P-FRP so we call it the AR model in this thesis.

In the AR model, tasks cannot access resources directly. Rather, tasks make copies of the resource at the beginning of their execution. The updated data is then copied back into the system once the tasks have completed their execution. In some situations, higher priority tasks preempt lower priority tasks. Once the higher priority tasks have completed execution, the lower priority tasks are aborted and restarted. The operation of AR is to delete the old copy of the resource, and take a new copy from the system.

The classical preemptive model must deal with the problem of resource sharing. These problems can bring serious consequences. They may lead to inaccurate data, missed deadlines or deadlock. To cater for these problems various forms of priority inheritance and priority ceiling protocols have been developed [49, 50, 68, 75, 79]. One advantage of the AR model is that it does not face these problems because tasks do not access resources directly or concurrently. The disadvantage is that aborted tasks delete the old copy of the resource and restart as new, hence the time spent before preemption is wasted. In this thesis, we call this wasted time the *abort cost*.

1.3.1 Copy-and-Restore Operation

The AR model deals with resources by the Copy-and-Restore operation [7, 9, 10, 11], which occurs when tasks begin or restart execution, and they get a copy of the current state from the system. We call the copy *scratch state*, which is actually a set of data which will be used during the execution of the task. Tasks only change their copy so no tasks lock the data resource. If higher priority tasks arrive, the lower priority task discards its copy. Once the higher priority tasks have completed execution, the lower priority tasks are aborted and restarted. When a task has finished, the copy is restored into the system as an atomic action; this is illustrated in Figure 1.1 where τ_1 starts at time 0 and copies a set of data from the system. After six ticks, its job is done and then it restores the updated data into the system.

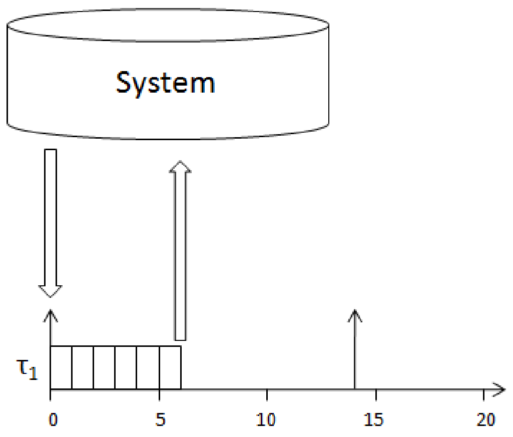

Figure 1.1: Copy-and-Restore Operation.

1.4 Motivation

Nowadays, computers have more power of execution than before. In concurrent programming, sometimes programmers consider how to enhance the correctness of programs rather than reduce the overhead. For a real-time system, it is more complicated because of timing constraints and priorities. A concurrency control mechanism for a system is important because it affects correctness and schedulability.

The AR model provides strong correctness guarantees in dealing with shared resources. And it also supports FRP which has been used for the domains of computer animation, computer vision, robotics and control systems [48]. Original FRP cannot be used for real-time systems but P-FRP has rectified this. Hence the AR model allows P-FRP to be used for real-time systems.

A real-time database system can be simply defined as a database system with timing constraints [64]. The system receives a high demand of requests and the responses are required to be sent out before their deadlines. A transactional memory can handle shared resource in a convenient way, and the AR scheme can be applied to it. For example, all transactions must be consistent and up to date for the stock market. The AR model has the properties of atomic execution and preemption. A transaction will not conflict with another transaction, and an urgent transaction can be executed immediately.

Real-time Java is designed to allow programmers to develop a real-time application using the Java language. The paper given by Manson et al [62] introduced the example of Preemptible Atomic Regions (PAR) for real-time Java. It is a new concurrency control abstraction for real-time systems. The basic notions of the AR model and the PAR model are similar but PAR makes a log of shared resource and then the state of resource will be rolled back if the task is preempted.

1.5 Thesis Proposition and Contributions

The central proposition of this thesis is:

While the abort-and-restart (AR) model can deal effectively with P-FRP in terms of the problems of resource usage, applicable schedulability analysis has not been demonstrated for this model. This thesis contends that it is possible to derive an appropriate scheduling approach for the model.

The research contributions of this thesis are:

Critical Instant — Demonstrating the critical instant for the AR model with periodic and sporadic tasks is intractable.

New formulation for scheduling — This is introduced and can be applied to the standard response time analysis for the AR model.

New priority assignment schemes — New priority assignment schemes are developed for both the AR and deferred abort (DA) models, and they have good performance and are tractable for large systems.

Deferred Abort (DA) model — This model provides better schedulability and dominates both the AR and non-preemptive models.

Multi-bag approach — This approach offers a tighter analysis on scheduling task-sets under both AR and DA models.

1.6 Thesis Structure

This thesis consists of eight chapters. In this chapter, a general introduction and overview of the area of research is given. The motivation, thesis proposition and contribution were discussed above. The structure of each remaining chapter is as follows:

Chapter 2 introduces the system model and reviews the related work of the AR model.

Chapter 3 analyses the schedulability for the AR model. The results show how critical instants occur in AR scheduling systems. A new formulation is developed for the AR schedulability test.

Chapter 4 improves the priority assignment for the AR model. A new algorithm is introduced and the time complexity is discussed to show the improvement. Lastly, an experimental evaluation is undertaken.

Chapter 5 presents a new idea, *deferred abort*, to reduce the number of aborts. The approach of DA is analysed and an experimental evaluation shows it is effective.

Chapter 6 provides a tighter analysis for the AR model, and presents a new approach, *multi-bag*. Again, an experimental evaluation is reported.

Chapter 7 applies the multi-bag approach to the DA model. Again, an experimental evaluation is reported.

Chapter 8 concludes the entire thesis and discusses future work.

Chapter 2

Related work

In this chapter, some literature related to the work of this thesis is introduced. Firstly, the system model used in this thesis is described. Secondly, the real-time scheduling is introduced. Lastly, the existing techniques and analysis of the abort-and-restart (AR) model are reviewed.

2.1 System Model

The system model is built on the fixed priority scheduling of a set of sporadic tasks on a single processor. Each task consists of a potentially unbounded sequence of jobs.

In general we allow constrained deadlines, although previous work and many of the examples in this thesis have implicit deadlines. We restrict implementation to single processor systems.

2.2 Real-Time System Scheduling

Firstly, we review the basic concept of real-time system scheduling, which is organised into five subsections: 1) Characteristics of tasks, which introduces worst-case execution time, deadline, release offsets, preemption, non-preemption, blocking, deferred preemption and preemption costs. 2) Critical instant, which is important when doing exact analysis. 3) A review of priority assignments, which consists of rate monotonic, deadline monotonic and Audsley's algorithm. 4) Schedulability analysis, which introduces utilisation based analysis and response time analysis. 5) The solutions of the shared resources problem, which are non-preemptive protocol, priority inheritance protocol, priority ceiling protocol and AR.

2.2.1 Characteristics of Tasks

A real-time system is connected to a number of devices which generate inputs to the real-time program. When scheduling a real-time system, we have terms such as task-sets, tasks and jobs. A task-set has a number of tasks. A task has period, deadline, worst-case execution time, etc. A job is some computation of a task. We know that tasks may operate in different ways so there are different types: periodic, aperiodic and sporadic.

A periodic task [12, 30, 43, 58] arrives into the system with a constant inter-arrival time, and has a relative deadline. An aperiodic task [22, 36, 38, 42, 43, 56, 57] has non periodic arrival time and a relative deadline. A sporadic task [41, 43, 45, 76] can arrive into the system at any time within defined minimum inter-arrival times between two consecutive releases. We mostly consider only periodic tasks in this thesis.

Worst-Case Execution Time

Each task has an execution time which depends on the task design and the hardware environment. When scheduling a real-time system, we need to analyse the execution times of tasks. According to the actual execution time, which may vary on each release, calculating the WCET [67] is required to ensure that tasks meet deadlines. We have to be concerned with a WCET that is either optimistic or excessively pessimistic when doing the time estimation. There is considerable literature on WCET analysis, but it is not reviewed in this thesis.

Deadline

In real-time systems, we always consider deadlines because a deadline miss may cause a critical system failure. A hard deadline [21, 51, 80, 87] is such that a task must meet the deadline otherwise the system is deemed to have failed. A soft deadline [56] is such that a task may miss the deadline sometimes but it causes no harm. We are concerned with hard deadlines in this thesis.

Release offsets

To improve the flexibility, tasks can be assigned offsets for their first releases. In Section 2.3.6, there is an example for release offsets shown in Table 2.18 and Figure 2.37.

Preemption and non-preemption

A preemption [27, 62, 73, 74] in real-time systems means that a lower priority task is paused by the arrival of a higher priority task. The lower priority task only continues to execute if the higher priority task is completed and no other higher priority task is waiting. In Figure 2.1, the diagram shows that the high priority task has two ticks to execute and the low priority task has four ticks. The low priority task is stopped after two ticks because the high priority task arrives. After the high priority task completes its two ticks of work, the low priority task does the rest of its execution. The advantage is that if the task is urgent, it can be assigned a higher priority. But there are further issues [44, 49, 50]; e.g. overheads for context switching [53, 54], blocking and interference.

The non-preemptive model [23, 37, 39, 46] is such that once a task begins its execution, it will not be paused by any tasks. To illustrate the idea, the diagram also shows in Figure 2.1, that the high priority task is postponed in its execution because the system is used by the low priority task. The advantage of non-preemption is that shared resources do not need to be locked [1, 33, 88].

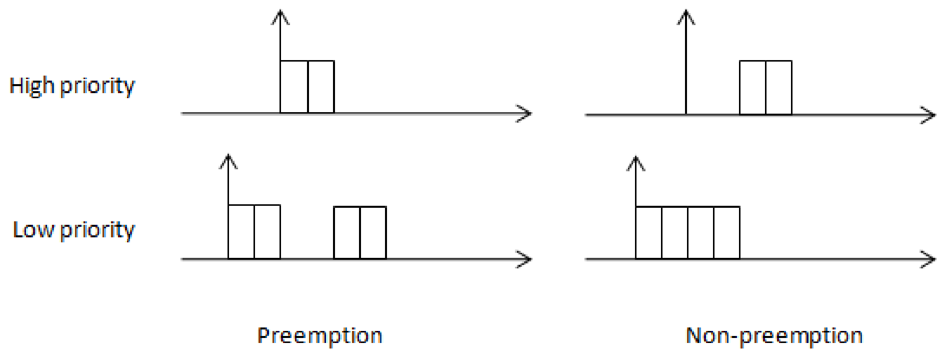

Figure 2.1: A diagram for preemption and non-preemption

Blocking

A blocking [14] can occur when a non-preemptive task τ_j with lower priority is released before a high priority task τ_i, or a lower priority task τ_j locks the resource which is also needed by another higher priority task τ_i.

Deferred preemption

Fixed priority scheduling with deferred preemption (FPDS) [15] allows that a lower priority task turns to non-preemptive when it is almost completed. A threshold for each task is set to prevent preemption from other higher priority tasks.

Preemption costs

In the analysis of cache related preemption delays (CRPD) [2], there are different amounts of overheads for preemptions because the time spent on context switching depends on the complexity of the task. It is similar to the abort cost.

2.2.2 Critical Instant

A critical instant for a task means the time at which a release of that task will lead to the greatest response time. The motivation of critical instant is that when scheduling a task-set, we need to find out the worst-case response time to check if the task-set is schedulable in the system. In fixed priority scheduling with the classical preemptive model, Liu and Layland [59] proved that a task is at its critical instant when the task is released with releases for all higher priority tasks at the same time.

Task	Period	WCET	Priority
τ_1	5	1	1
τ_2	10	2	2
τ_3	12	3	3

Table 2.1: A task-set with constrained deadlines

In Table 2.1, the task-set has three tasks: τ_1, τ_2 and τ_3. The highest priority task is τ_1 because of the smallest number for its priority. τ_2 is a medium priority task and τ_3 is the lowest priority task.

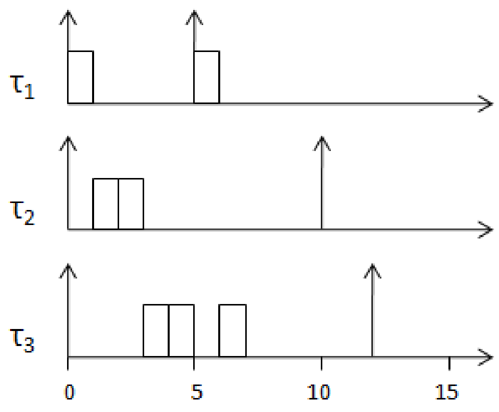

Figure 2.2: All tasks release together.

To illustrate the idea of critical instant, Figure 2.2 shows all tasks released together. The lowest priority task, τ_3, is ineffectively preempted by τ_1 and τ_2 so τ_3 starts at 3. The second job of τ_1 is arrived at 5 then τ_3 is preempted again. As we know [59] that a synchronous release leads to a critical instant, the worst-case response time for τ_3 is 7.

Least Common Multiple

In real-time systems, the response times of periodic task-sets can be predicted by using the Least common multiple (LCM) approach. There is an example in Table 2.16 and Figure 2.33. The periods for each task in the task-set are 4, 5 and 10, therefore the LCM is 20. If all tasks meet their deadlines within this period, the task-set is schedulable. There are difficulties in using or even computing LCM if there are a large number of tasks, or periods are unknown and arrival of tasks is unpredictable.

2.2.3 Priority Assignment

In a task-set, tasks are assigned with either static or dynamic priorities to deal with the ordering of execution in the real-time system. A bigger integer number is usually represented as a higher priority in programming. For most academic papers, a smaller integer number is represented as a higher priority so this thesis follows this style for all examples. If more than one task is released at the same time, a higher priority task executes first. In a classical preemptive real-time system, a higher priority task can execute immediately, even if a lower priority task is being executed. In Figure 2.3, the WCET of all tasks is 2 ticks. The high priority task executes first. The second is the medium priority task. The last is the low priority task.

Rate Monotonic Priority Assignment

Rate monotonic (RM) priority assignment [55] is such that priorities are assigned to tasks according to their periods; a shorter period task has a higher priority, as shown in Equation (2.1). In fixed priority scheduling with the classical preemptive model, RM priority assignment is optimal[1] [59] when the task-set has implicit deadlines. Ties are broken arbitrarily.

$$T_i < T_j \Rightarrow P_i > P_j, for D = T \tag{2.1}$$

[1]The definition of optimal is that if any other static priority assignment algorithm can meet all the deadlines then RM scheduling can also.

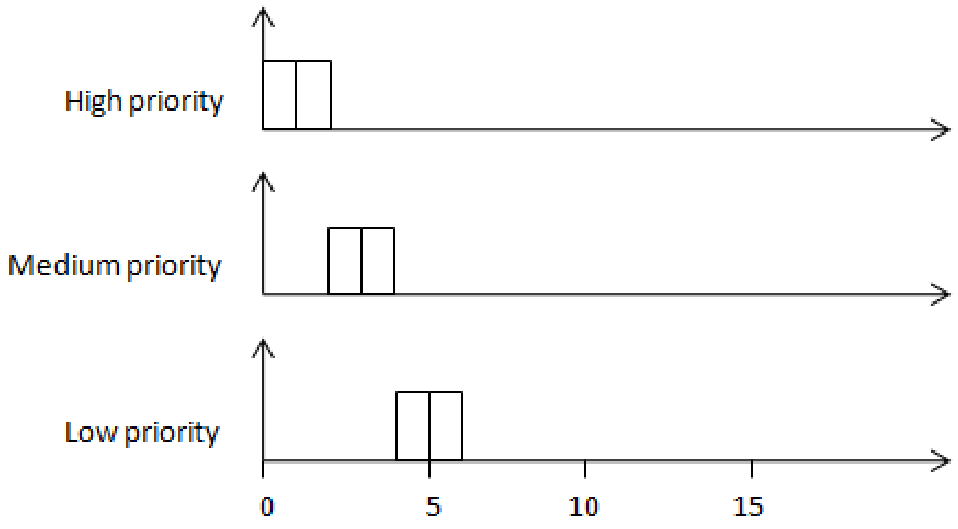

Figure 2.3: All three tasks have 2 ticks for their WCET.

To illustrate, we consider the task-set in Table 2.2 which has five tasks without priorities.

Task	Period	WCET
τ_1	40	2
τ_2	25	2
τ_3	10	3
τ_4	12	2
τ_5	16	3

Table 2.2: A task-set without priorities.

In Table 2.3 priorities are added after RM priority assignment. τ_3 is the highest priority task because of the smallest period and τ_1 is the lowest priority task because it has the largest period.

Deadline Monotonic Priority Assignment

Deadline monotonic (DM) [18] is similar to RM but it is based on relative deadlines rather than periods. In fixed priority scheduling with the classical preemptive model, it is an optimal priority assignment when deadlines of

Task	Period	WCET	Priority
τ_1	40	2	5
τ_2	25	2	4
τ_3	10	3	1
τ_4	12	2	2
τ_5	16	3	3

Table 2.3: A task-set with priorities. (The highest priority is 1)

tasks are less than or equal to (constrained deadlines) periods of tasks. In an actual task-set table, there should be a column, deadline, but the column of deadline is removed in many of our examples because we assume that period is equal to deadline. In Table 2.4, deadlines of tasks are less than periods of tasks. τ_3 is assigned the highest priority although τ_4 has a smaller period.

Task	Period	Deadline	WCET	Priority
τ_1	50	40	2	5
τ_2	60	25	2	4
τ_3	30	15	3	1
τ_4	20	16	2	2
τ_5	25	20	3	3

Table 2.4: A task-set with constrained deadlines. (The highest priority is 1)

For arbitrary deadline task-sets with potentially some tasks having $D > T$ neither the RM nor the DM scheme can be used; instead Audsley's algorithm must be applied.

Audsley's algorithm

Audsley's algorithm furnishes an optimal priority assignment if an optimal schedulability test exists for that model; i.e. the algorithm can find a schedulable priority ordering if such an ordering exists [5, 6]. In Figure 2.4, there is a 5-tasks task-set and the algorithm starts with fitting the lowest priority slot. Assume τ_1 is assigned the lowest priority and other tasks are assigned higher priorities. If τ_1 is schedulable, τ_1 stays at that slot. Otherwise another task is assigned the lowest priority until a schedulable task is found, or the task-set is deemed unschedulable. After a schedulable task for each slot is found, the task-set is schedulable.

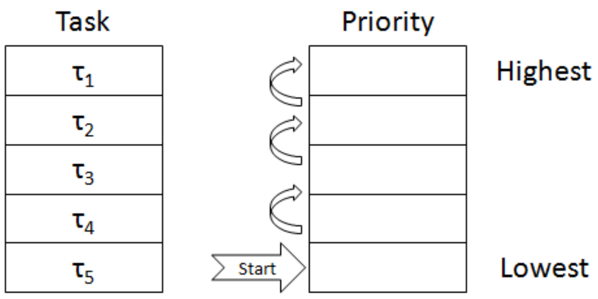

Figure 2.4: It tries to fit a task into the lowest priority.

To apply Audsley's algorithm requires the task model to satisfy a set of prerequisites [17, 28, 29].

- The schedulability of a task must be a function of the set of higher priority tasks, but not their specific priority ordering.
- The schedulability of a task may depend on the set of lower priority tasks, but not on their specific priorities.
- A schedulable task that has its priority raised cannot become unschedulable, and conversely an unschedulable task that has its priority lowered cannot become schedulable.

2.2.4 Schedulability Analysis

In this subsection, we discuss fixed priority scheduling for real-time systems. A real-time system has a number of tasks to execute. Each task has priorities for the system to execute in order. To ensure the system meets all deadlines in the real world, a schedulability test is required for the task-set. In Burns and Wellings' book [18], four characteristics are defined for scheduling testing: Sufficient, Necessary, Exact and Sustainable. A list of the descriptions is shown below.

Sufficient If a task-set passes the test, it will meet all deadlines.

Necessary If a task-set fails the test, it will miss at least one deadline.

Exact Both characteristics of sufficient and necessary.

Sustainable Keep schedulable if the conditions for scheduling have improved (for example, by reducing the utilisation of a task).

Utilisation Based Analysis

Utilisation Based Analysis [18] is a scheduling test which uses the utilisation of tasks but it is only for task-sets which have the characteristic of implicit deadline. This is a simple sufficient schedulability test but not necessary. The utilisation of a task-set, consisting of N tasks, can be calculated via Equation (2.2). If the result is less than or equal to the utilisation bound ($N\left(2^{1/N}-1\right)$), it means the task-set will meet all deadlines. The task-set is surely schedulable.

$$U \equiv \sum_{i=1}^{N} \frac{C_i}{T_i} \leq N\left(2^{1/N}-1\right) \tag{2.2}$$

Table 2.5 shows that the number of tasks will directly affect the utilisation bound. Obviously, if the number of tasks is 1, the utilisation bound is 100%. When the number of tasks is very large, the utilisation bound will approach .693 asymptotically.

Number of task	Utilisation bound
1	100.0%
2	82.8%
3	78.0%
4	75.7%
5	74.3%
10	71.8%
∞	69.3%

Table 2.5: A table for utilisation bounds.

When some tasks are in families in the task-set, the utilisation bound is different. The definition of family [18] for tasks is that the period of a task is an integer multiple of a common value. We can take these tasks as a group and adjust the utilisation bound.

An alternative sufficient schedulability test is shown in Equation (2.3) [18].

$$\prod_{i=1}^{N}\left(\frac{C_i}{T_i}+1\right)\leq 2 \tag{2.3}$$

To conclude, all utilisation based tests are not exact for general task sets because they are sufficient but not necessary. They can only confirm a task-set is schedulable if the task-set passes the test. And also this test is not general because it can only be applied for implicit deadline. The advantage of these tests is simplicity because they are $O(N)$.

Response Time Analysis

Response time analysis (RTA) [47] is an "exact" schedulability test to calculate the worst-case response time of a task which includes the time of interference from other higher priority tasks and blocking from lower priority tasks (due to shared resources or non preemptive scheduling). RTA is not exact unless blocking is exact — which it is usually not. If the worst-case response time of a task is bigger than its deadline, it means the task will not meet its deadline. The opposite situation is that if the worst-case response time of the task is less than or equal to its deadline, the task will meet its deadline. The analysis can be applied for arbitrary deadline.

Before calculating the worst-case response time, the number of releases from other higher priority tasks is needed to be known because a lower priority task will be interfered with by the releases of other higher priority tasks. Equation (2.4) is used to calculate the number of releases (assuming a critical instant) and we take the value of the ceiling function.

$$NumberofReleases=\left\lceil\frac{R_i}{T_j}\right\rceil \tag{2.4}$$

After understanding the number of releases, we apply it into Equation (2.5). The response time R_i for τ_i is calculated by its computation time plus the sum of the time of interference from all higher priority tasks. The time of interference is that the number of releases multiplies its computation time. There is a problem that the number of releases is the ceiling functions of R_i divided by T_j and the result of Equation (2.5) is R_i. Equation (2.5) is solved by forming a recurrence relationship. Please note the blocking B_i is assumed to be 0 for most examples in this thesis as we are concerned with preemptive system with no explicit shared resources.

$$R_i = B_i + C_i + \sum_{j \in hp_{(i)}} \left\lceil \frac{R_i}{T_j} \right\rceil C_j \tag{2.5}$$

$$W_i^{n+1} = B_i + C_i + \sum_{j \in hp(i)} \left\lceil \frac{W_i^n}{T_j} \right\rceil C_j \tag{2.6}$$

We start the first value from the computation time of τ_i. The recursion will continue until $W^n = W^{n+1}$ or the result is greater than the deadline. The final result will be the worst-case response time or showing the task will miss its deadline.

To illustrate, a task-set has three tasks, as shown in Table 2.6. We compute the response time for τ_3. It suffers interferences from τ_1 and τ_2. Initially, the response time is 4 and the second recursion is 9. The result of the third recursion is 11 and the fourth recursion is the same so the worst-case response time for τ_3 is 11.

Task	Period	WCET	Priority
τ_1	8	2	1
τ_2	13	3	2
τ_3	30	4	3

Table 2.6: The highest priority task is τ_1

1. $R_3^0 = 4$
2. $R_3^1 = 4 + (\lceil \frac{4}{8} \rceil \cdot 2 + \lceil \frac{4}{13} \rceil \cdot 3) = 9$
3. $R_3^2 = 4 + (\lceil \frac{9}{8} \rceil \cdot 2 + \lceil \frac{9}{13} \rceil \cdot 3) = 11$
4. $R_3^2 = 4 + (\lceil \frac{11}{8} \rceil \cdot 2 + \lceil \frac{11}{13} \rceil \cdot 3) = 11$

2.2.5 Shared Resources Problem

In a real-time system, there are many tasks and many resources. Some tasks are attempting to execute at the same time and also resources may be used by more than one task. This is called shared resources. The problem of priority inversion [4, 31, 69, 75] appears when a higher priority task is blocked because of a lower priority task using the same resource. At the same time, the lower priority task is preempted by a medium priority task. In this case, the higher priority task effectively executes at the level of the lower priority task.

Task	Priority	Execution blocks	Release time
τ_1	1	XXXABX	5
τ_2	2	XBBBX	3
τ_3	3	XXX	3
τ_4	4	XXAAAXXX	0

Table 2.7: An example task-set.

To illustrate an example of the priority inversion problem, we create an example similar to the one from the book [18]. Consider there are four periodic tasks as shown in Table 2.7.

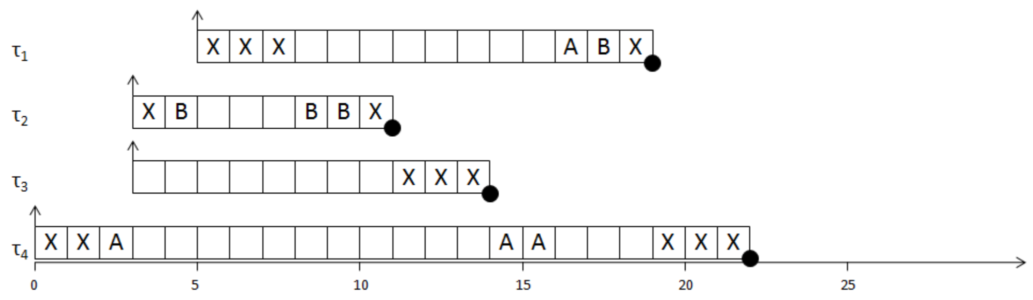

Figure 2.5: The time chart for execution.

The highest priority is 1. In the column of execution blocks, the letter X represents execution without resource; the letters A and B are different resources. Each block is one time unit.

In Figure 2.5, the lowest priority τ_4 releases first, then τ_2 and τ_3 at 3 after the resource A is held by τ_4. τ_2 executes in preference to τ_3 because of its lower priority. τ_1 arrives after the resource B is held by τ_2. After τ_1 executed 3 time units, it cannot access the resource A being held by τ_4, then τ_1 is blocked and τ_2 executes again. After τ_3 has completed, τ_4 executes 2 time units. Finally, the resource A is released at 16 then τ_1 is able to finish the rest. This example shows that τ_1 finished its job after τ_2 and τ_3 although it has highest priority.

Non-preemptive protocol

Non-preemptive protocol [68, 75] is such that when a task enters the critical section then the task will be assigned the highest priority temporarily to achieve the effect of non-preemption. The advantage of this protocol is simplicity but the problem is that other higher priority tasks do not use the critical section may also be blocked. In Figure 2.6 the response time of τ_1 is shorter although the response time of τ_4 is the same.

Priority Inheritance Protocol

Priority Inheritance Protocol (PIP) [68, 75] increases the priority of a task to the highest priority when the task locks one or more shared resources which

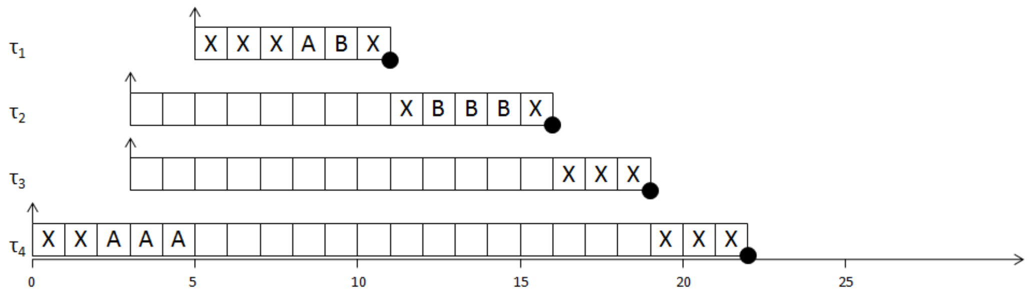

Figure 2.6: The time chart for execution with non-preemptive protocol.

are also needed by other higher priority tasks. It eliminates priority inversion problems. See Figure 2.7.

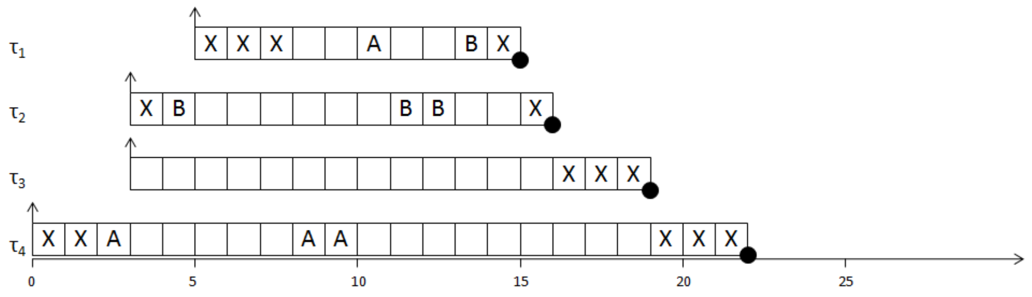

Figure 2.7: The time chart for execution with Priority Inheritance Protocol.

Priority Ceiling Protocol

Priority Ceiling Protocol (PCP) [68, 75] additionally assigns a priority ceiling[2] to each resource so tasks can only be blocked once. Both priority inversion problems and deadlock problem can be solved. See Figure 2.8.

Abort-and-Restart

For completeness we illustrate what would occur if AR is employed with this example. In Figure 2.9 the response time for τ_4 is long, but for τ_1 it is at its minimum as it suffers no preemption and can be release immediately.

[2]When a task enters a critical section, the priority of the task is assigned to the value of the priority ceiling. The priority will be changed back once the task has left the critical section.

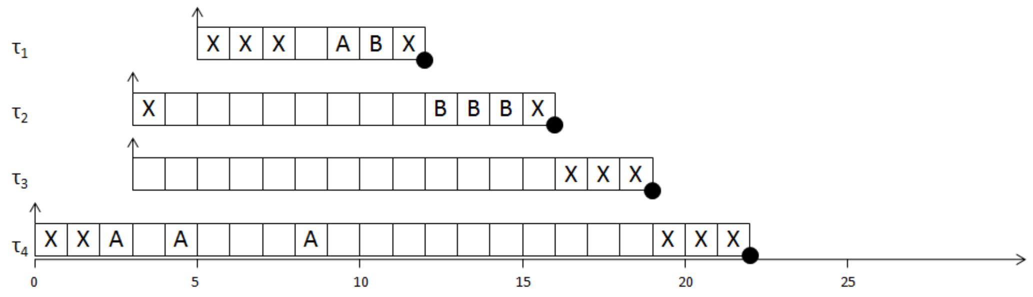

Figure 2.8: The time chart for execution with Priority Ceiling Protocol.

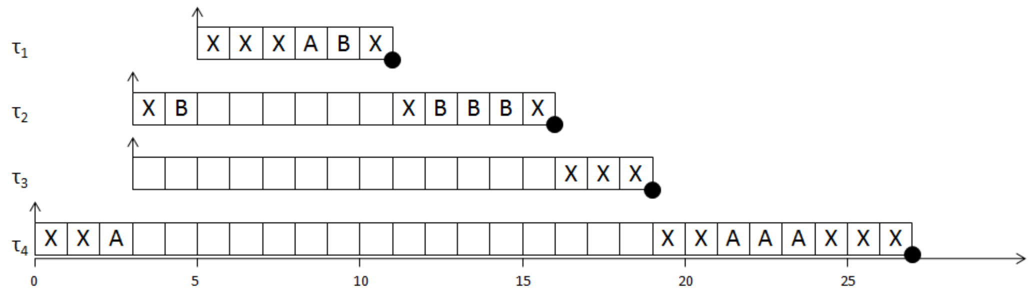

Figure 2.9: The time chart for execution with AR.

2.3 Abort-and-Restart Model

The AR model [70, 69] is an implementation scheme for P-FRP. The classical preemptive model does not fit with P-FRP although it is similar to the AR model except for the operation of AR. In the classical preemptive model, preempted tasks continue their job once higher priority tasks have completed execution. The key concept of the AR model is that lower priority tasks are preempted and aborted by releases of higher priority tasks. Once the higher priority tasks have completed, the lower priority task are restarted as new.

Consider Table 2.8: there is a 2-tasks task-set. τ_1 is the highest priority task and has 3 ticks for WCET. Task τ_2 has 4 ticks for WCET.

Task	Period	WCET	release offset	Priority
τ_1	12	3	3	1
τ_2	15	4	0	2

Table 2.8: An example task-set. (τ_1 has the highest priority)

In Figure 2.10, τ_2 is released at 0 and executes until time 3; because of the arrival of τ_1, τ_2 is aborted at 3. τ_1 finishes its job at 6 and τ_2 is restarted as a new job so the spent time between 0 and 3 is wasted.

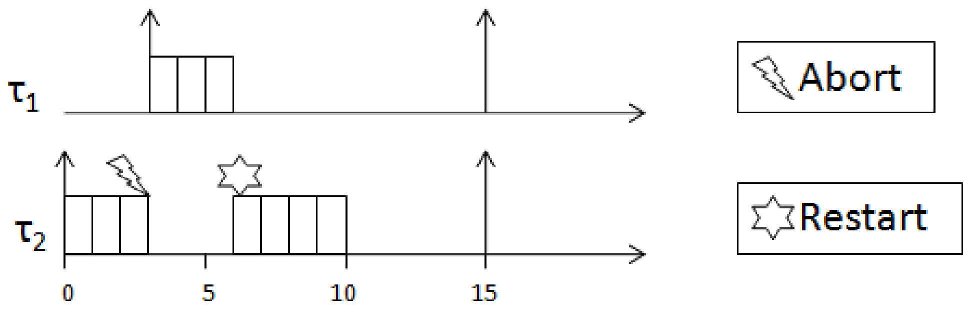

Figure 2.10: An example task-set.

In fixed priority (FP) scheduling, all tasks are statically assigned fixed priorities when developing a real-time system. The ordering of execution of jobs is decided by what priority assignment algorithm is used. Once the priorities are assigned, the system strictly executes tasks based on their priorities. In this thesis, we are concerned with the AR model under fixed priority in uniprocessor systems.

This section contains six subsections which are critical instant, response time analysis, exact response time, priority assignment, case study and aborts reduction.

2.3.1 Critical Instant in P-FRP

The paper given by Ras and Cheng [69] states that the critical instant argument from Liu and Layland [59] may not apply fully to the AR model. In another paper from Belwal and Cheng [9], the authors also realised that a synchronous release of tasks does not lead to the worst-case response time. The simple example in Table 2.8 and Figure 2.10 illustrate this: if τ_1 and τ_2 are released together then $R_2 = 6$. Figure 2.10 shows clearly $R_2 \geq 10$.

A synchronous release implies a critical instant in fixed priority scheduling under the classical preemptive model [59], but the AR model has a different property in that an aborted job will be restarted as a new job. The time spent on a job is wasted and then the total executed time is longer. We consider that P-FRP may have a different nature for its critical instant. An example is given to show that a synchronous release of tasks leads to a shorter response time than an asynchronous release of tasks. To further illustrate the idea, a similar example is given below,

Task	Period	WCET	Priority
τ_1	8	2	1
τ_2	10	2	2
τ_3	12	3	3

Table 2.9: The highest priority task is τ_1.

Consider the task-set in Table 2.9 and the response time for τ_3. If all tasks are released together, the response time of τ_3 is 7, shown in Figure 2.11. Please note that there is no abort. When τ_1 and τ_2 are released at 2 and 4, Figure 2.12 shows that the response time of τ_3 is 9, which is a longer time. An abort happened at 2 and the job is restarted at 6. Unfortunately, 2 ticks are wasted by the abort.

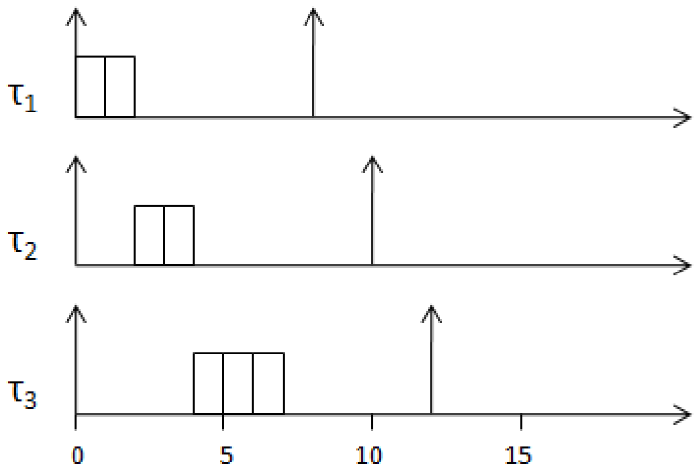

Figure 2.11: All tasks release together.

The example shows that a synchronous release of tasks cannot lead to the worst-case response time in this case and we observe that aborts make

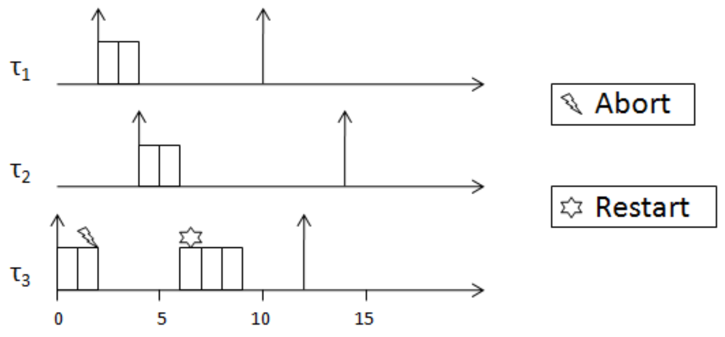

Figure 2.12: τ_1 and τ_2 release at 2 and 4.

the response time of a task longer. When all tasks are released together, a lower priority task has not executed yet and higher priority tasks execute first, so a lower priority task will not be aborted by the first job of any higher priority tasks. The second job of higher priority tasks will only abort the lower priority task when the lower priority task is not finished before the release of the second job for higher priority tasks. We realised that a synchronous release of tasks avoids the aborts that occur with the first job of higher priority tasks.

Currently, there is no previous work that identifies how the critical instant for the AR task model can be found.

2.3.2 Response Time Analysis for P-FRP

The paper given by Ras and Cheng [69] states that standard response time analysis is not applicable for the AR model, and asserts that the abort cost can be computed by the following equation:

$$\alpha_i = \sum_{j=i+1}^{N} \left\lceil \frac{R_i}{T_j} \right\rceil \cdot \max_{k=i}^{j-1} C_k \tag{2.7}$$

α_i is the maximum abort cost for τ_i because the worst case is when a higher priority task aborts the lower priority task which has the biggest WCET. Equation (2.7) uses the number of releases for a task, which has a higher priority than τ_i, then multiplies this by the value of C_k which is the maximum WCET between τ_i and highest priority task.

The central idea of their analysis is that the response time for the AR model can be computed by the combination of standard response time analysis and Equation (2.7). The new equation is as follows:

$$R_i = C_i + \sum_{\forall j \in hp_i} \left\lceil \frac{R_i}{T_j} \right\rceil \cdot C_j + \alpha_i \tag{2.8}$$

$$R_i = C_i + B_i + \sum_{\forall j \in hp_i} \left\lceil \frac{R_i}{T_j} \right\rceil \cdot C_j + \sum_{\forall j \in hp_i} \left\lceil \frac{R_i}{T_j} \right\rceil \cdot \max_{k=i}^{j-1} C_k \tag{2.9}$$

Task	Period	WCET
τ_1	40	3
τ_2	12	4
τ_3	9	3

Table 2.10: A task-set given from the paper [9].

Table 2.10 is a task-set given from the paper [9]. τ_3 is the highest priority task and τ_1 is the lowest priority task. They applied Equation (2.8) to the task-set and the calculation looks as below for τ_3:

1. $R_1^1 = 3 + (\lceil \frac{3}{9} \rceil \cdot 3 + \lceil \frac{3}{12} \rceil \cdot 4) + \lceil \frac{3}{9} \rceil \cdot 3 + \lceil \frac{3}{12} \rceil \cdot 4 = 17$
2. $R_1^2 = 3 + (\lceil \frac{17}{9} \rceil \cdot 3 + \lceil \frac{17}{12} \rceil \cdot 4) + \lceil \frac{17}{9} \rceil \cdot 3 + \lceil \frac{17}{12} \rceil \cdot 4 = 31$
3. $R_1^3 = 3 + (\lceil \frac{31}{9} \rceil \cdot 3 + \lceil \frac{31}{12} \rceil \cdot 4) + \lceil \frac{31}{9} \rceil \cdot 3 + \lceil \frac{31}{12} \rceil \cdot 4 = 51$

The task-set is deemed unschedulable.

To further illustrate Equation (2.7), there are two examples shown in Figures 2.13 and 2.14. The examples assume that the task-set in Table 2.11 is schedulable and all aborted jobs are aborted just before they finish. The result of α_1 is 0 because the highest priority task does not suffer any abort. α_2 is 2 because of one abort by τ_1. α_3 is 6 because of aborts from τ_1 and τ_2.

In Figure 2.13 this case is to show that all tasks, except the highest priority task, have one abort. In this case, the response time of τ_3 is 11.

The next case in Figure 2.14 is again to find out the response time for τ_3. It is also using the same task-set from Figure 2.13. Following the equation

Task	Period	WCET	Priority
τ_1	7	1	1
τ_2	9	2	2
τ_3	12	3	3

Table 2.11: The highest priority task is τ_1

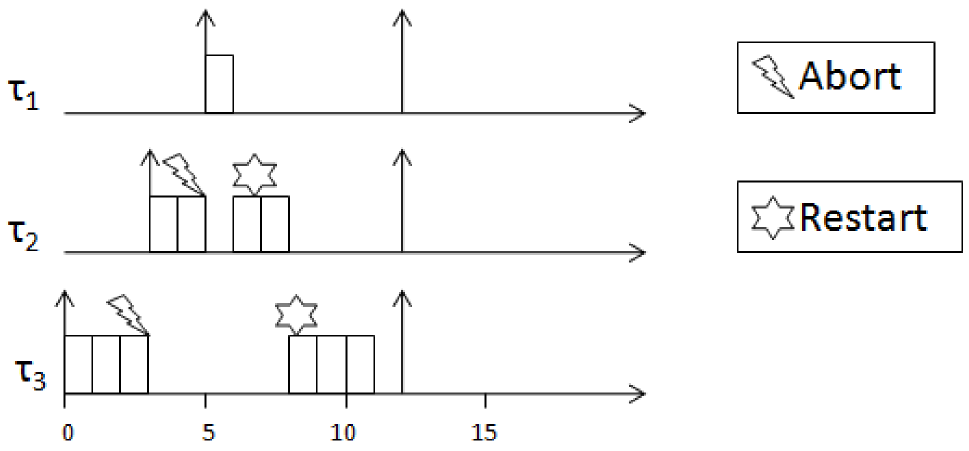

Figure 2.13: The response time for τ_3 is 11.

of AR cost, τ_3 suffers two aborts from τ_1 and τ_2 to maximise the AR cost because the computation time of τ_3 is larger than the computation time of τ_2. As Figure 2.14 shows, the response time for τ_3 is 12.

To compare the results of response time for τ_3, the case in Figure 2.14 is the worst case and also it fulfils the equation of AR cost.

Equation (2.7) from Ras and Cheng [69] is used to compute the cost for AR. The cost is the WCET that can be wasted because of arrivals of other higher priority tasks. If Figure 2.14 is the worst case, the critical instant for τ_3 should be when τ_2 releases at 3 and τ_3 releases at 8. It also again proves that a critical instant for a task is not when all higher priority tasks are released with the release of the task.

The calculation of AR costs was derived by Ras and Cheng [69]. They tried to apply the equation to the standard response time analysis for utilising the AR model. The idea from the paper of Ras and Cheng [69] is that (assuming a critical instant) the AR cost is computed by their Equation

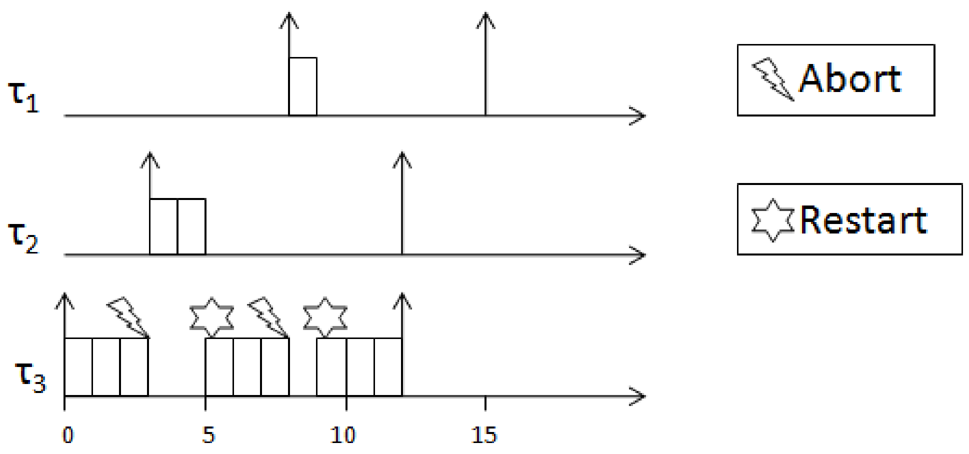

Figure 2.14: The response time for τ_3 is 12.

(2.10), where w_i^{m+1} is the worst-case response time calculated by the standard response time analysis. After the values of AR cost and worst-case response time are found, the final worst case response time for the AR model is the sum of those values. The mathematical expression is shown as follows:

$$R_i = w_i^{m+1} + \alpha_i \tag{2.10}$$

The concept of Equation (2.10) is potentially very pessimistic. The response time analysis for the classic model is exact, which means sufficient and necessary. If the task-set passes the RTA test, all tasks will meet their deadlines. If the task-set fails the RTA test, a task will miss its deadline at run-time (if WCETs are accurate). The result of this analysis is that standard response time analysis with AR cost is sufficient but not necessary. This approach of calculation of the worst-case response time for AR model is degraded from exact to sufficient only.

2.3.3 Exact Response Time for P-FRP

After the sufficient test is introduced, some methods to compute the exact response time for P-FRP are given by Belwal and Cheng [9]. They are Time-Accurate Simulation (TAS), Gap-Enumeration Method and Idle-period Game Board Algorithm. A synchronous release is assumed as a critical instant for all the methods.

Time-Accurate Simulation

Time-Accurate Simulation (TAS), Belwal and Cheng [9] believe, is a simple way to calculate the response time in P-FRP. The approach is to execute a simulation through each time unit and run tasks under the P-FRP execution model. They state that the upper-bound time complexity is $O((T_j - C_j) \cdot (n - j)^2 \cdot T_k^2)$. We are not concerned with the accuracy but we take this as a reference for the worst case. It is an inefficient method for response time analysis in P-FRP. So the contributions of this algorithm are the correct response time calculation and the worst performance of response time algorithm in P-FRP. When we design or analyse an algorithm for response time in P-FRP, the time-accurate algorithm can be a reference.

The accuracy of another algorithm can be checked by using the value generated from TAS, for example, with an algorithm A and a task-set T. If the algorithm A can schedule the task-set T as TAS can, it means the algorithm A is correct. But we should consider that an algorithm may produce correct results in only some cases. The best way is to create a large number of task-sets so that the algorithm can be tested in many cases. It is not a full-covered test but it increases the reliability. And also, the performance of another algorithm can be compared to the time complexity of TAS. If an algorithm has a higher time complexity than TAS, it means the design of this algorithm is worse than TAS. The authors ran TAS with the task-set shown in Table 2.10 and the result for the computational steps is 145, but no result of response time is given in their paper. The TAS approach is exact if the task-set has a critical instant, but if the critical instant for an AR task-set cannot be found then this approach is not usable.

Gap-Enumeration Method

The Gap-Enumeration Method was developed by Belwal and Cheng [9]; enumeration of k-gaps is used for reducing the time complexity for a calculation of response time analysis in P-FRP. The idea is that there is a time slot which is first allocated with the highest priority task and second with the next higher priority task and so on. When a task is fitted, the response time for the task is found. If the task τ_i cannot find a gap, it means it misses the deadline. The paper by Belwal and Cheng illustrates their idea with an example using the same task-set shown in Table 2.10. In their paper, there are two mistakes regarding the example for the Gap-Enumeration Method: that 1-Gap should be shown in the last diagram but they wrongly labelled it 3-Gap, and the final calculation of the response time for τ_i should be $R_1 = 21 + 3 = 24$ but they incorrectly presented $R_1 = 21 + 4 = 24$. However, the result of the example is correct and the performance is better than TAS.

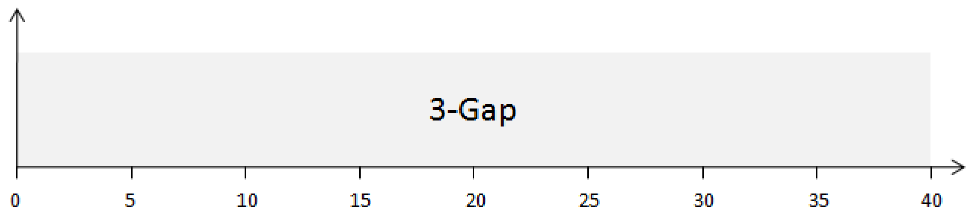

Figure 2.15: It shows the gap for τ_3.

The example is to calculate the response time for τ_1. We know that task τ_3 is the highest priority task shown in Table 2.10. As Figure 2.15 shows, the deadline and period of τ_1 is 40 and we do not consider the time after the deadline so the gap-set is between 0 and 40. Obviously, the gap-set is all available because we have not yet tried to fit the highest priority task.

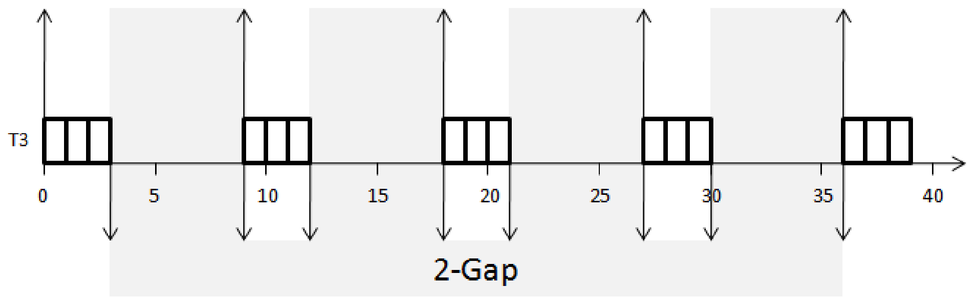

Figure 2.16: It shows the gaps for τ_2.

Figure 2.16 shows task τ_3 fitted into the gap-set. The available gaps between 0 and 40 are 3–9, 12–18, 21–27, 30–36 and 39–40. Task τ_2 is now fitted into the gap-set. Figure 2.17 shows that τ_2 is aborted at 27 but the deadline is still met. And the gaps become narrow but we have to put τ_1 into a suitable gap, otherwise the task-set is not schedulable. At this stage, the gap-set has 7–9, 16–18, 21–24 and 34–36. Fortunately, τ_1 is fitted with the gap between 21 and 24. The response time for τ_1 is $21 + 3 = 24$.

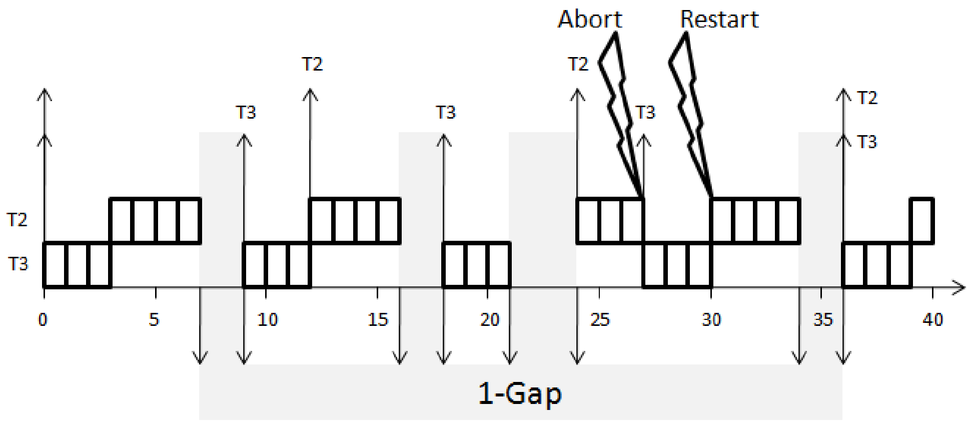

Figure 2.17: It shows the gaps for τ_1.

The gap-search function can be simply explained in that it searches the first k-gap which is fitted with the size of P_k. Figure 2.18 shows that the authors use a red-black tree (RB-tree) [24]. A RB-tree is a binary tree where the colour of nodes can be either red of black. The properties of an RB-tree are directly cited from the paper [9] listed as below:

- The root node is black
- All leaf nodes are black
- Children of every red node are black
- Path from leaf to root contains the same number of black nodes

Figure 2.18 shows a sample of an RB-tree which contains a gap-set: $\sigma_k(T|_0^{320}) = \{[10, 40), [50, 80), [90, 100), [120, 140), [170, 190), [230, 260),$ $[300, 320)\}$. The RB-tree is started from the left side so the order for the node index is 10, 50, 90, 120, 170, 230, 260 and 300.

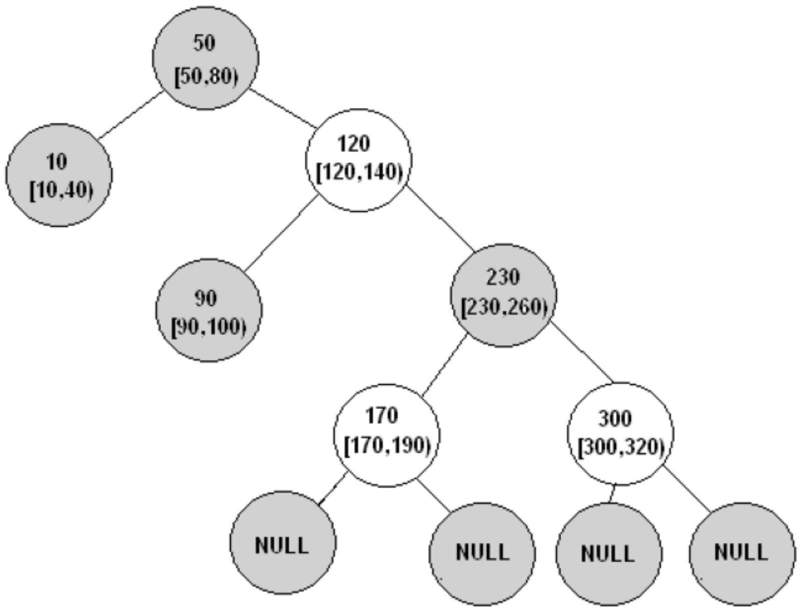

Figure 2.18: RB-tree for Gap-Search Function. (Cited from the paper [9])

The authors ran the algorithm with the task-set shown in Table 2.10 and showed that the number of computational steps is 33. To compare this result to the result of TAS, it is much faster in the task-set with only three tasks.

After the Gap-Enumeration Method was discussed, we noted that the concept of k-gap is similar to the concept of idle period, and the examples in the paper [9] are not completed tests because the length of the time chart is not the LCM for the task-set. The LCM for the task-set shown in Table 2.10 is 360. In P-FRP, the first job of τ_i meets the deadline but that does not mean other jobs of τ_i will meet deadlines because a synchronous release may not lead to a critical instant. It remains an open question as to whether the Gap-Enumeration Method is sufficient if there is no critical instant.

Idle-period Game Board Algorithm

Belwal and Cheng [10] think that Gap-Enumeration Method is hard to program because an RB-tree is not available as a native function in programming languages. Another technique using a game board is an easier way because the method can be implemented by using a simple array. Moreover, they changed the term Gap-Enumeration to idle-period. Idle-period game board algorithm is also the same as the Gap-Enumeration Method. In Figures 2.15, 2.16 and 2.17, the term Gap is changed to *idle period*. To illustrate, the task-set in Table 2.10 is used again.

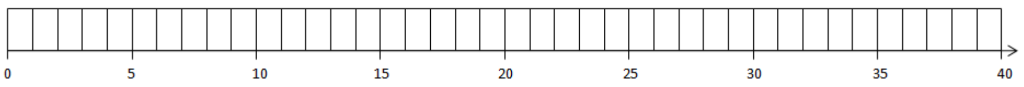

Figure 2.19: An empty game board.

In Figure 2.19, there is an empty game board. The LCM of the task-set in Table 2.10 is 360 but the length of the game board is 40. The reason is that the deadline of $\tau 1$ is 40. If a job of $\tau 1$ cannot be fitted in, it means the task-set is not schedulable.

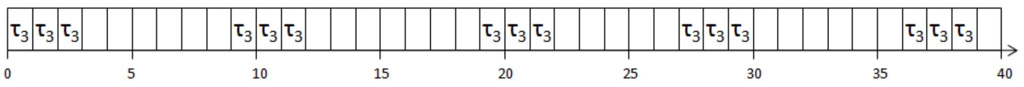

Figure 2.20: Jobs of τ_3 are fitted.

In Figure 2.20, jobs of τ_3 are fitted into the game board and the blank slots are idle periods.

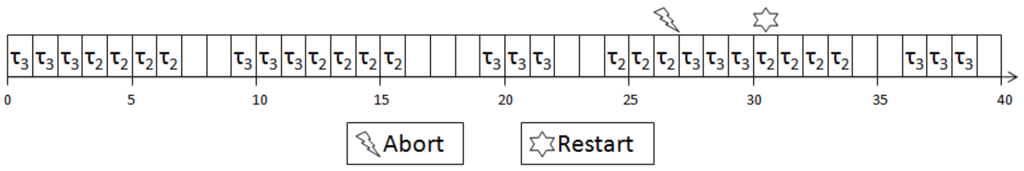

Figure 2.21: Jobs of τ_2 are fitted.

In Figure 2.21, the method next puts jobs of τ_2 into the game board. There is an abort at 27 and then the job is restarted at 30.

The idea of the Idle-period game board algorithm is the same as the Gap-Enumeration Method except for the implementation so they share the same issues. Lastly, we finish the example as shown in Figure 2.22.

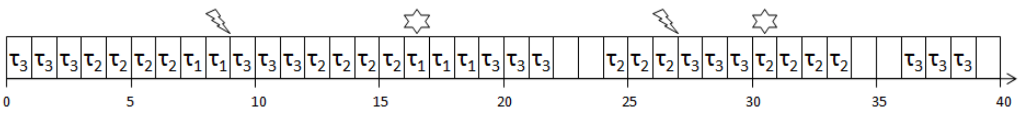

Figure 2.22: Jobs of τ_1 are fitted.

2.3.4 Priority Assignment in P-FRP

Rate monotonic priority assignment and Utilisation-and-Rate Monotonic priority assignment are reviewed for the AR model using Equation (2.8).

Rate Monotonic Priority Assignment

Rate monotonic priority assignment is not optimal in the AR model [7]. To prove this, we can give an example to show that a task-set is not schedulable with a rate monotonic priority assignment but is schedulable by using other fixed priority scheduling. In the paper by Belwal and Cheng [7], an example was given to prove the above statement.

Task	Priority	WCET	Period	Utilisation
τ_1	2	7	15	0.46
τ_2	1	3	12	0.25

Table 2.12: Task-set A.

The task-set A in Table 2.12 shows that τ_2 has the highest priority because of the rate monotonic priority assignment with the task having the shortest period. According to priorities assigned in Table 2.12, Figure 2.23 shows τ_1 will miss its deadline at 45 because τ_1 is aborted at 36, which is also the release for τ_2. After the job of τ_2 is done at 39, τ_1 is restarted immediately. Unfortunately, there are only 6 time units left but τ_1 needs 7 time units for its job.

But Belwal and Cheng [7] said that if τ_1 and τ_2 swap their priorities, as given below:

then the task-set B will be schedulable, as shown in Figure 2.24. In Table 2.13, after switching their priorities, τ_1 has the highest priority and has no

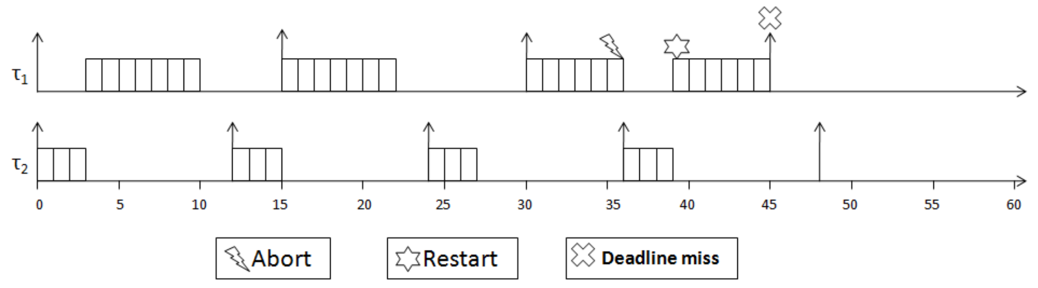

Figure 2.23: It is not schedulable under RM scheduling.

Task	Priority	C	Period	Utilisation
τ_1	1	7	15	0.46
τ_2	2	3	12	0.25

Table 2.13: Task-set B.

abort by τ_2. τ_2 has delayed its releases at 36 and 48, but they still meet all their deadlines.

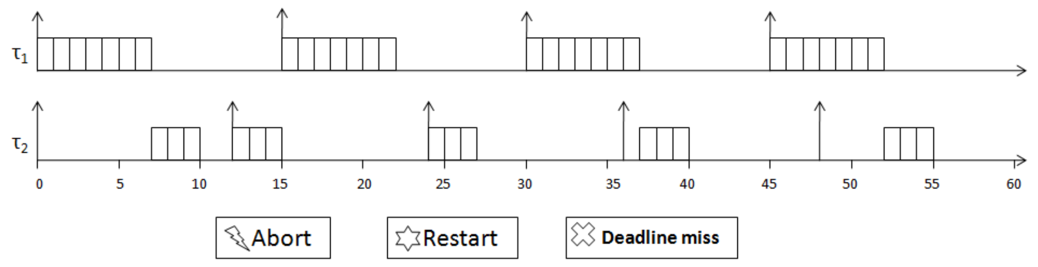

Figure 2.24: It is schedulable after changing their priorities.

The example apparently proves that rate monotonic priority assignment with the AR model is not optimal, but the worst case must be considered when doing a schedulability test. In the section on critical instant, we showed that the AR model and classical preemptive model have different characteristics of critical instant. As Figures 2.23 and 2.24 show, τ_1 and τ_2 are released at the same time and that means the example is not the worst case.

In Figure 2.25 Task τ_1 is released 7 time units earlier than task τ_2 releases, then τ_1 is aborted at 7. According to another paper[69] by Ras and Cheng, the equation of AR cost for fixed priority was proposed to calculate the

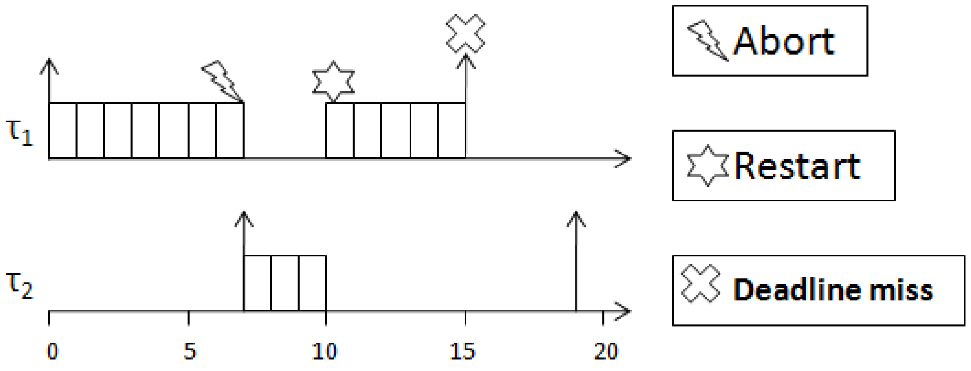

Figure 2.25: It is the worst case under RM.

maximum value. In their paper, they used the WCET of a task as the maximum time that can be wasted as AR cost so we assume that task τ_2 arrives before task τ_1 almost finishes, then we take the WCET of task τ_1 as the worst case. The task-set A in Table 2.12 is not schedulable whether in the worst case or not, as shown in Figure 2.25.

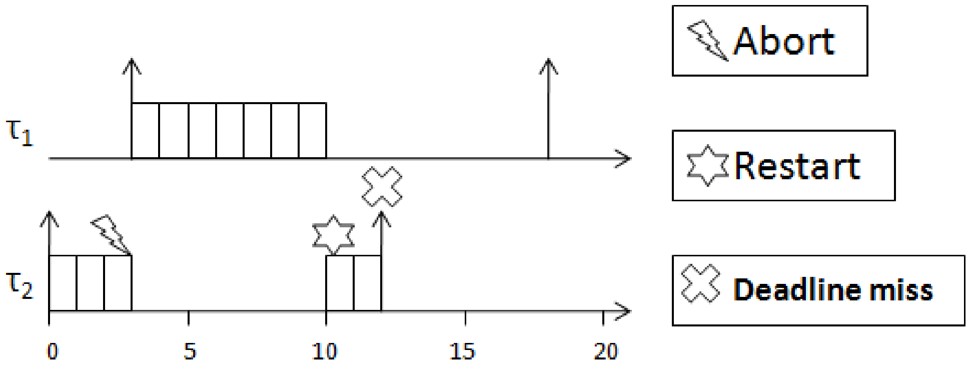

Figure 2.26: It is the worst case under non-RM.

We must conclude that the task-sets A and B fail to prove their assertion that RM is not optimal. We consider the task-set B in Table 2.13 may not be schedulable in the worst case. Figure 2.26 shows that τ_2 is aborted by arrival of τ_1 at 3 then τ_2 misses its deadline at 12. It means that the task-set B in Table 2.13 is also not schedulable although their priorities changed.

As the above worst-case example cannot prove that the rate monotonic priority assignment is not optimal in the AR model, a modified example is

Task	WCET	Period	Utilisation
τ_1	6	14	0.43
τ_2	3	12	0.25

Table 2.14: task-set C.

created in Table 2.14. The WCET and period of τ_1 are changed to 6 and 14.

Firstly, the rate monotonic priority assignment for the task-set in Table 2.14 means that τ_2 has the highest priority. In Figure 2.27 τ_1 is aborted by the arrival of τ_2 at 6 and it restarts at 9 but misses its deadline at 14. This example shows the task-set in Table 2.14 is not schedulable with rate monotonic priority assignment.

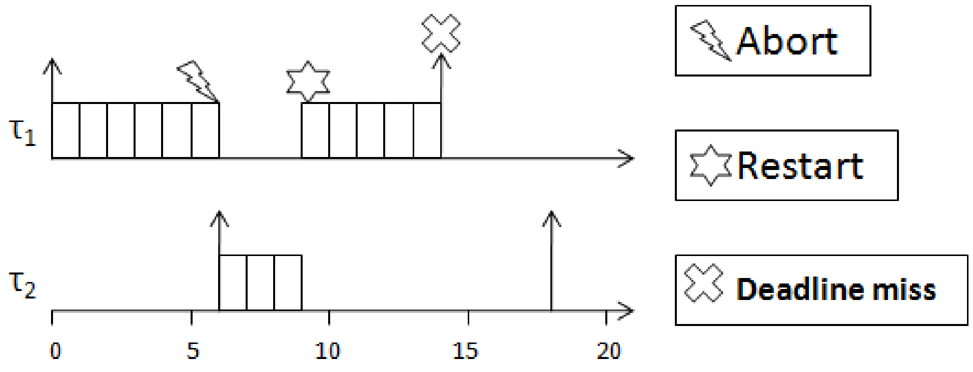

Figure 2.27: Task-set C in worst case under RM.

Secondly, we assign the highest priority to τ_1 to test another fixed priority scheduling rather than RM priority assignment. In Figure 2.28, τ_2 is aborted at 3 when τ_1 arrives. τ_2 restarts at 9 and finishes its job at 12 which meets its deadline. With this priority assignment, the task-set C in Table 2.14 is schedulable. This is clear evidence to prove that RM priority assignment with the AR model is not optimal in the worst case.

To sum up this section, we discussed the fact that the rate monotonic priority assignment with the AR model is not optimal. The worst case for a task-set needs to be considered when comparing priority assignments. We also provided a worst-case example as stronger evidence.

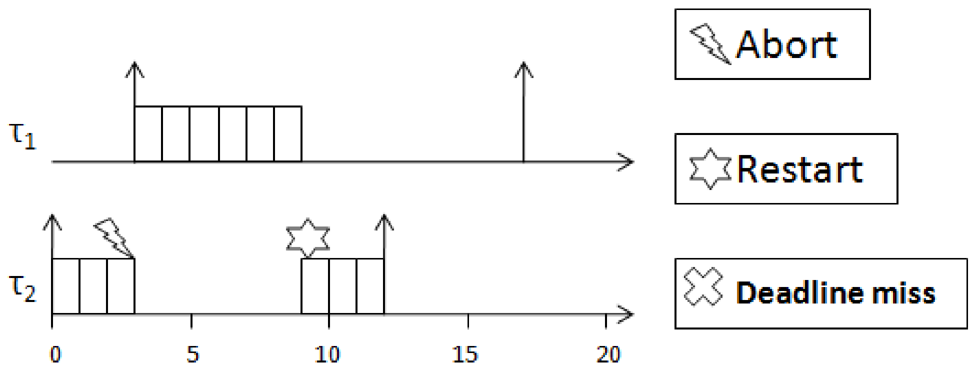

Figure 2.28: Task-set C in worst case under non-RM.

Utilisation-and-Rate Monotonic

Utilisation-and-Rate Monotonic (U-RM) priority assignment is mentioned in the paper given by Belwal and Cheng [7]. The idea of U-RM priority assignment is that a task is assigned with a higher priority if it has a smaller arrival rate and bigger utilisation. To illustrate, consider the task-set shown in Figure 2.15. The task-set contains five tasks. The highest priority task, τ_1, has the smallest period and the biggest utilisation.

Task	WCET	Period	Utilisation	Priority
τ_1	3	10	0.30	1
τ_2	2	12	0.16	2
τ_3	2	20	0.10	3
τ_4	1	25	0.04	4
τ_5	1	40	0.02	5

Table 2.15: This is U-RM priority assignment. (The highest priority is 1)

Clearly, this priority assignment algorithm is only applicable to task-sets that have U-RM property. In general this will not be the case.

2.3.5 Case Study for the AR model

There are two cases [69] which are implementations in both software and hardware. The case study for software that the authors used is the Generic

Avionics Platform (GAP) task-set to evaluate the AR model under RM and Earliest Deadline First (EDF) scheduling and then compare the result with other models such as non-preemptive, PCP and Stack Resource Policy (SRP). The other case is for hardware; the Analog Devices' ADuC814 microcontroller is used for running the P-FRP compiled code with RM and EDF scheduling. The result of this case study is the number of tasks against the average number of aborts and the average number of aborts against system load.

There are two diagrams of the case study for software shown in Figures 2.29 and 2.30. Figures 2.31 and 2.32 are the result of the case study for the hardware. We directly cited all diagrams from the paper by Ras and Cheng [69].

Software — Generic Avionics Platform

The case study for software is the GAP [52, 60], is used in the simulation. GAP models the functionality of an aircraft computer system and data handling that was created by Locke et al [61]. In the paper by Ras and Cheng [69], the authors list some of the avionics timing constraints, as below:

1. Navigation: The frequency of navigation is 20 Hz, which is based on the requirements of accuracy.
2. Display: The period is between 65 ms and 100 ms.
3. Ballistics Computation: The vehicle trajectory, altitude and attitude require 5 ms in period.
4. Sensor Control: The frequency of radar antenna search is 10 Hz. 1KHz or more is for electromagnetic surveillance equipment.

Ras and Cheng [69] used the theory of the GAP task-set which has sixteen periodic tasks and one sporadic. In the experiment, the authors assume that all tasks are periodic and have no release jitter. The motivation is to observe how much penalty is introduced by the AR model because they know that PCP and SRP are apparently more efficient. RM priority assignment is used with the heavy and lighter resources usage that produces the two results shown in Figures 2.29 and 2.30. They also state that the performance of RM is based on the arrival pattern and synchronous release is the worst case for RM.

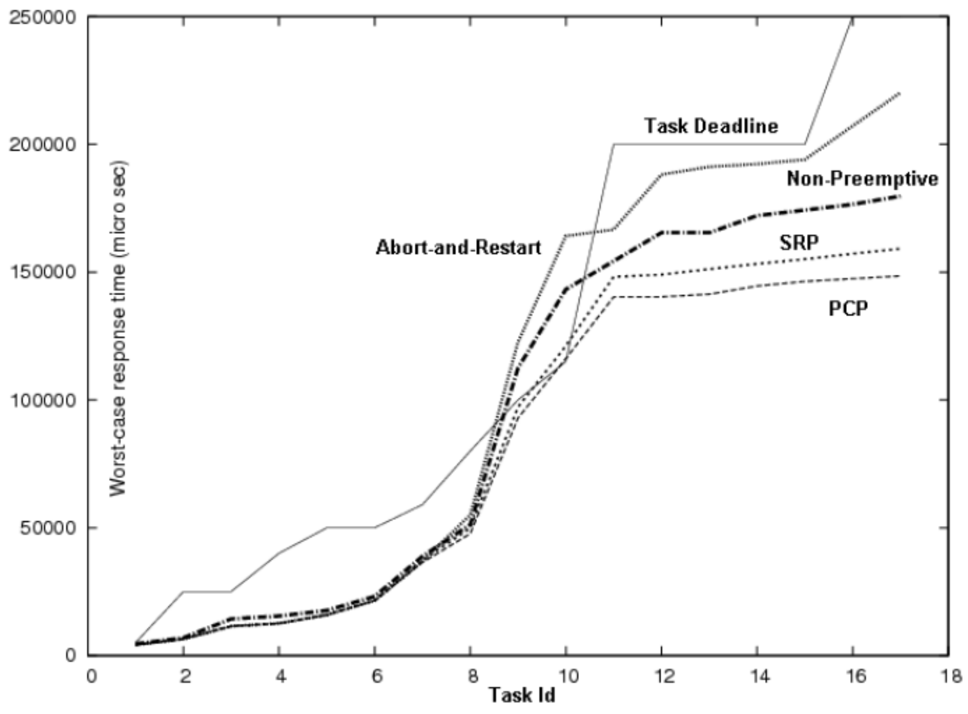

Figure 2.29: Heavy resource usage (long critical sections). (Cited from the paper [69])

Figure 2.29 shows the results of different policies (AR, SRP, PCP and non-preemptive) with RM priority ordering under heavy resource usage. The AR model has the worst performance, PCP and SRP perform well and the non-preemptive model is just better than the AR model. This diagram shows the result under heavy resource usage which means the system has long critical sections.

The next diagram, in Figure 2.30, is about the lighter resource usage in which the system has shorter critical sections. In this test, SRP is better in performance. PCP, AR and non-preemptive have the same performance as the test with heavy resource usage. In the paper [69], there are also two tests based on EDF scheduling which is a dynamic priority scheduling; we do not discuss this approach in detail in this review.

According to the observation of the experiments, Ras and Cheng did not realise that the critical instant for P-FRP may not occur when all tasks are released at the same time. The result of the AR model could get worse because it is not based on the worst case. From our observation, the performance of the AR model has a lot of room for improvement because RM priority assignment is not optimal for the AR model.

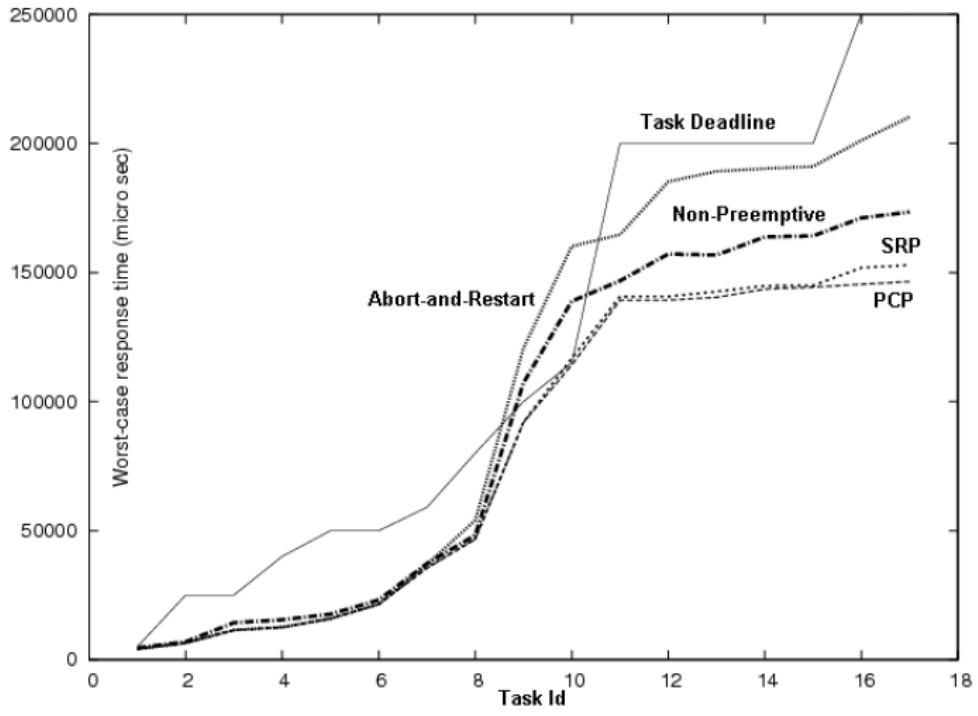

Figure 2.30: Light resource usage (short critical sections). (Cited from the paper [69])

Hardware — Analog Devices' ADuC814

Ras and Cheng also present an experiment with hardware implementation in their paper [69]. They run P-FRP compiled code on the hardware board which uses an Analog Devices' ADuC814 micro-controller. The results are shown in Figures 2.31 and 2.32. This time we discuss both fixed and dynamic priority scheduling.

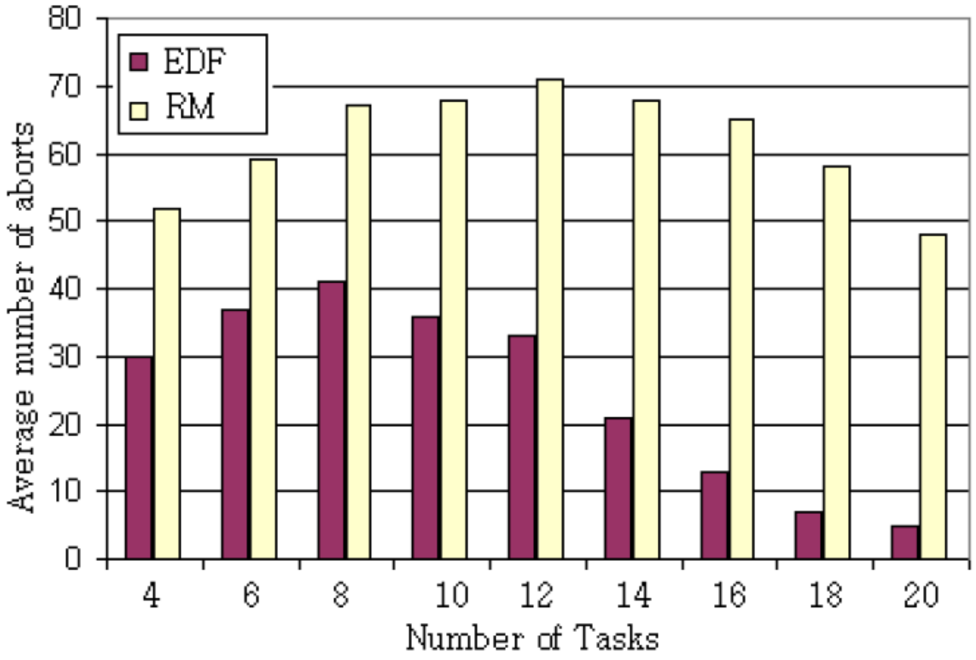

Figure 2.31: Number of tasks. (Cited from the paper [69])

In Figure 2.31, there is a diagram of the average number of aborts against the number of tasks with RM and EDF scheduling, and the utilisation is fixed. The average number of aborts for RM scheduling is always higher than EDF scheduling. When the number of tasks gets higher, the average number of aborts decreases. The authors believe the reason is that the higher number of tasks makes the worst-case computation time get smaller for each task. It makes the possibility of aborts occurring lower.

Figure 2.32 shows the average number of aborts against the CPU load, and the utilisation is fixed. In this result, Ras and Cheng have less discussion about it. But we can see from the diagram that the average number of aborts for the RM is increasing when the CPU load is getting higher. On the other hand, the average number of aborts for EDF is decreasing after the CPU load is 0.85.

Based on our observation, EDF scheduling is much more efficient than fixed priority assignment with RM priority assignment for the AR model. RM is not optimal for P-FRP but the number of aborts is an important

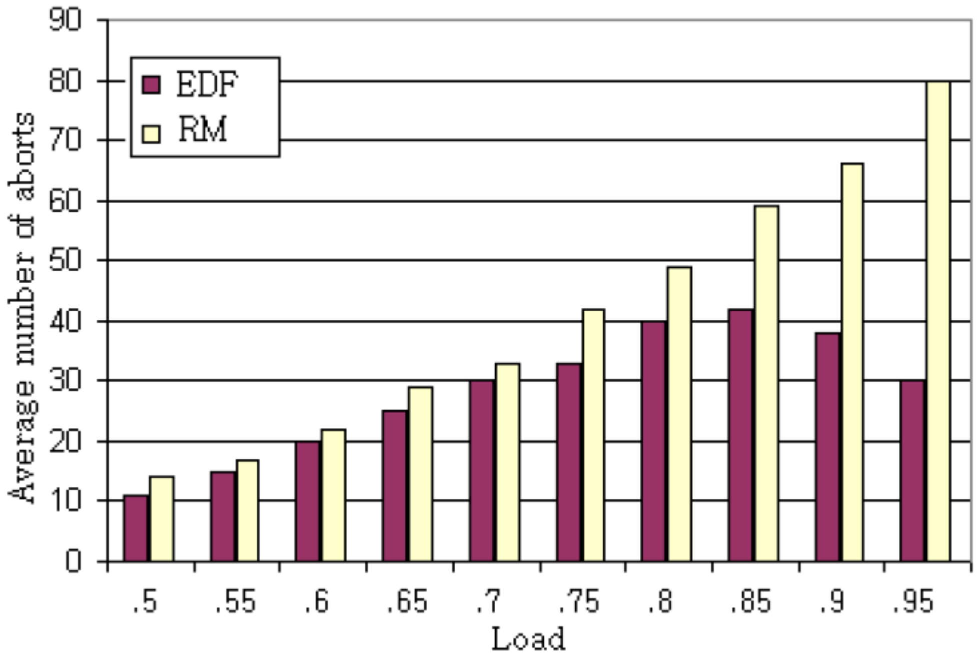

Figure 2.32: System load. (Cited from the paper [69])

property for schedulability analysis in P-FRP.

2.3.6 Methods to Reduce Aborts in P-FRP

In P-FRP, aborts are overheads as time spent on a task is wasted if a higher priority task arrives before the task completes. Therefore, the number of aborts is crucial for the performance of the system. From the hardware case study, we noted that the number of aborts is very high for the AR model. To improve the schedulability of systems, we consider any method to reduce the number of aborts. The paper given by Belwal and Cheng [8] presents two methods to reduce the number of preemptions in the classic task model.

The authors [8] discuss the modifications in task attributes to reduce the number of preemptions. There are three methods: priority reassignment for individual jobs, release offset of tasks and release times of individual jobs. And also the modifications can be in the level of scheduler: preemption threshold and deferred preemptions.

Modifications in Task Attributes

We firstly discuss the modifications in task attributes. In general, task attributes are release times, release offsets, priorities, WCET, periods and deadlines. We are concerned only with release times, release offsets and priorities because those properties are flexible for amending. If the number of preemptions can be reduced, it can also reduce the number of aborts.

Job-level Fixed Priority

As we concerns fixed priority in this thesis, job-level scheduling can reduced the number of aborts. A synchronous release is assumed as a critical instant. Priority reassignment for individual jobs of a task can be assigned with individual priorities to avoid preemptions occurring between two consecutive jobs. A classical task-set only contains priorities for tasks; all jobs of a task strictly execute based on the priority of the task in the system. This approach is not efficient because it usually increases the number of preemptions. Dobrin and Fohler [32] found that priority reassignment for individual jobs is one of the methods to reduce the number of preemptions. The method is that the authors expand a classical task-set to a new task-set which has individual jobs with independent priorities. For example, if the period of task τ_i is 10 and the LCM for the task-set is 50, then there are five jobs for the task τ_i which are τ_i^0, τ_i^1, τ_i^2, τ_i^3 and τ_i^4. The jobs between τ_i^0 and τ_i^4 are input into a new table as tasks. All tasks are assigned with the LCM as periods. The release offsets of tasks are assigned by using the position of the job multiplied by its original period, starting from 0. So the new offset for τ_i^0 is still 0, τ_i^1 is 10, τ_i^2 is 20 and so on. When the system starts, all tasks release together but they execute based on their priorities and release offsets.

In this section, we consider how the number of aborts in P-FRP can be reduced. Unfortunately, the paper by Dobrin and Fohler [32] focuses on the classical preemptive model and the paper by Belwal and Cheng [8] has no example for this approach. We create an example with two tables and two figures to illustrate this method using the AR model. Tables 2.16 and 2.17 show that a general task-set extends to the priority reassignment for individual jobs. Figure 2.33 illustrates that the task-set is not schedulable. After the priority reassignment for individual jobs, the task-set becomes schedulable, as shown in Figure 2.34.

Task	Period	WCET	Priority
τ_a	4	1	1
τ_b	5	2	2
τ_c	10	2	3

Table 2.16: An example task-set. τ_a has the highest priority

We begin the example with Table 2.16, which has four columns (Task, Period, WCET and Priority) as task attributes. The LCM for this task-set is 20 and hence the diagrams are only presented between 0 and 20. Looking at Figure 2.33, the task τ_a executes its jobs well because of its highest priority. Task τ_b has an abort at 16 but it still meets the deadline. Task τ_c completely fails because no job can be done before the deadlines. And there are three aborts occurring at 4, 8 and 15. The task-set is not schedulable under the AR model.

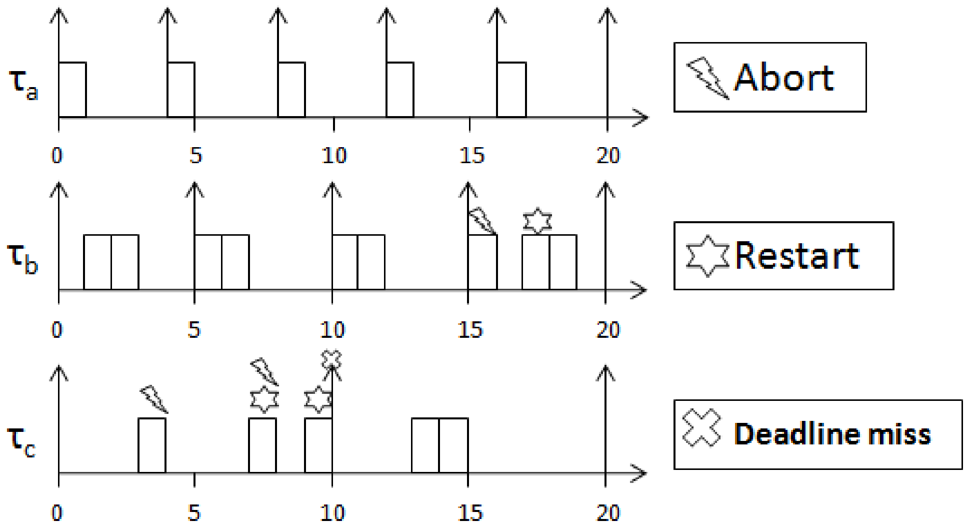

Figure 2.33: The task-set is not schedulable under the AR model.

Now, we modify the task-set using the priority reassignment for individual jobs [32]. In Table 2.17, it is a slightly different form from Table 2.16; a column for release offset is added and periods for all task are assigned with the LCM, 20. The number of tasks are expanded from 3 to 11. The task τ_a has 5 tasks expanded, from 1 to 5, because its period is 4 and the LCM is 20 then 5 jobs are executed in this interval. The task τ_b has 4 expansions and the task τ_c has 2. The values of release offsets are assigned by following the rule of the number of position for the original task times the old value of the period. So the new value for τ_a^0 is 0, τ_a^1 is 4, τ_a^2 is 8 and so on. Looking at the column of priority, each individual task can be assigned with different priorities. We note that the expansion for the table is related to the LCM and the number of tasks. The size of a table is also a factor for overheads in real-time systems. Each time a task arrives, the preemptive system will check the table and this action is an overhead. We note also that the method applies only to periodic tasks.

Task	Period	WCET	Release offset	Priority
τ_a^0	20	1	0	4
τ_a^1	20	1	4	2
τ_a^2	20	1	8	1
τ_a^3	20	1	12	3
τ_a^4	20	1	16	2
τ_b^0	20	2	0	3
τ_b^1	20	2	5	1
τ_b^2	20	2	10	4
τ_b^3	20	2	15	1
τ_c^0	20	2	0	2
τ_c^1	20	2	10	2

Table 2.17: The new task-set after priority reassignment for individual jobs.

As Figure 2.34 shows, all tasks release at 0 and τ_a^0 executes first with its highest priority and 0 release offset. τ_b^0 executes as second at 1 and finishes at 3. τ_c^0 executes until it finishes whatever τ_a^1 releases at 4 and τ_b^1 releases at 5. After τ_c^0 is completed, τ_a^1 executes before τ_b^1 because τ_a^1 has higher priority. τ_a^2 arrives at 8 but it has to wait until τ_b^1 finishes because of lower priority. At 10, both τ_b^2 and τ_c^1 release, τ_b^2 executes first and finishes at 12. In the meantime, τ_a^3, higher priority, releases so τ_c^1 executes at 13. τ_b^3 releases at 15 but τ_c^1 is still executing. Although τ_c^1 is completed at 16, τ_b^3 cannot execute. The reason is that τ_a^4 is just released at that time. The last task τ_b^3 is finished at 19. All tasks meet their deadlines and the task-set is schedulable using this method under the AR model.

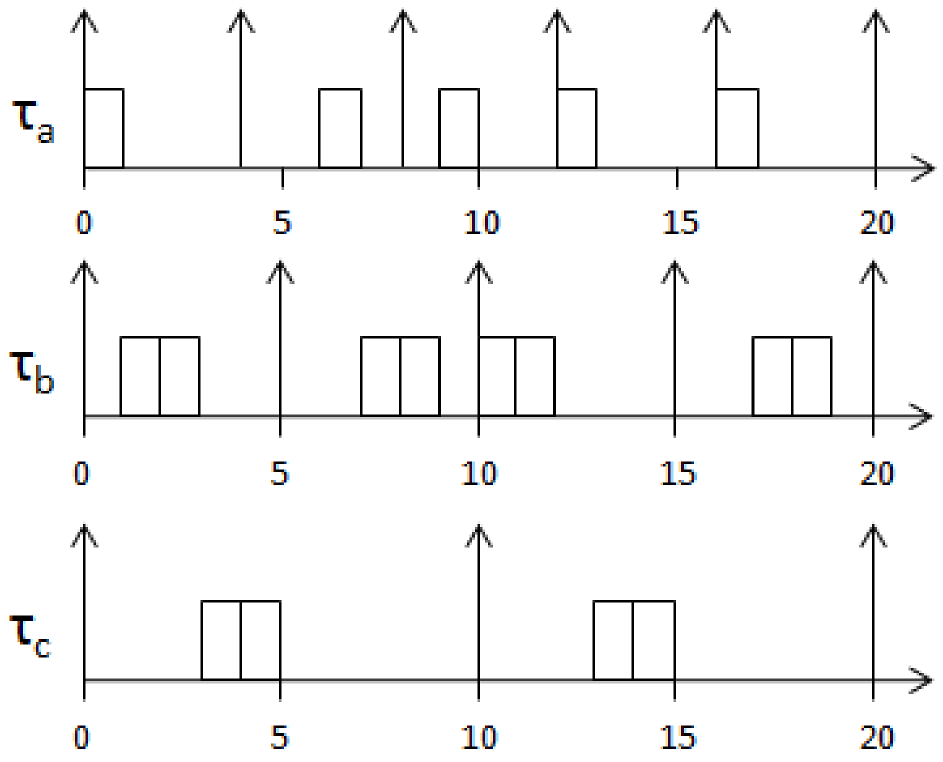

Figure 2.34: The time chart after priority reassignment for individual jobs.

As illustrated in the example presented above, priority reassignment for individual jobs provides a solution so that the number of aborts is reduced, even to zero abort, and the schedulability for the AR model is improved. But we realise that the expansion of the size of table can be a potential issue because it increases the overheads on arrival tasks. On the other hand, the concept of this method is similar to the dynamic priority scheduling. The priority of a task can be changeable by separating all jobs of the task into tasks so that the task is able to have different priorities at different stages. The difference is that this method, in effect, is offline analysis but dynamic priority is online. And also, some restrictions are applied to this method such as periodic tasks only. The main advantage is that with this algorithm there is no requirement to modify the basic fixed priority scheduling mechanism [32].

Release Offset

If the release offsets [8] of higher priority tasks are changed then preemptions can be avoided [63], but it can potentially bring other additional preemptions. In the paper by Belwal and Cheng [8], the authors illustrate the idea by an example. In Table 2.18, the task-set has two tasks and contains five columns: task, period, WCET, release offset and priority.

Task	Period	WCET	Release offset	Priority
τ_a	8	4	0	2
τ_b	12	3	3	1

Table 2.18: A task-set from the paper [32].

Firstly, the authors note that the task-set is not schedulable, as shown in Figure 2.35. Both τ_a and τ_b release together but τ_b has 3 release offset then τ_a executes first. τ_a is aborted at 3 because τ_b begins its job and finishes at 6. τ_a is restarted as new then it misses the deadline.

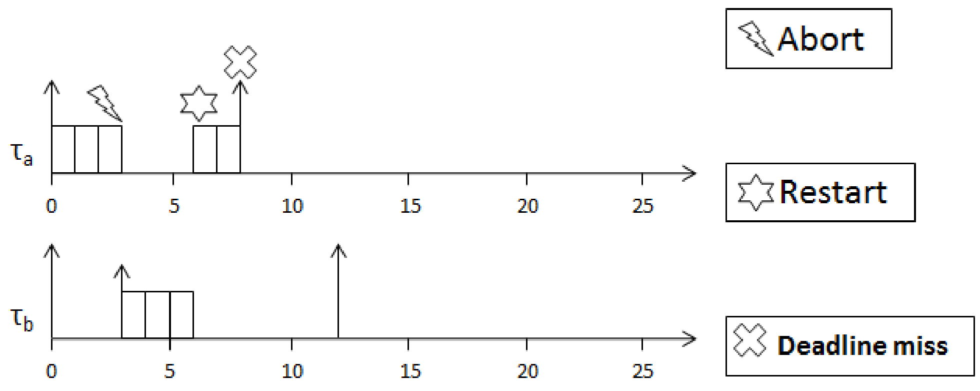

Figure 2.35: The task-set is not schedulable.

In Figure 2.36, the authors changed the release offset of τ_b to 0 so now τ_b executes first and finishes at 3. τ_a starts at 3 and ends at 7. The second job of τ_a arrives and executes at 8 then it is completed at 12. In the meantime, τ_b arrives and finishes at 15. The last job of τ_a arrives at 16 and is done at 20. The task-set with this modification is now schedulable.

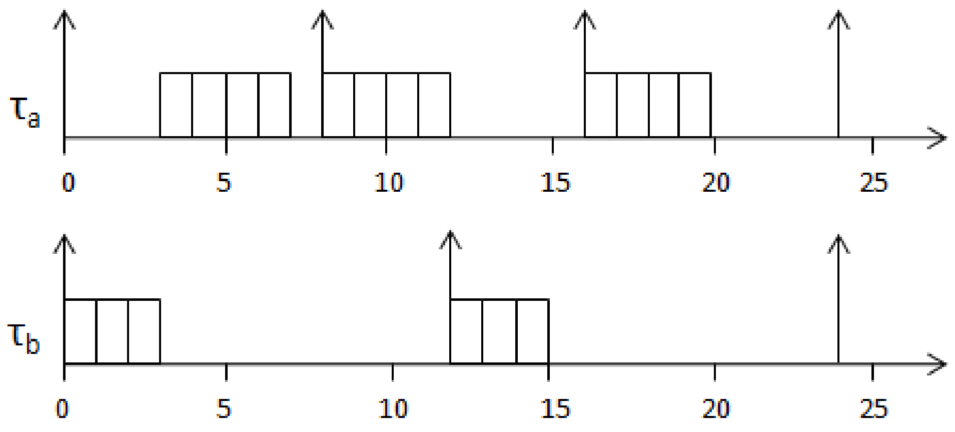

Figure 2.36: The task-set is now schedulable after removing the offset.

The objective of this example is that the authors illustrate the concept of release offset of tasks. They provide an unschedulable task-set because of 3 release offset and then make it schedulable by changing the release offset to 0. In other words, they removed the release offset from τ_b. This example is far-fetched although it is correct. We believe that a good example for release offset of tasks should switch a task-set from unschedulable to schedulable by adding release offsets rather than removing release offsets.

Afterwards, the authors [8] presented another example that if the priorities of τ_a and τ_b are reversed, the task-set also becomes schedulable, as shown in Figure 2.37. The example is about changing the priority of tasks, which is not related to the three conditions. There is also no clear explanation for the example. We believe that this example adds little to the paper[8].

The idea of release times of individual jobs [8] is that if the system has two tasks, $\Gamma_2 = \tau_i, \tau_j$ and $P_i > P_j$, and the finish time of the lower priority task is later than the release time of the higher priority task, then a preemption (abort) will occur. In the paper by Dobrin and Fohler [32], they can eliminate

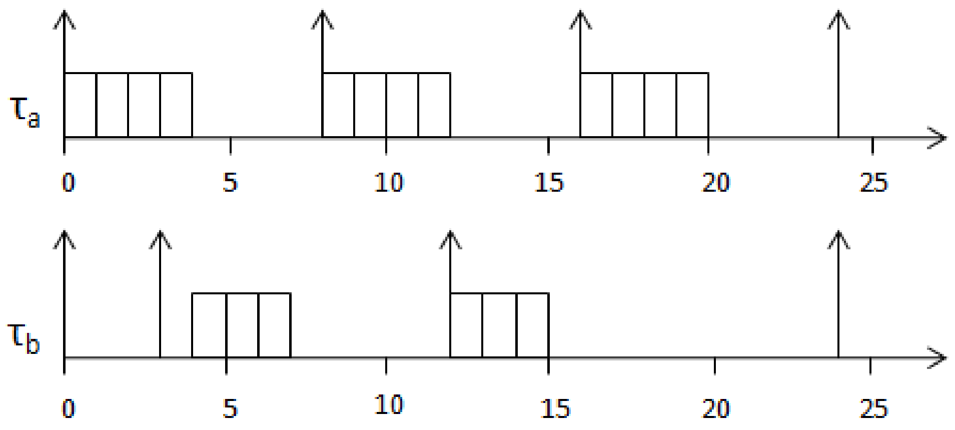

Figure 2.37: The time chart for the task-set.

the preemption by changing the release time of the higher priority task. Belwal and Cheng [8] state that a preemption will occur when the condition of the Equation (2.11) is true.

$$finish(\tau_{j,p}) > R_{i,q} \tag{2.11}$$

The notation of $finish(\tau_{j,p})$ means the finish time of τ_j for the p-th job. Also, they cited Equation (2.12) from the paper by Dobrin and Fohler [32] to show that a preemption can be removed by changing the release time of the higher priority task.

$$R_{i,q} = finish(\tau_{j,p}) - C_i \tag{2.12}$$

The authors [32] explain that if a higher priority task, τ_i^m, arrives at the time between start(τ_j^n) and finish(τ_j^n), τ_j^n will suffer a preemption from τ_i^m. The period between start(τ_j^n) and finish(τ_j^n) is termed as p_block(τ_j^n). To remove the preemption, τ_i^m is moved to the last part of p_block(τ_j^n). The task τ_j^n will finish earlier with no interrupt from τ_i^m.

So far the discussion is about offline analysis and assumption for WCET. Dobrin and Fohler [32] mentioned that the actual computation times of tasks are usually less than WCET at runtime so additional preemptions can occur. There is a diagram shown in Figure 2.38. As the diagram shows, the system runs perfectly at the stage of offline analysis, but there is an abort for the low priority task at runtime because the finish time of the high priority task is shorter.

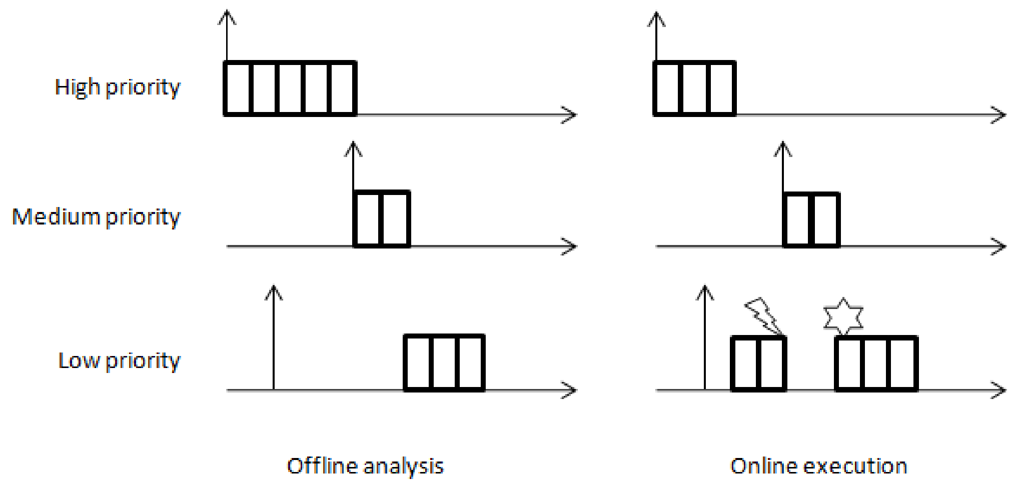

Figure 2.38: The differences between offline and online.

Although the task-set is still schedulable, the number of aborts is increased. When doing a software or hardware simulation for the AR model, we should consider the computation times as well. The priorities of tasks are also important for reducing the number of aborts.

Modifications in the scheduler

After discussing the modifications in task attributes which keep the basic FPS mechanism, we now look at the modifications in the scheduler: that if the preemption policy of the scheduler is amended, preemptions can be removed as well [8]. Preemption Threshold and Deferred Preemptions (DP) are discussed in the paper by Belwal and Cheng [8].

Preemption Threshold means that tasks execute in a non-preemption policy if the system is assigned with a preemption threshold and the priorities of those tasks are equal to or higher than the threshold. Wang and Saksena [83] have already found the response time analysis for tasks with a preemption threshold. Belwal and Cheng [32] believe that the schedulability of a task-set can be affected by a preemption threshold so the threshold number should be decided carefully.

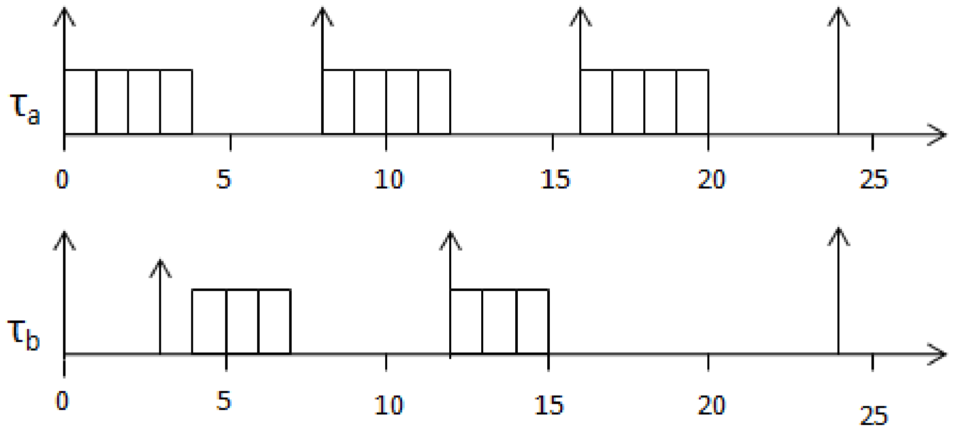

Figure 2.39: There is a deferred preemption occurring at 3. (using the task-set in Table 2.18)

DPs are similar in that if a lower priority task has already executed for a predefined time unit, the system will defer the preemption when a higher priority task arrives. So the higher priority starts after the lower priority is completed. There is an example from the paper [32] presented in Figure 2.39. In the diagram, task τ_b has higher priority but it cannot start at 3 because the system uses the DP policy. By the advantage of DPs, task τ_a did not waste the time spent on the execution before τ_b arrived, but τ_b still met its deadline. We will consider the application of DP in the AR model in Chapter 5.

2.4 Summary

To summarise, a synchronous release of tasks does not lead to a critical instant in P-FRP. All researchers from the reviewed papers used synchronous releases of tasks for their examples. The output of their examples is either inaccurate or over-pessimistic.

For the review of response time analysis for P-FRP: the AR cost is too pessimistic so the quality of schedulability analysis for the AR cost equation is very low. Time-Accurate simulation is a simple way to compute the actual response time but the efficiency is low. Gap-Enumeration Method is an improved implementation but an RB-tree is hard to program. Idle-Period Game Board Algorithm is a modified version of the Gap-Enumeration Method to simplify in programming. All are inefficient critical instant. RM priority assignment is not optimal in P-FRP. U-RM priority assignment is optimal in some cases but that is only with a synchronous release of tasks.

In the case studies for the AR model, the software case study provides the result that the current technique of P-FRP performs worse than other methods which also deal with the priority inversion problem. And dynamic priority scheduling seems to be better than static priority scheduling. The hardware case study shows that the number of aborts reduces the schedulability for P-FRP because each abort results in overheads in the system.

The review outcome of the methods to reduce aborts in P-FRP is that we are concerned about that number of aborts because it is a factor of overheads and also decreases the schedulability for the AR model. The material reviewed is from two papers, with the paper by Belwal and Cheng [8] focusing on P-FRP, and the paper by Dobrin and Fohler [32] on the classical preemptive model.The findings from the paper [8] are that the reduction of aborts can be implemented by two methods: modifications in the task attributes and the scheduler.

For modifications in task attributes, the priority reassignment for individual jobs expands a task-set by breaking jobs of tasks into individual tasks, and those tasks are assigned with suitable priorities to avoid aborts. We realise that the size of table is expanded, which is also a potential issue, and the method is similar to the way of dynamic priority scheduling but restricted to periodic tasks only [32]. The advantage is that the basic FPS mechanism can be kept. Release offset of tasks is to remove preemptions by changing the release offset of tasks. The idea of release times of individual jobs is that the higher priority task is postponed in its release if it arrives at a time during the busy period of a lower priority task. And also, we note the differences between offline analysis and online execution.

For modifications in the scheduler, the preemption threshold is to set a border for tasks which can be executed in a non-preemptive way. Other tasks need to suffer preemptions. The last approach is DP that a lower priority task can keep executing if it is close to completion. This latter approach seems the most effective and is referred to in Chapter 5.

Chapter 3

Schedulability Analysis for the AR Model

In this chapter we analyse the schedulability of the AR model, which consists of finding the critical instant and developing schedulability tests. Finding the critical instant for the AR model is analysed in the first section. Afterwards a new sufficient test for the AR model is derived. The analysis and findings in this chapter have been published as a technical report [85] and a workshop paper [84].

3.1 Critical Instant for the AR Model

First we consider periodic tasks and then sporadic tasks. In the AR model, a critical instant occurs when a higher priority task aborts a lower priority task which is almost completed, because the abort cost is added to the response time. For 2-tasks task-sets, there is only one case where the highest priority task aborts the lowest priority task.

This was illustrated in an earlier example (in Table 2.8 and Figure 2.10). For 3-tasks task-sets, there are two cases as the highest priority task can abort either of the two lower priority tasks. To generalise:

Lemma 3.1.1. *A task-set with N periodic tasks under the AR model has at least (N-1)! abort combinations.*

Proof. Consider a pure periodic task-set $\Gamma_N = \{\tau_1, \tau_2, ..., \tau_n\}$ and all tasks only release once. The highest priority task is τ_1 and the lowest priority task is τ_n. Task τ_1 has N - 1 choices of lower priority tasks to abort in each of their cases; τ_2 has N - 2 choices of lower priority tasks to abort. This continues until τ_{n-1} which has only one choice to abort. Finally, τ_n has zero choices because there is no lower priority task. When higher priority tasks are released more than once, the number of choices for those tasks is increased. The number of abort combinations is therefore at least $(N-1) * (N-2) * ... * 1$, which is (N-1)!. □

There is no information within the task-set that would indicate which set of abort combinations could give rise to the worst-case response times. Hence they all need to be checked for exact analysis.

For sporadic tasks, there is a further issue to consider.

Lemma 3.1.2. *A sporadic task with an arbitrary release may bring a longer response time.*

Proof. In general, a sporadic task with its maximum arrival rate delivers the worst-case response time. Lemma 3.1.2 can be proved by showing a counter example. In Table 3.1, there is a 3-tasks task-set. Task τ_1 is a sporadic task and has the highest priority. It has a minimum inter-arrival time, 8. Other tasks are periodic tasks.

Table 3.1: A task-set with a sporadic task.

Task	Period	WCET	Priority
τ_1	8	1	1
τ_2	20	2	2
τ_3	40	4	3

In Figure 3.1, the response time of τ_3 is 16 when the second job of τ_1 is released with the minimum inter-arrival time, 8.

If, however, the second job of τ_1 is released 1 tick later, the response time of τ_3 will be 17. In this condition, a sporadic task with a later release may bring a longer response time. □

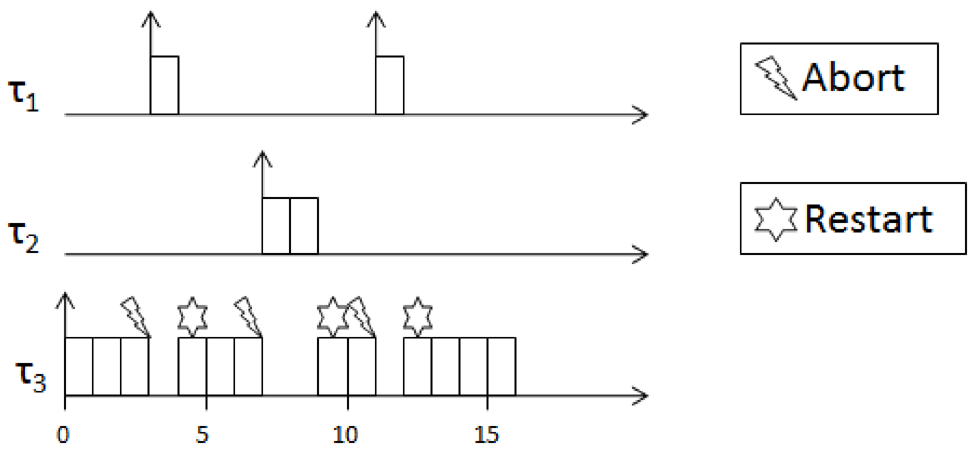

Figure 3.1: A time chart.

For a set of sporadic tasks exact analysis would require all possible release times to be checked.

Theorem 3.1.3. *Finding the critical instant for the AR model with periodic and sporadic tasks is intractable.*

Informal Proof. Lemma 3.1.1 shows that there is at least $(N-1)!$ abort combinations for N periodic tasks, all of which must be checked for the worst case to be found. For sporadic tasks all possible release times over a series of releases must be checked to determine the worst-case impact of the sporadic task. These two properties in isolation and together show that this is an intractable number of release conditions to check in order to define the critical instant.

In real-time scheduling, a tractable schedulability test cannot be exact (sufficient and necessary) if the critical instant cannot be found in polynomial time.

3.2 New Formulation for Schedulability Tests

As an exact analysis for the AR model is intractable, a sufficient test is derived in this section. The sufficiency is traded with tractability, and this new test is more intuitive than those previously published.

Given a priority assignment, the worst-case response time of task τ_i (priority P_i) will depend only on the behaviour of tasks of priority greater than P_i. Consider the interference caused by a single release of task τ_j ($P_j > P_i$). In the worst case τ_j will abort (just before it completes) a task with a lower priority than τ_j but with the maximum execution time of all lower priority tasks. Let the aborted task be τ_k, so $P_j > P_k \geq P_i$ and $C_k = \max_{\forall k \in hep_i \bigcap lp_j} C_k$.

The impact of τ_j on τ_i will therefore be, in the worst case, C_j at priority P_j and C_k at priority P_k. As $P_k \geq P_i$ this is equivalent (for τ_i) to τ_j having an execution time of $C_j + C_k$ at priority P_j. Let $\tilde{C}_j^i = C_j + C_k$. The original task-set with computation times C_j is transposed into a task-set with $\tilde{C}_j^i$. This is now a conventional task-set, so the critical instant is when there is a synchronous release. (The maximum interference on τ_i must occur when all higher priority tasks arrive at their maximum rate, initially at the same time, and all have their maximum impact.)

The worst case for the AR model is that any higher priority task aborts a lower priority task which has the biggest possible WCET, and that this abort occurs just before the aborted task would actually complete. By this process, a new value $\tilde{C}_j^i$ for τ_j is obtained by combining C_j and C_k:

$$\tilde{C}_j^i = C_j + \max_{\forall k \in hep_i \bigcap lp_j} C_k \tag{3.1}$$

where $\tilde{C}_j^i$ is the new value for the WCET of τ_j, C_j is the original WCET of τ_j and C_k is the biggest execution time of a task with priority between τ_i and τ_j but τ_j is not included. The response time analysis applies to τ_i. Note that in general the $\tilde{C}_j^i$ values will depend on the task under investigation.

In Table 3.2, there is an example task-set. Deadline is equal to period and the time unit is a tick. The highest priority is 1. The response time of task τ_4 is being computed.

The $\tilde{C}_j^4$ values are computed by Equation (3.1). In this example we consider the response time for τ_4 so $i = 4$. For $\tilde{C}_1^4$, j is 1 and C_k is higher than or equal to τ_4 but lower than τ_1. The calculation is $\tilde{C}_1^4 = C_1 + C_4$, so the result of $\tilde{C}_1^4$ is $2 + 5 = 7$.

For $\tilde{C}_4^4$, i and j are 4. C_k is higher than or equal to τ_4 but lower than τ_4

Table 3.2: An example with new WCETs for 4-tasks task-set.

Task	Period	C	$\tilde{C}_j^4$	Priority
τ_1	28	2	7(2+5)	1
τ_2	120	3	8(3+5)	2
τ_3	140	4	9(4+5)	3
τ_4	200	5	5(5+0)	4

so no task is matched, so the result of $\tilde{C}_4^4$ is $5+0=5$. After all the $\tilde{C}_j^4$ values had been calculated, we used those values instead of C for the response time analysis; that is:

$$R_4 = \tilde{C}_4^4 + \sum_{\forall j \in hp_4} \left\lceil \frac{R_4}{T_j} \right\rceil \cdot \tilde{C}_j^4 \tag{3.2}$$

This is solved in the usual way by forming a recurrence relationship. The calculations are as follows:

1. $R_4^1 = 5 + (\lceil \frac{5}{28} \rceil \cdot 7 + \lceil \frac{5}{120} \rceil \cdot 8 + \lceil \frac{5}{140} \rceil \cdot 9) = 29$
2. $R_4^2 = 5 + (\lceil \frac{29}{28} \rceil \cdot 7 + \lceil \frac{29}{120} \rceil \cdot 8 + \lceil \frac{29}{140} \rceil \cdot 9) = 36$
3. $R_4^3 = 5 + (\lceil \frac{36}{28} \rceil \cdot 7 + \lceil \frac{36}{120} \rceil \cdot 8 + \lceil \frac{36}{140} \rceil \cdot 9) = 36$

To compare the result with the equation of Ras and Cheng [69] (given in Section 2.3), their calculation would be:

1. $R_4^1 = 5+(\lceil \frac{5}{28} \rceil \cdot 2 + \lceil \frac{5}{120} \rceil \cdot 3 + \lceil \frac{5}{140} \rceil \cdot 4) + \lceil \frac{5}{28} \rceil \cdot 5 + \lceil \frac{5}{120} \rceil \cdot 5 + \lceil \frac{5}{140} \rceil \cdot 5 = 29$
2. $R_4^2 = 5+(\lceil \frac{29}{28} \rceil \cdot 2 + \lceil \frac{29}{120} \rceil \cdot 3 + \lceil \frac{29}{140} \rceil \cdot 4) + \lceil \frac{29}{28} \rceil \cdot 5 + \lceil \frac{29}{120} \rceil \cdot 5 + \lceil \frac{29}{140} \rceil \cdot 5 = 36$
3. $R_4^3 = 5+(\lceil \frac{36}{28} \rceil \cdot 2 + \lceil \frac{36}{120} \rceil \cdot 3 + \lceil \frac{36}{140} \rceil \cdot 4) + \lceil \frac{36}{28} \rceil \cdot 5 + \lceil \frac{36}{120} \rceil \cdot 5 + \lceil \frac{36}{140} \rceil \cdot 5 = 36$

The results are the same but Equation (3.2) clearly involves less computation.

To compute the worst-case response time for τ_3 requires the $\tilde{C}_j^3$ values to be recomputed (as show in Table 3.3).

The test derived above is more efficiently solved is nevertheless equivalent to that previous published.

Table 3.3: $\tilde{C}_j^3$ values for τ_3

Task	Period	C	$\tilde{C}_j^3$	Priority
τ_1	28	2	6(2+4)	1
τ_2	120	3	7(3+4)	2
τ_3	140	4	4(4+0)	3

Theorem 3.2.1. *Equations (2.8) and (3.2) are equivalent.*

Proof. We rephrase Equation (2.8) as below:

$$R_i = C_i + \sum_{\forall j \in hp_i} \left\lceil \frac{R_i}{T_j} \right\rceil \cdot C_j + \sum_{\forall j \in hp_i} \left\lceil \frac{R_i}{T_j} \right\rceil \cdot \max_{k=i}^{j-1} C_k \tag{3.3}$$

and simplify:

$$R_i = C_i + \sum_{\forall j \in hp_i} \left\lceil \frac{R_i}{T_j} \right\rceil \cdot C_j + \max_{k=i}^{j-1} C_k \tag{3.4}$$

both $\max_{k=i}^{j-1} C_k$ and $\max_{\forall k \in hep_i \bigcap lp_j} C_k$ are to pick a bigger WCET task when priority is higher or equal to τ_i and lower than τ_j, so we rephrase it again.

$$R_i = C_i + \sum_{\forall j \in hp_i} \left\lceil \frac{R_i}{T_j} \right\rceil \cdot C_j + \max_{\forall k \in hep_i \bigcap lp_j} C_k \tag{3.5}$$

Equation (3.1) changes into Equation (3.5) as below:

$$R_i = C_i + \sum_{\forall j \in hp_i} \left\lceil \frac{R_i}{T_j} \right\rceil \cdot \tilde{C}_j^i \tag{3.6}$$

Finally, $\tilde{C}_j^i$ replaces C_i using Equation (3.1). $\square$

As Equation (2.8) was previously proved to be sufficient for the AR model [69] it follows that Equation (3.2) is similarly sufficient.

Although the equations are equivalent, Equation (3.2) is in the standard form for response time analysis and is therefore amenable to the many ways that have been found to efficiently solve this form of analysis [78]. It is also in a form that allows the issue of priority assignment to be addressed (see next chapter).

3.3 Summary

To summarise, the AR model has a different property of critical instant from the classical preemptive model. A task-set with N periodic tasks under the AR model has at least (N-1)! abort combinations. A sporadic task with an arbitrary release may bring a longer response time. Finding the critical instant for the AR model is intractable as it cannot be found in polynomial time. To deal with this situation, a sufficient test is derived by trading the sufficiency with tractability. A new formulation, $\tilde{C}_j^i$, is developed for the standard response time analysis as a new sufficient test for the AR model.

Chapter 4

Improved Priority Assignment for the AR Model

This chapter introduces a new algorithm which offers better priority assignment for the AR model. Firstly, some priority assignment schemes are addressed, as they are needed for the new algorithm and the experiments. The new algorithm with pseudo-code is given, and the time complexity is discussed. Lastly, the new algorithm and other addressed algorithms are compared and evaluated in an experiment. The material in this chapter has been published as a technical report [85] and a workshop paper [84].

4.1 Priority Assignment Schemes

Rate Monotonic (RM) and Utilisation Monotonic (UM) have been introduced as possible priority assignment schemes for the AR model. Here, we introduce another priority assignment called Execution-time Monotonic (EM), which assigns a higher priority to a task which has a bigger WCET[1]. An inspection of Equation (3.1) shows that the minimum execution times (the $\tilde{C}_j^i$ values) are obtained when priority is ordered by execution time. Although this does not necessarily minimise utilisation, it may furnish an effective pri-

[1]Tasks with the same execution time are ordered by deadline (higher priority to shorter deadline). If they also have the same deadline then the shortest period is used to break the tie.

ority assignment scheme. For example, if the biggest WCET task is assigned to the lowest priority then all higher priority tasks will abort that task. The consequence is the total abort cost gets much bigger.

For many scheduling problems, Audsley's algorithm furnishes an optimal priority assignment; i.e. the algorithm can find a schedulable priority ordering if such an ordering exists [5, 6]. Unfortunately one of the prerequisites for Audsley's algorithm does not hold (see Section 2.2.3). Specifically, the response time of a task depends not only on the set of higher priority tasks but also on their relative order (which is not permitted).

Table 4.1: The response time of τ_4 is 23.

Task	Period	WCET	$\tilde{C}_j^4$	Priority
τ_1	100	5	9(5+4)	1
τ_2	120	4	7(4+3)	2
τ_3	140	3	5(3+2)	3
τ_4	200	2	2(2+0)	4

In Table 4.1 τ_4 is the lowest priority task and its response time is 23. After τ_2 and τ_3 (higher priority) swapped their priorities, the response time for τ_4 is changed to 24 as shown in Table 4.2.

Table 4.2: The response time of τ_4 is 24.

Task	Period	C	$\tilde{C}_j^4$	Priority
τ_1	100	5	9(5+4)	1
τ_3	140	3	7(3+4)	2
τ_2	120	4	6(4+2)	3
τ_4	200	2	2(2+0)	4

4.2 New Algorithm

The Exhaustive Search (ES) algorithm is optimal for any model but the complexity is the factorial of the number of tasks. Therefore, it is not applicable in general but it can validate other algorithms for small values of N. By comparison with ES, both UM and EM are not optimal. Sometimes, there is more than one schedulable ordering for a task-set. Some tasks are scheduled by EM but not UM, and vice versa. Their relationship is shown in Figure 4.1.

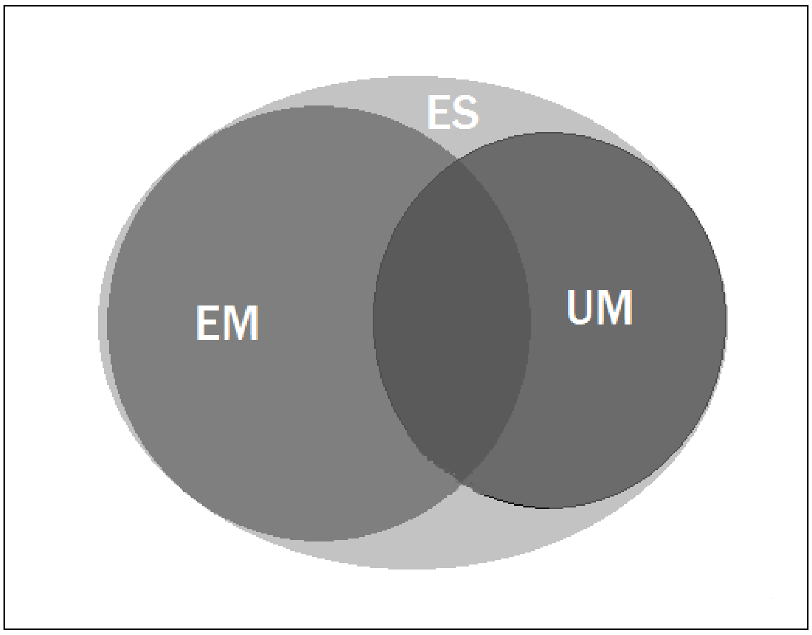

Figure 4.1: An Euler diagram for UM, EM and ES.

These circles represent task-sets that are scheduled by the labelled algorithms. White space is task-sets that are unscheduled by any algorithm. ES covers both UM and EM because it is optimal. UM and EM are overlapped because some task-sets are scheduled by both of them. In a later section, the experiments show that UM and EM have similar results. If an algorithm dominates both UM and EM, it will offer a better schedulability rate.

We derive a new algorithm that starts with EM ordering and tests the tasks in priority order, starting with the highest priority task. If any task cannot be scheduled then we try to find a higher priority task which has less utilisation. The ordering begins from the failed task to the top. If a task is found then shift down the higher priority task below the lower priority task. If no task is found, the task-set is deemed to be not schedulable. Pseudo-code of the new algorithm is given in Algorithm 1.

The explanations of functions are listed below:

```
sortByEM(taskset);
for $i$ in $2$ .. $N$ do
    if $st(i) = true$ then
        continue;
    else
        for $j$ in $i$-$1$ .. $1$ do
            if $u(j) < u(i)||(u(j) = u(i)\&d(j) > d(i))$ then
                found = true;
                move(j,i);
                i=j-1;
            else
                found = false;
            end
            if not found then
                return fail;
            end
        end
    end
end
return pass;
```

Algorithm 1: A pseudo-code of the new algorithm.

- sortByEM(ts) = do an EM priority assignment for task-set ts
- st(i) = Schedulability Test for task i
- u(i) = get utilisation of task i
- d(i) = get deadline of task i
- move(x, y) = move task x below task y (top is higher priority).

An example of the use of the algorithm is given in Table 4.3. Again deadline is equal to period; R is response time. Note only C values are given in the table; the necessary $\tilde{C}_j^i$ values are dependent on which task is actually been tested, and they must be recomputed for each task.

The task-set is initially ordered by the EM algorithm. The schedulability test begins from the top. τ_1, τ_2 and τ_3 meet their deadlines. A missed deadline occurs at τ_4 so the algorithm searches for a less utilisation task from

Table 4.3: An example task-set fails in EM ordering.

Task	Period	C	U	Priority	R
τ_1	60	6	0.1	1	6
τ_2	50	5	0.1	2	16
τ_3	32	4	0.125	3	24
τ_4	25	3	0.12	4	30 (X)
τ_5	100	2	0.02	5	

τ_3 to τ_1. The utilisation of τ_2 is 0.1 which is less than τ_4, and τ_2 shifts down below τ_4. The priority of τ_3 shifts up to 2. The priority of τ_4 shifts up to 3. The priority of τ_2 changes to 4.

Table 4.4: The task-set is scheduled by EUM algorithm.

Task	Period	C	U	Priority	R
τ_1	60	6	0.1	1	6
τ_3	32	4	0.125	2	14
τ_4	25	3	0.12	3	20
τ_2	50	5	0.1	4	50
τ_5	100	2	0.02	5	88

In Table 4.4, the task-set has had its priorities changed and is schedulable after the shifting. By the nature of shifting down the less utilisation tasks to the bottom, UM ordering is the worst case. The algorithm performs a set of transformations starting at EM and moving towards UM. It dominates both EM and UM. We name it the Execution-time-toward-Utilisation Monotonic (EUM) priority assignment scheme.

Figure 4.2 is an Euler diagram for EUM where the circle of EUM covers all circles of EM and UM.

4.3 Time Complexity

To analyse the complexity of the EUM policy, we count each single task schedulability test required (each test is itself of pseudo-polynomial complexity). In the worst case, an N-task task-set starts with EM ordering and

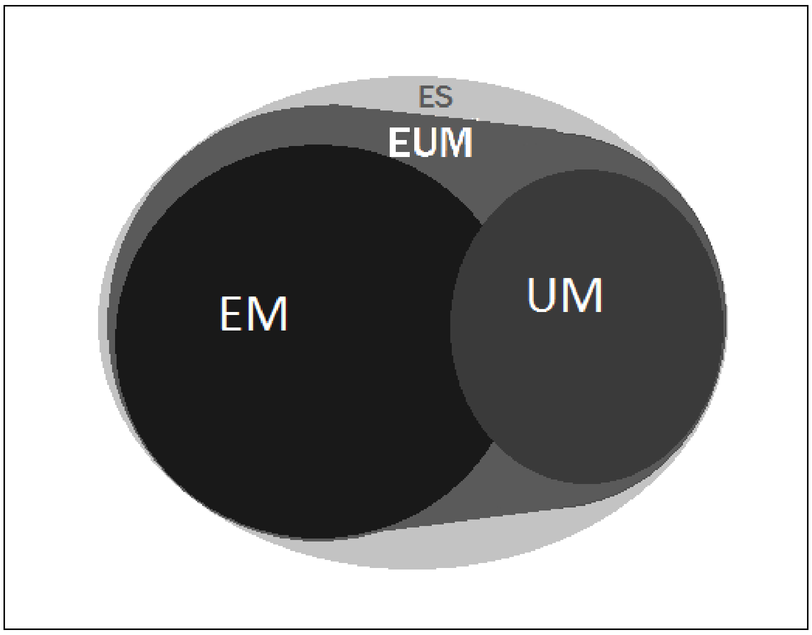

Figure 4.2: An Euler diagram for EUM.

the task-set is only scheduled by UM ordering, which is the completely opposite to EM. It is easy to see that in this case, $2N-1$ schedulability tests are required before the task that starts out at priority N is placed at priority 1, and that a further $2(N-1)-1$ tests are needed before the next task (that started at priority $N-1$) is placed at priority 2. Overall, the number of single task schedulability tests required to transform EM ordering into UM ordering is given by:

$$\sum_{k=1}^{N-1}(2k-1) \leq N^2 \tag{4.1}$$

So the complexity of EUM priority assignment is $O(N^2)$ single task schedulability tests. EUM dominates EM and UM because the EUM algorithm starts with EM ordering and ends at UM ordering in the worst case; however, unlike ES it is a tractable priority assignment policy.

4.4 Experimental Evaluation

The experiments undertaken are separated into two parts. First, the EUM algorithm is compared with the non-optimal group (RM, UM and EM). Secondly, the EUM algorithm is compared with ES, the optimal algorithm for the largest feasible value for N (8). All experiments used the same parameters but different priority assignments. The parameters are:

- Deadline is equal to period.
- All tasks are periodic (or sporadic arriving at their maximum rate).
- A set of N utilisation values U_i was generated by the UUniFast Algorithm [13].
- Task periods were generated between 500 and 5000 according to a log-uniform distribution[2]. And the computed value T_i is assigned to τ_i.
- Task execution times are: $C_i = U_i \cdot T_i$
- Utilisation for task-sets are ranged between 10% and 70%.
- Each utilisation rate generates 10000 different task-sets, i.e. $U = 10\%$ generates 10000 task-sets, $U = 11\%$ generates another 10000 task-sets, and so on.
- The numbers of tasks for the non-optimal group are 5, 10, 15 and 20. A maximum of 8 tasks is all that can be accomplished by ES. The final experiment is therefore restricted to just 8 tasks.

For all diagrams, the X-axis is Utilisation rate and the Y-axis is the Schedulability rate, i.e. the percentage of task-sets that were deemed schedulable.

In Figure 4.3 the number of tasks is 5. We observe that RM has the worst schedulability, and UM and EM are quite similar before $U = 27\%$.

[2]The log-uniform distribution of a variable x is such that ln(x) has a uniform distribution.

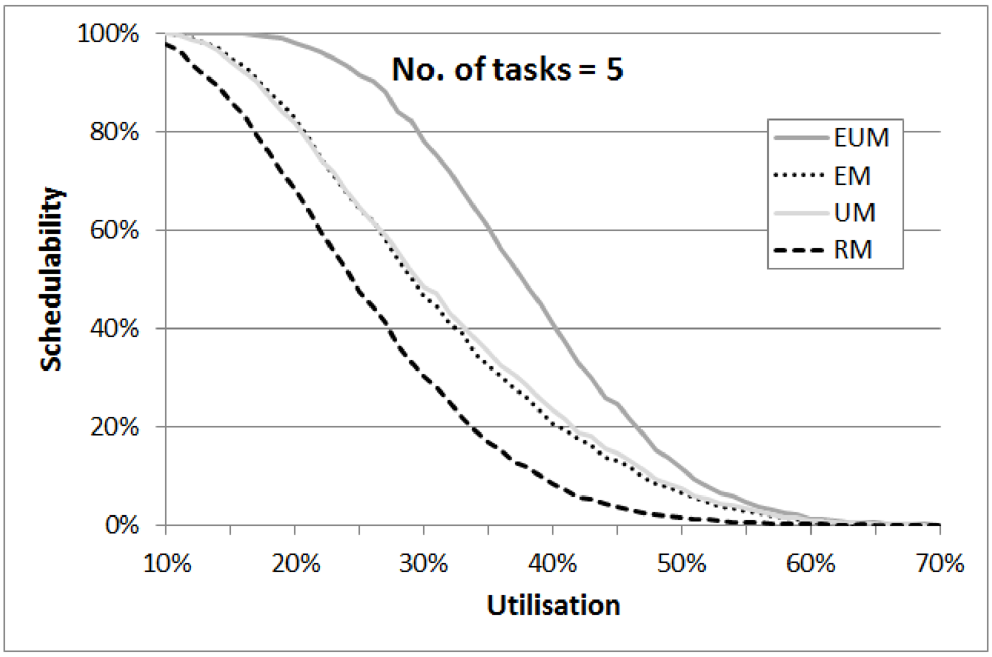

Figure 4.3: EUM compares with others for 5-tasks task-set.

After that, UM is better than EM. EUM is of course always better than the others and is significantly so. The maximum improvement is between 30% and 40% in utilisation.

In Figure 4.4 the number of tasks is 10; RM is still the worst and EM is better than UM.

In Figures 4.5 and 4.6 the numbers of tasks is 15 and 20. Again RM is the worst; EM is better than UM and EUM is the best. The two diagrams have a similar pattern. Results for the larger value of N are similar (but not included).

For the final comparing experiment of EUM and ES, ES is the factorial of the number of tasks so we picked the number of tasks to be as large as possible. In Figure 4.7 the number of tasks is 8 because the test has already taken more than a week to run. The diagram shows the result that EUM is very close to ES. Indeed it is impossible to distinguish between them in the diagram. Nevertheless EUM is not optimal; the figure contains in total 410,000 task-sets of which ES deemed 137,366 schedulable and EUM 136,712, a difference of 654 (i.e. schedulable by ES but not by EUM). Tables 4.5 and

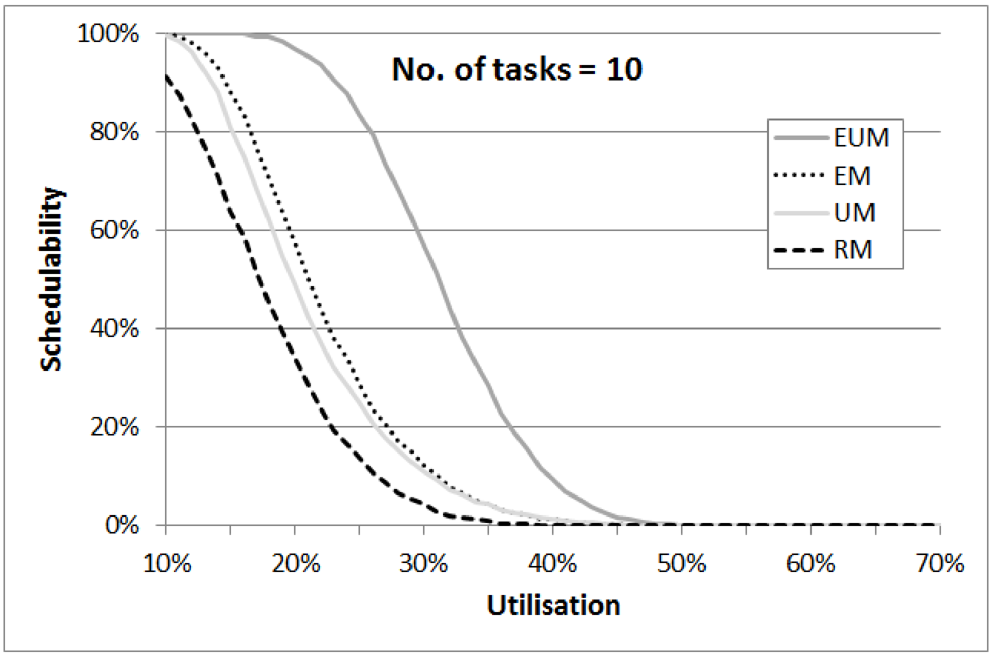

Figure 4.4: EUM compares with others for 10-tasks task-set.

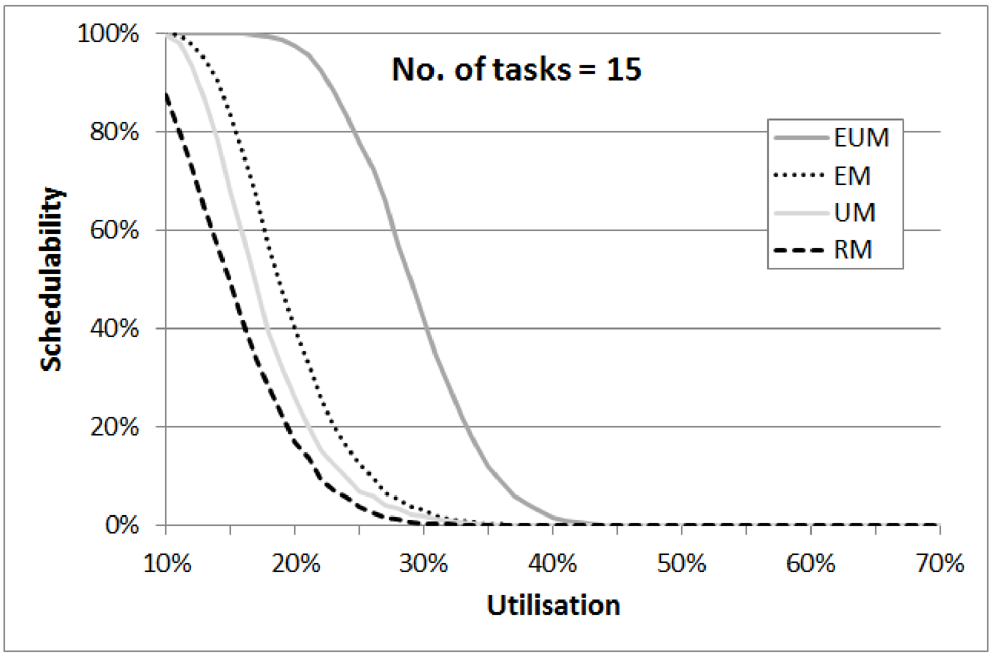

Figure 4.5: EUM compares with others for 15-tasks task-set.

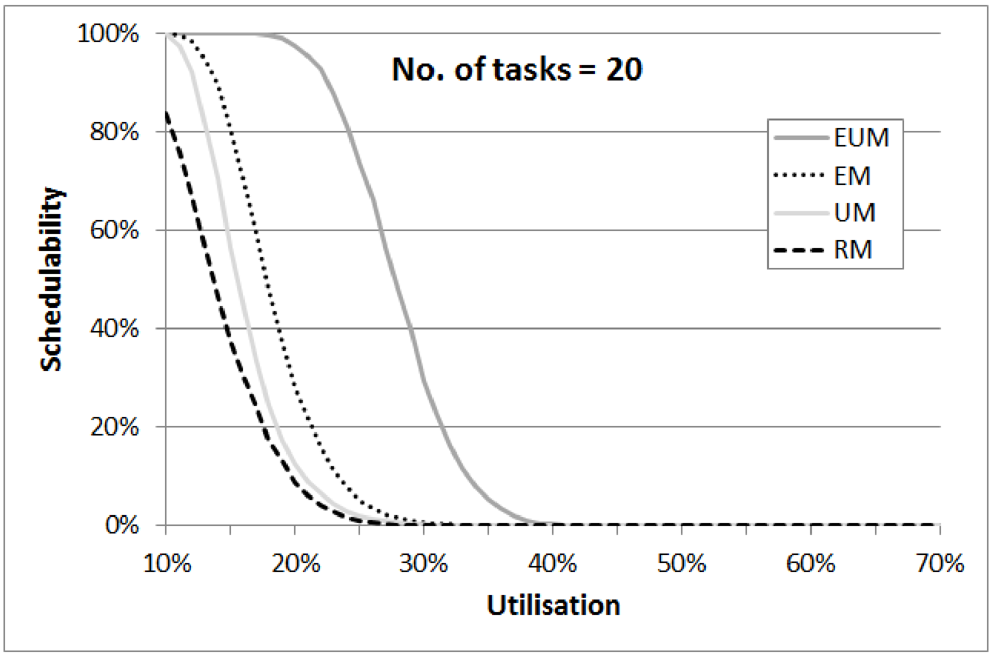

Figure 4.6: EUM compares with others for 20-tasks task-set.

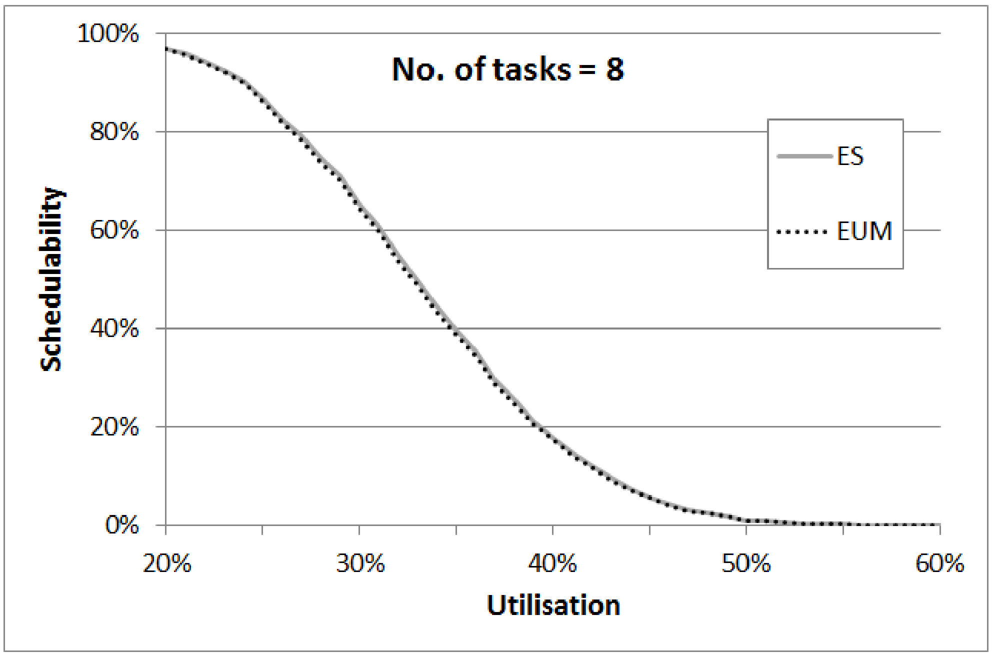

Figure 4.7: EUM compares with ES for 8-tasks task-set.

Table 4.5: A task-set deemed not schedulable by EUM algorithm.

Task	Period	C	U	Priority	R
τ_3	1430	179	0.125	1	179
τ_6	1035	90	0.087	2	359
τ_2	656	49	0.075	3	457
τ_5	1269	27	0.021	4	511
τ_7	1925	131	0.068	5	X
τ_4	2579	31	0.012	6	X
τ_1	2688	8	0.003	7	X
τ_8	1042	7	0.007	8	X

Table 4.6: The task-set is schedulable by ES algorithm.

Task	Period	C	U	Priority	R
τ_7	1925	131	0.068	1	131
τ_3	1430	179	0.125	2	489
τ_2	656	49	0.075	3	587
τ_6	1035	90	0.087	4	947
τ_8	1042	7	0.007	5	961
τ_5	1269	27	0.021	6	1035
τ_4	2579	31	0.012	7	1264
τ_1	2688	8	0.003	8	1746

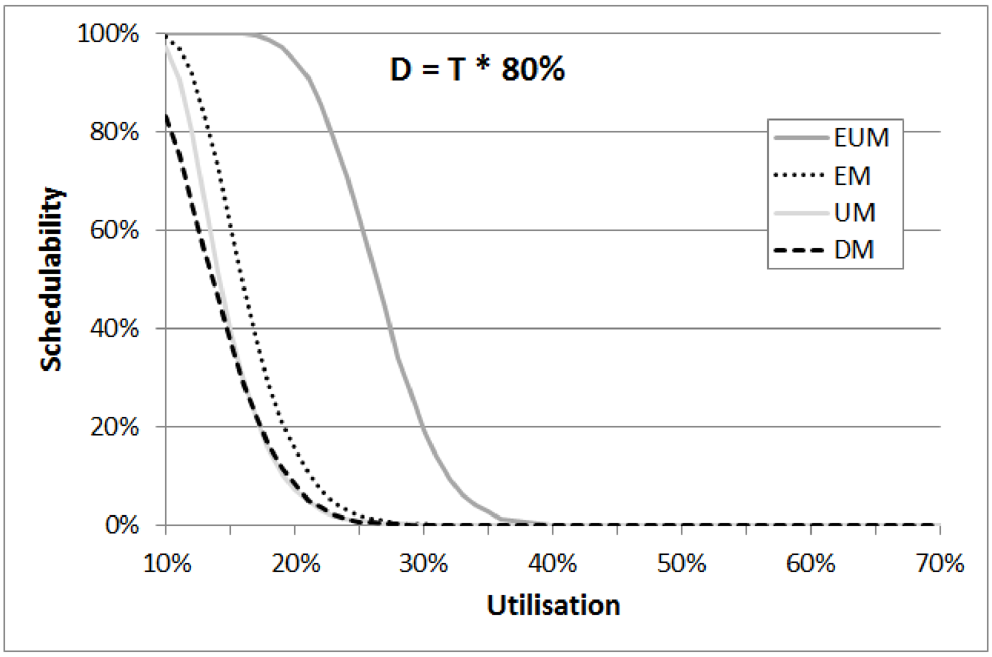

Figure 4.8: The number of tasks is 20 with D = T * 80%.

4.6 show an example task-set which is schedulable by ES but not by EUM. Although not exact, the performance of EUM for $N = 8$ leads to a reasonable conclusion that EUM is an effective and near optimal priority ordering for the AR model.

For a further analysis, Figures 4.8 and 4.9 show two examples when deadline is less than period. In Figure 4.8, the number of tasks is 20 and deadline is 80% of period. The schedulability of using the UM ordering is getting worse at 18% utilisation rate. In Figure 4.9, the number of tasks is also 20 but deadline is 50% of period. It shows that DM is much better than both UM and EM. EUM is still far better than the other priority assignments.

4.5 Summary

To sum up, optimal priority ordering is problematic with the AR model. Deadline (or Rate) monotonic ordering is demonstrably not optimal. Also the optimal Audsley's algorithm is not applicable. We have however developed a heuristic (called EUM) that performs well and has only N^2 complexity (for

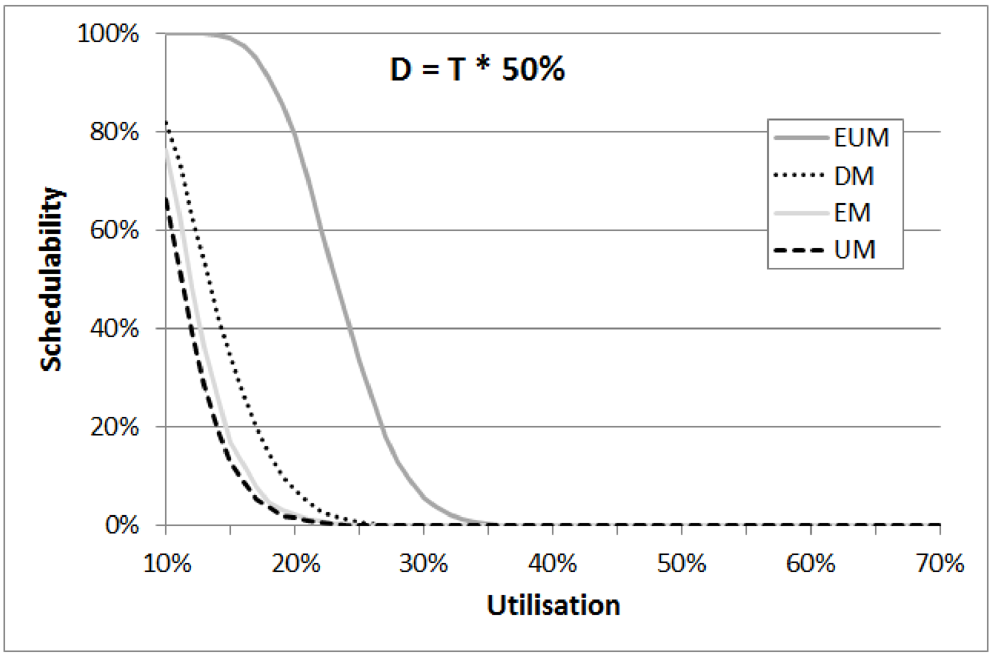

Figure 4.9: The number of tasks is 20 with D = T * 50%.

N tasks). On small sized systems ($N = 8$) EUM performs almost identically to an optimal scheme (using exhaustive search). For larger numbers of N (where exhaustive search is unfeasible) it performs much better than previous published approaches.

Chapter 5

Deferred Abort Model

This chapter introduces an alternative scheme to improve the schedulability of the AR model. Higher priority tasks sometimes do not need an immediate abort to meet their deadlines. For example, a higher priority job is released and an executing lower priority task is almost completed; the higher priority task has enough time to wait until the lower priority task finishes.

To implement this approach, the technique of DP noted in Chapter 2 is employed. One of the implementations of DP is that a task is assigned two regions; the first region is preemptive and the second region is non-preemptive. To apply this technique, a task is also set two regions; first region is AR and the second region is non-preemptive and non-abortable. In this thesis, we call this scheme the *deferred abort* (DA).

Before the DA model, the non-preemptive model is introduced and compared to the AR model. Afterwards, the technique of deferred preemption is discussed and used to develop response time analysis of the DA model. Lastly, experimental evaluation and conclusions are given.

5.1 Non-preemptive Model

To support P-FRP, atomic execution is required; hence the AR model is an implementation scheme but the Non-Preemptive (NP) model can be used as well. Below, the NP model will be introduced and compared with the AR model.

In the NP model, once a task, whatever its priority, is executing, no task can interrupt it. Figure 5.1 shows that if a higher priority task is released when a lower priority task is executing then the higher priority task has to wait until the lower priority task finishes.

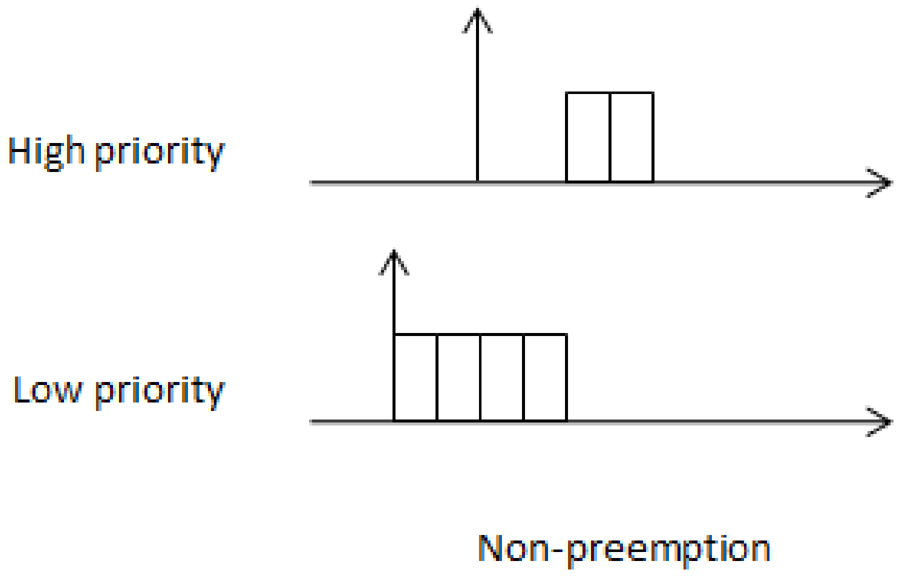

Figure 5.1: This is the non-preemptive model.

This figure shows the lower priority task executed atomically so the NP model can also be used for P-FRP. Now, we consider if the AR model can be replaced by the NP model completely. In Table 5.1, there is a 3-tasks task-set with constrained deadlines. The response time analysis in Equation (5.4), given later in this chapter, is used to analyse the task-set.

Task	T	D	C	Priority
τ_1	30	3	3	1(H)
τ_2	50	50	5	2
τ_3	70	70	7	3

Table 5.1: The NP model cannot schedule the task-set.

In this task-set, the execution time of τ_1 is equal to its deadline. τ_1 cannot suffer any blocking time or interference so the NP model cannot schedule this task-set.

To apply the AR model, the $\tilde{C}_j^i$ approach in Equation (3.1) and the response time analysis in Equation (3.6) are used. As the AR model does not have blocking time, the worst-case response time of τ_1 is 3. The worst-case response time of τ_2 is 13. Table 5.2 is the task-set with $\tilde{C}$ values for τ_3. The worst-case response time of τ_3 is 29. The task-set can be scheduled by the AR model.

Task	T	D	C	$\tilde{C}_j^3$	Priority
τ_1	30	3	3	10(3+7)	1(H)
τ_2	50	50	5	12(5+7)	2
τ_3	70	70	7	7(7+0)	3

Table 5.2: The AR model can schedule the task-set.

The calculations are as follows:

1. $R_1^1 = 3$
2. $R_2^1 = 5 + (\lceil \frac{3}{30} \rceil \cdot 8) = 13$, see footnote[1].
3. $R_2^2 = 5 + (\lceil \frac{13}{30} \rceil \cdot 8) = 13$
4. $R_3^1 = 7 + (\lceil \frac{7}{30} \rceil \cdot 10 + \lceil \frac{7}{50} \rceil \cdot 12) = 29$
5. $R_3^2 = 7 + (\lceil \frac{29}{30} \rceil \cdot 10 + \lceil \frac{29}{50} \rceil \cdot 12) = 29$

The above example shows that the NP model does not dominate the AR model. On the other hand, The AR model does not dominate the AR model. To illustrate, Table 5.3 shows that the AR model cannot schedule the task-set.

For the NP model, Table 5.4 shows the task-set can be scheduled.

The above examples show that the NP and AR models do not dominate each other. In the later section of experimental evaluation, the result shows

[1] $\tilde{C}_1^2 = 8(3+5)$

Task	T	D	C	$\tilde{C}_j^3$	Priority
τ_1	30	30	10	20(10+10)	1(H)
τ_2	30	30	10	20(10+10)	2
τ_3	30	30	10	10(10+0)	3

Table 5.3: The AR model cannot schedule the task-set.

Task	T	D	C	B	Priority
τ_1	30	30	10	10	1(H)
τ_2	30	30	10	10	2
τ_3	30	30	10	0	3

Table 5.4: The NP model can schedule the task-set.

that the NP model has better schedulability than the AR model. Intuitively, the combination of them is possible to offer better performance as there is an existing model called deferred preemption which combines the techniques of preemptive and non-preemptive. In the following section, deferred preemption will be introduced and discussed.

5.2 Deferred Preemption

The motivation is to reduce the number of preemptions by deferring some unnecessary preemptions. The definition of unnecessary preemption is that a higher priority task can wait until a lower priority task finishes and the higher priority task also meets its deadline. As DP is designed for non-atomic-execution systems, it cannot directly apply to P-FRP. Before adapting this technique for the AR model, DP is introduced and analysed.

To implement DP, there are a number of approaches. In this thesis, the approach from Davis and Bertogna's paper [26] is used. In this paper, a task can set the length of the final non-preemptive region. Symbol F is used to represent the length of the final non-preemptive region. The range of F_i for τ_i is from 1 to C_i. If $F_i = 1$, τ_i is fully preemptive[2]. If $F_i = C_i$, τ_i is fully

[2]The discrete time granularity Δ is 1 time unit so the last 1 time unit is deemed to be

non-preemptive. If $F_i = (C_i/2)$, the first half of τ_i is preemptive and the last half is non-preemptive.

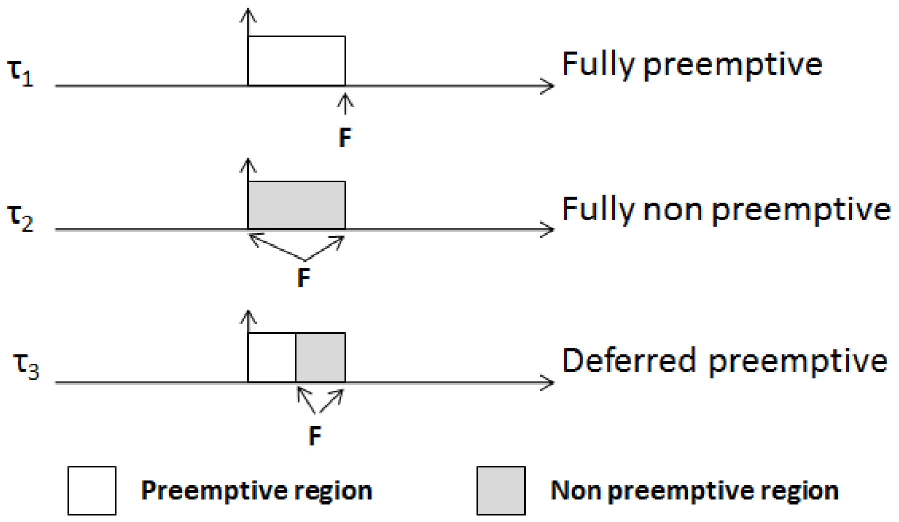

Figure 5.2: Types of deferred preemptive tasks.

To illustrate, Figure 5.2 shows 3 tasks: τ_1, τ_2 and τ_3. The white boxes represent preemptive regions and the grey boxes represent non-preemptive regions. τ_1 with a fully white box is fully preemptive and F points at the end. τ_2 with a fully grey box is fully non-preemptive and F is the entire box. τ_3 with a half white and half grey box is deferred preemptive and F points at the middle.

Now, we introduce some equations below to analyse the DP model. In Davis and Bertogna's paper [26], the authors studied the work of Bril et al. [16], and the results are rephrased according to the notation adopted in Buttazzo et al's paper [20] to deal with the discrete time domain. The concepts of priority level-i active period, and Δ-critical instant are introduced.

The definition of priority level-i active period [26] is a continuous period of time $[t_1, t_2)$ during which tasks, of priority i or higher, are executing.

The definition of Δ-critical instant for τ_i is that τ_i is released simultaneously with all higher priority tasks, and a lower priority task τ_k with the biggest blocking time is released a bit earlier to enter its final non-preemptive

non-preemptive.

region. The discrete time granularity Δ is 1 time unit in the paper.

Bril et al [16] showed that the worst-case response time occurs within the priority level-i active period starting at a Δ-critical instant. To calculate the period, the below equation can be used.

$$A_i = B_i + \sum_{\forall j \in hep(i)} \left\lceil \frac{A_i}{T_j} \right\rceil C_j \tag{5.1}$$

The result of A_i is the length of priority level-i active period. B_i is the biggest blocking time from a lower priority task. The blocking time can be calculated by the below equation.

$$B_i = \max_{\forall l \in lp(i)} (F_l - 1) \tag{5.2}$$

B_i is calculated by picking up the value of the final non-preemptive region and minus one. In this thesis, we do not consider the shared resources so the part of shared resource for this equation is eliminated.

In the priority level-i active period, there can be more than one job for a task τ_i so we need to calculate the number of jobs G_i. The calculation is given by:

$$G_i = \left\lceil \frac{A_i}{T_i} \right\rceil \tag{5.3}$$

Due to more than one job for a task, the calculations of response time of each job for a task are needed. The calculation is given by:

$$W_{i,g}^{NP} = B_i + (g+1)C_i - F_i + \sum_{\forall j \in hp_{(i)}} \left(\left\lfloor \frac{W_{i,g}}{T_j} \right\rfloor + 1 \right) C_j \tag{5.4}$$

The result of $W_{i,g}$ is the start time of the final non-preemptive region of job g (where $g = 0$ is the first job) for τ_i. Here, we can use the technique of iteration to solve this equation. The worst-case response time of task τ_i is given by:

$$R_i = max_{\forall g=0,1,2...G_i}(W_{i,g}^{NP} + F_i - gT_i) \tag{5.5}$$

The result of R_i is the worst-case response time of τ_i, calculated by picking the maximum value from a set of results retrieved by Equation (5.4). Task τ_i is schedulable if $R_i \leq D_i$.

To illustrate, there is an example from Davis and Bertogna's paper [26]. Table 5.5 shows a 3-tasks task-set and deadline is less than period. The priority level-3 active period for this task-set is 700 using Equation (5.1).

Task	T	D	C
τ_1	250	175	100
τ_2	400	300	100
τ_3	350	325	100

Table 5.5: A 3-tasks task-set with constrained deadlines.

First, Figure 5.3 shows what happens if the tasks are fully non-preemptive and the priority order is (τ_1, τ_2, τ_3). The top task τ_1 with the highest priority executes first but the second job is blocked by τ_3. Task τ_3 misses its deadline for the second job.

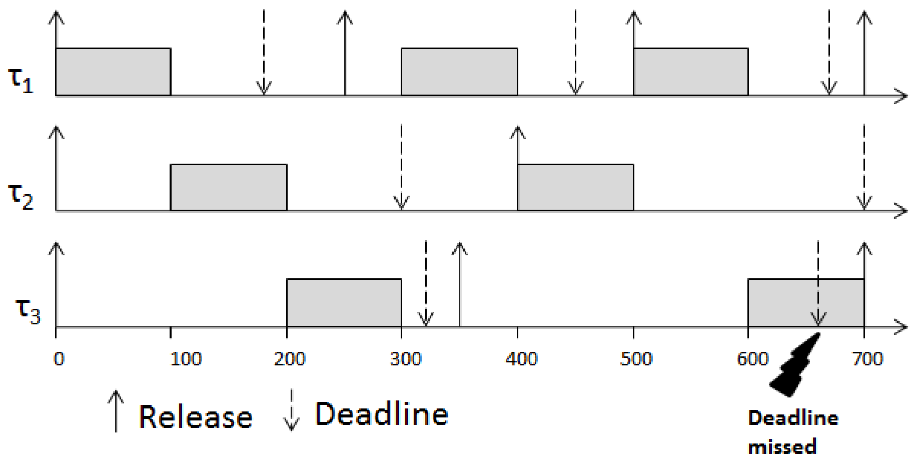

Figure 5.3: It is not schedulable under fully non-preemptive.

Now, Figure 5.4 shows what happens if the priority order changes to (τ_1, τ_3, τ_2) and final non-preemptive region lengths are $F_1 = 1$, $F_3 = 1$ and $F_2 = 51$. All tasks are released at 0 and τ_1 executes first. τ_2 starts to execute at 200, and it enters the final non-preemptive region at 249. Although τ_1 releases at 250, τ_2 continues its job. This task-set is schedulable with this setting.

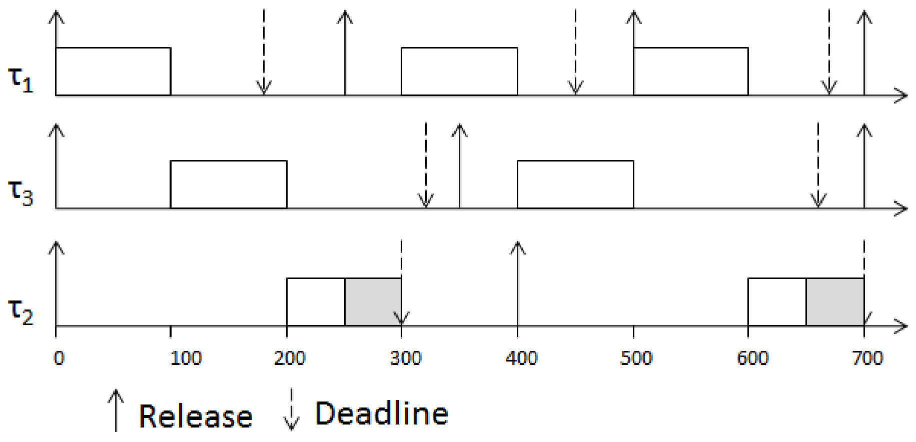

Figure 5.4: It is now schedulable with final non-preemptive regions.

This example shows the DP model can schedule a task-set which cannot be scheduled under the NP model. Figure 5.2 shows as the DP model can be fully non-preemptive or fully preemptive for tasks then the DP model dominates both the preemptive model and the NP model. Unfortunately, the DP model combines the technique of non-preemptive and preemptive, and the technique of preemptive does not support atomic execution. Therefore the DP model cannot apply directly to P-FRP. The following section will adapt the DP model for the technique of AR. We call it the *Deferred Abort* (DA) model in this thesis.

5.3 Analysis for the DA Model

The DA model refers to the DP model but combines abort regions and non-preemptive regions in tasks. The allocation of regions for a task must be abort regions first and then non-preemptive regions.The motivation of the DA model is to eliminate unnecessary aborts, reduce the abort costs and deal with the trade-off of blocking time. The definition of unnecessary aborts is that if a higher priority task does not abort a lower priority task, the system can still be schedulable. To reduce the abort cost, a smaller or zero abort region can be assigned to a task. Sometimes a higher priority task cannot suffer the blocking time from a lower priority task so there is a trade-off to decide the size of non-preemptive regions; reducing the size of non-preemptive regions means increasing the size of abort regions.

To illustrate, Figure 5.5 shows that there are three types of DA tasks. The white boxes represent AR regions and the grey boxes represent non-preemptive regions. τ_1 with a fully white box is fully AR and F points at the end. τ_2 with a fully grey box is fully non-preemptive and F is the entire box. τ_3 with a half white and half grey box is a DA task and F is a half of the box.

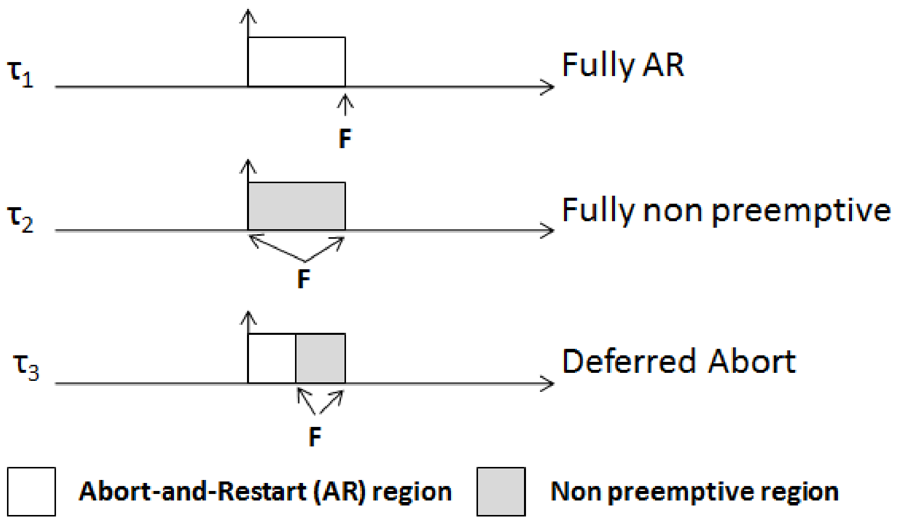

Figure 5.5: Types of deferred abort tasks.

The benefit of using the $\tilde{C}^i_j$ approach (as described in Chapter 3) for the AR model is that the response time analysis can directly use the new execution time, regardless of the abort costs. The response time of the first job for a task is required to check if the task is schedulable or not. Intuitively, the preemptive model and the AR model using the $\tilde{C}^i_j$ approach are equivalent in term of a schedulability analysis, although the test for the AR model is sufficient. By this intuition, the implementation of the DP model can be employed for the DA model by adapting the $\tilde{C}^i_j$ approach.

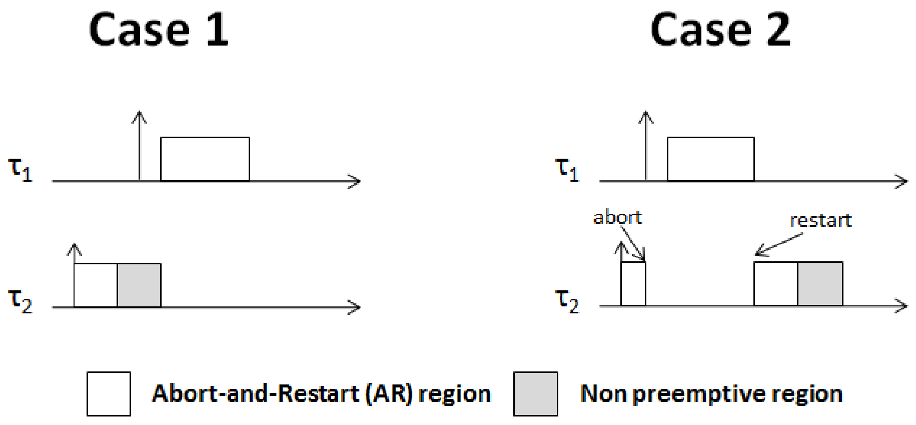

Figure 5.6: Two cases of DA tasks.

Figure 5.6 shows there are two cases for DA tasks. The first case is that the lower priority task τ_2 executes first and the higher priority task τ_1 is released within the final non-preemptive region of τ_2. τ_1 needs to wait until τ_2 completes. This case explains that when a lower priority task is about to complete, it enters the non-preemptive regions to avoid any interference. The second case is that a lower priority task τ_2 executes first and a higher priority task τ_1 is released soon. As τ_2 has not executed much, it has not entered the non-preemptive region yet so it is aborted by τ_1. After τ_1 is completed, τ_2 restarts.

To adapt the $\tilde{C}^i_j$ approach for the DA model, the equation is given by:

$$\tilde{C}^{DA}_{i,j} = C_j + \max_{\forall k \in hep_i \bigcap lp_j} (C_k - F_k) \tag{5.6}$$

The symbol is rephrased by $\tilde{C}^{DA}_{i,j}$. The symbol of DA indicates the value is for the DA model. The $\tilde{C}$ of analysing τ_i for a task τ_j is the sum of the original execution time of τ_j and the maximum value of a task τ_k between priority level i and j, which is after the execution time of τ_k minus its non-preemptive region.

To introduce the equations for the DA model, the structure of the explanation for the DP model above is employed. Firstly, the priority level-i active period is discussed and then the blocking time. Afterwards, the number of jobs required to check within the active period. Lastly, the equations for the response time analysis are derived.

Referring to the priority level-i active period in Equation (5.1), $\tilde{C}^i_j$ can be replaced by $\tilde{C}^{DA}_{i,j}$ directly. The equation for the DA model is given by:

$$A_i = B_i + \sum_{\forall j \in hep(i)} \left\lceil \frac{A_i}{T_j} \right\rceil \tilde{C}^{DA}_{i,j} \tag{5.7}$$

For the calculations of blocking time B_i and the number of jobs G_i , Equation (5.2) and (5.3) can still be used as the execution time C is not involved.

$$B_i = \max_{\forall l \in lp(i)} (F_l - 1) \tag{5.8}$$

$$G_i = \left\lceil \frac{A_i}{T_i} \right\rceil \tag{5.9}$$

Now, the equation for the response time is given by:

$$W_{i,g} = B_i + (g+1)C_i - F_i + \sum_{\forall j \in hp_{(i)}} \left(\left\lfloor \frac{W_{i,g}}{T_j} \right\rfloor + 1 \right) \tilde{C}^{DA}_{i,j} \tag{5.10}$$

The above equation is similar to Equation (5.4). $\tilde{C}^{DA}_{i,j}$ is used and the result of $W_{i,g}$ is the start time of the final non-preemptive region of g^{th} job for τ_i. The number of jobs is required to check that it is based on the value of the priority level-i active period. After a set of response times for a task within the priority level-i active period is calculated, Equation (5.5) still works without any change for finding out the worst-case response time.

$$R_i = max_{\forall g=0,1,2\ldots G_i}(W^{NP}_{i,g} + F_i - gT_i) \tag{5.11}$$

To illustrate the test, there is a 3-tasks task-set which will be scheduled by the models of NP, AR and DA. Deadline is less than period and the priority ordering is τ_1, τ_2 and τ_3.

Task	T	D	C
τ_1	300	80	5
τ_2	400	90	10
τ_3	500	110	80

Table 5.6: This is a 3-tasks task-set.

For the NP model, τ_1 cannot suffer any blocking time or interference which is bigger than 75 as its deadline is 80 and the execution time is 5. As the execution time of τ_3 is 80, τ_1 will miss its deadline wherever τ_3 is placed. Immediately, this task-set is deemed to be unschedulable under the NP model.

For the AR model, τ_1 cannot assign to the lowest priority because it cannot suffer the interference from τ_3. τ_2 cannot assign to the lowest priority too because it cannot suffer the interferences from τ_1 and τ_3. After applying the $\tilde{C}_j^i$ approach, τ_3 will suffer a big interference which is obviously bigger than its deadline. The task-set is apparently not schedulable under the AR model.

Lastly, the DA model is applied with τ_1 and τ_2 fully non-preemptive. For τ_3, the AR region is 5 and the non-preemptive region is 75. The priority ordering is the same; τ_1 has the highest priority, τ_2 is at the middle and τ_3 has the lowest priority. The response time of τ_1 is 80 as it has no interference from higher priority tasks and blocking time is 75 from τ_3. τ_1 is schedulable at the highest priority.

For τ_2, Table 5.7 shows the values of $\tilde{C}_{2,j}^{DA}$ where τ_1 is 5 and τ_2 is 10. The response time of τ_2 is 90 as it suffers 5 time unit interference from τ_1 and 75 blocking time from τ_3 so τ_2 meets its deadline.

Task	T	D	C	AR	F	$\tilde{C}_{2,j}^{DA}$
τ_1	300	80	5	0	5	5
τ_2	400	90	10	0	10	10
τ_3	500	110	80	4	76	-

Table 5.7: The values of $\tilde{C}_{2,j}^{DA}$.

For τ_3, it suffers no blocking time as it is the lowest priority task. By using the values of $\tilde{C}^{DA}_{3,j}$, the interferences for τ_3 are 9 from τ_1 and 14 from τ_2 so the response time is 103 (see calculation below). The DA model can schedule this task-set.

Task	T	D	C	AR	F	$\tilde{C}^{DA}_{3,j}$
τ_1	300	80	5	0	5	9
τ_2	400	90	10	0	10	14
τ_3	500	110	80	4	76	80

Table 5.8: The values of $\tilde{C}^{DA}_{3,j}$.

This example shows the DA model can schedule a task-set but the NP model and the AR model cannot. As tasks can be fully AR or fully non-preemptive, the DA model dominates both NP and AR models.

To apply the equations for the task-set, the calculations are given by:

1. $B_1 = \max_{\forall l \in lp(1)} (F_l - 1)$
2. $B_1 = 75$
3. $A_1 = B_1 + \sum_{\forall j \in hep(1)} \left\lceil \frac{A_1}{T_j} \right\rceil \tilde{C}^{DA}_{1,j}$
4. $A_1^0 = 5$
5. $A_1^1 = 75 + \left\lceil \frac{5}{300} \right\rceil 5 = 80$
6. $A_1^2 = 75 + \left\lceil \frac{80}{300} \right\rceil 5 = 80$
7. $G_1 = \left\lceil \frac{A_1}{T_1} \right\rceil$
8. $G_1 = \left\lceil \frac{80}{300} \right\rceil = 1$
9. $W_{1,g} = B_1 + (g+1)C_1 - F_1 + \sum_{\forall j \in hp(1)} \left(\left\lfloor \frac{W_{1,g}}{T_j} \right\rfloor + 1 \right) \tilde{C}^{DA}_{1,j}$
10. $W_{1,0}^0 = 5$
11. $W_{1,0}^1 = 75 + (0+1)5 - 5 + 0 = 75$

12. $R_1 = max_{\forall g=0,1,2...G_1}(W_{1,g} + F_1 - gT_1)$

13. $R_1 = max_{\forall g=0}(75 + 5 - (0 \cdot 300)) = 80$

The above calculation is for the response time of τ_1. Firstly, B_1 is 75 and the active period is 80 so the number of jobs is 1. The start time of the final non-preemptive region is 75 and finally the worst-cast response time is 80.

1. $B_2 = \max\limits_{\forall l \in lp(2)}(F_l - 1)$
2. $B_2 = 75$
3. $A_2 = B_2 + \sum\limits_{\forall j \in hep(2)} \left\lceil \frac{A_2}{T_j} \right\rceil \tilde{C}^{DA}_{2,j}$
4. $A^0_2 = 10$
5. $A^1_2 = 75 + \left\lceil \frac{10}{300} \right\rceil 5 + \left\lceil \frac{10}{400} \right\rceil 10 = 90$
6. $A^2_2 = 75 + \left\lceil \frac{90}{300} \right\rceil 5 + \left\lceil \frac{90}{400} \right\rceil 10 = 90$
7. $G_2 = \left\lceil \frac{A_2}{T_2} \right\rceil$
8. $G_2 = \left\lceil \frac{90}{400} \right\rceil = 1$
9. $W_{2,g} = B_2 + (g+1)C_2 - F_2 + \sum\limits_{\forall j \in hp(2)} \left(\left\lfloor \frac{W_{2,g}}{T_j} \right\rfloor + 1 \right) \tilde{C}^{DA}_{2,j}$
10. $W^0_{2,0} = 10$
11. $W^1_{2,0} = 75 + (0+1)10 - 10 + \left(\left\lfloor \frac{10}{300} \right\rfloor + 1 \right) 5 = 80$
12. $W^2_{2,0} = 75 + (0+1)10 - 10 + \left(\left\lfloor \frac{80}{300} \right\rfloor + 1 \right) 5 = 80$
13. $R_2 = max_{\forall g=0,1,2...G_2}(W_{2,g} + F_1 - gT_2)$
14. $R_2 = max_{\forall g=0}(80 + 10 - (0 \cdot 400)) = 90$

Now, the calculation for τ_2 is that B_2 is 75 and the active period is 90 so the number of jobs is 1. The start time of the final non-preemptive region is 80 and then the worst-case response time is 90.

1. $B_3 = \max\limits_{\forall l \in lp(3)} (F_l - 1)$
2. $B_3 = 0$
3. $A_3 = B_3 + \sum\limits_{\forall j \in hep(3)} \left\lceil \frac{A_3}{T_j} \right\rceil \tilde{C}^{DA}_{3,j}$
4. $A^0_3 = 80$
5. $A^1_3 = 0 + \left\lceil \frac{80}{300} \right\rceil 5 + \left\lceil \frac{80}{400} \right\rceil 10 + \left\lceil \frac{80}{500} \right\rceil 80 = 95$
6. $A^1_3 = 0 + \left\lceil \frac{95}{300} \right\rceil 5 + \left\lceil \frac{95}{400} \right\rceil 10 + \left\lceil \frac{95}{500} \right\rceil 80 = 95$
7. $G_3 = \left\lceil \frac{A_3}{T_3} \right\rceil$
8. $G_3 = \left\lceil \frac{95}{500} \right\rceil = 1$
9. $W_{3,g} = B_3 + (g+1)C_3 - F_3 + \sum\limits_{\forall j \in hp(3)} \left(\left\lfloor \frac{W_{3,g}}{T_j} \right\rfloor + 1 \right) \tilde{C}^{DA}_{3,j}$
10. $W^0_{3,0} = 80$
11. $W^1_{3,0} = 0 + (0+1)80 - 76 + \left(\left\lfloor \frac{80}{300} \right\rfloor + 1 \right) 9 + \left(\left\lfloor \frac{80}{400} \right\rfloor + 1 \right) 14 = 27$
12. $W^2_{3,0} = 0 + (0+1)80 - 76 + \left(\left\lfloor \frac{27}{300} \right\rfloor + 1 \right) 9 + \left(\left\lfloor \frac{27}{400} \right\rfloor + 1 \right) 14 = 27$
13. $R_3 = max_{\forall g=0,1,2...G_3}(W_{3,g} + F_3 - gT_3)$
14. $R_3 = max_{\forall g=0}(27 + 76 - (0 \cdot 500)) = 103$

Lastly, B_3 is 0 and the active period is 95 then the number of job is 1. The start time of the non-preemptive region is 80 and the worst-case response time is 103. From the calculations using Equations (5.6), (5.7) and (5.10), it mathematically shows the task-set can be scheduled by the DA model using the above approach.

5.3.1 AR or Non-preemptive Region Assignment

This subsection discusses the assignment of the final non-preemptive region for the DA model. Previously, the equations for the response time analysis were introduced but the values of the final non-preemptive region were predefined. Now, we consider how to find out the best values of F for each task. Intuitively, tasks are initially assigned with fully non-preemptive as the NP model has high schedulability mostly. Once a task is not schedulable with fully non-preemptive, reducing the non-preemptive region is required.

Regardless of the priority assignment, a binary search algorithm is a simple method to deal with the problem of the final non-preemptive region assignment. To illustrate, Figures 5.7 and 5.8 show how the binary search algorithm applies to a DA task.

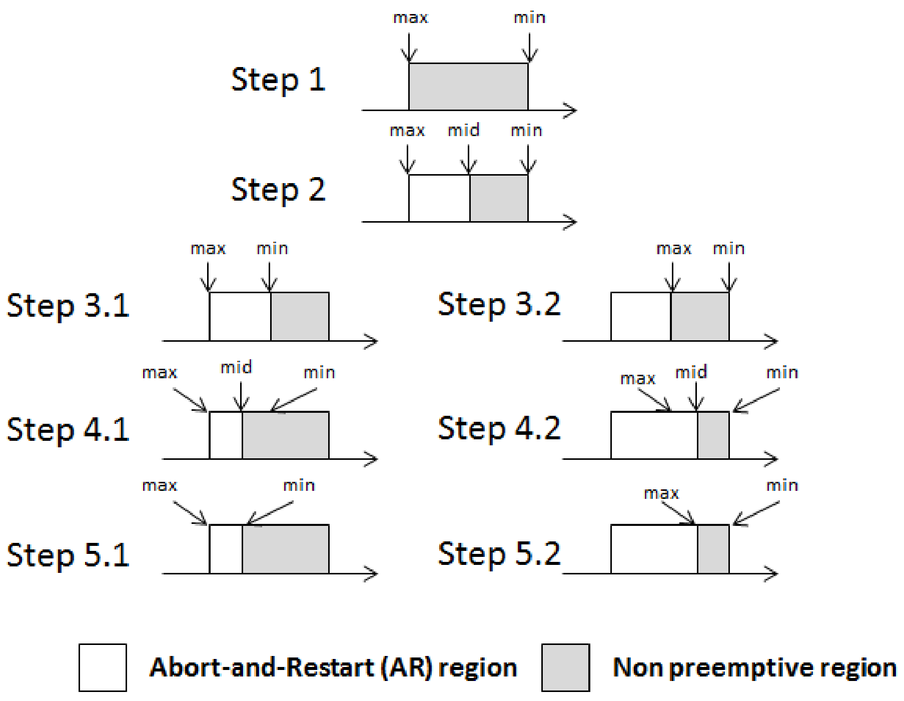

Figure 5.7: Procedures of using binary search for a DA task.

Step 1 When a task-set is not schedulable with fully non-preemptive, max is equal to C and min is equal to 1.

Step 2 Mid is equal to $(max + min)/2$. If the task is schedulable with $F = mid$ then go to Step 3.1 otherwise go to Step 3.2.

Step 3.1 update the value $min = mid$.

Step 3.2 update the value $max = mid$.

Step 4.1 Again, mid is equal to $(max + min)/2$. The task is schedulable with $F = mid$ then go to Step 5.1.

Step 4.2 Again, mid is equal to $(max+min)/2$. The task is not schedulable with $F = mid$ then go to Step 5.2.

Step 5.1 update the value $min = mid$.

Step 5.2 update the value $max = mid$.

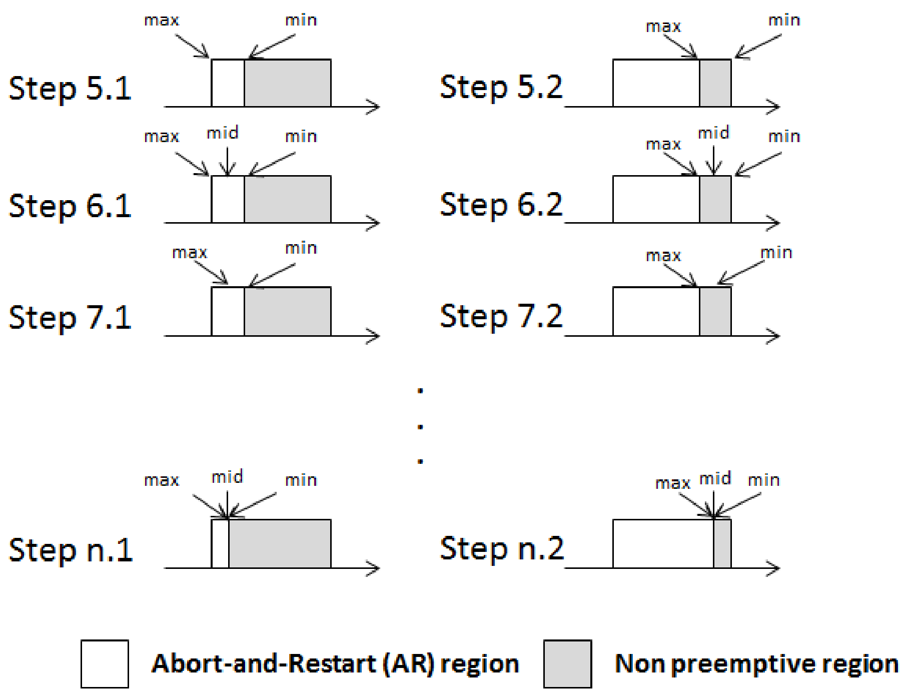

Figure 5.8: Procedures of using binary search for a DA task. (continued)

Step 6.1 Again, mid is equal to $(max+min)/2$. The task is not schedulable with $F = mid$ then go to Step 7.1.

Step 6.2 Again, mid is equal to $(max + min)/2$. The task is schedulable with $F = mid$ then go to Step 7.2.

Step 7.1 update the value $max = mid$ but this time max moves to mid.

Step 7.2 update the value $min = mid$ but this time min moves to mid.

Step n.1 Finally, the final non-preemptive region is found when mid is pointed at max or min.

Step n.2 Finally, the final non-preemptive region is found when mid is pointed at max or min.

The motivation of using binary search for the DA model is to find out the length of the final non-preemptive region as maximum as possible. Now, the assignment for the final non-preemptive region is introduced and then priority assignment schemes will be discussed in the following subsection.

5.3.2 Priority Assignment Schemes

This subsection introduces priority assignment schemes for the DA model, which is complicated because DA tasks can be fully AR, fully non-preemptive or combined with both AR and NP. For fully AR tasks, the EUM priority assignment can be used. For fully non-preemptive tasks, Audsley's algorithm can be used [26]. In the previous chapter, the EUM priority assignment, as close to optimal, was introduced. To apply the EUM priority assignment, the original $\tilde{C}^i_j$ in Equation (3.1) is required to change to $\tilde{C}^{DA}_{i,j}$ in Equation (5.6) as a part of execution time can be non-preemptive. And then it can apply to Algorithm 1. Intuitively, the EUM priority assignment does not offer good performance for non-preemptive tasks. Now, we consider a new algorithm to deal with the combination of AR and NP.

For a task-set consisting of AR, NP and DA tasks, we have to consider the blocking times from low priority tasks and the interferences from high priority tasks. Mostly, non-preemptive tasks have better schedulability than AR tasks but an NP task cannot always be used due to long blocking time and its influence on short deadline high priority tasks. For AR tasks using the $\tilde{C}^i_j$ approach, the priority ordering needs to be considered as the ordering of higher priority tasks is changed and that can affect the schedulability of lower priority tasks. Intuitively, the approaches to design a new algorithm should maximise NP regions and avoid backtracking.

There are many possible priority assignment for the DA model but a heuristic algorithm is initially given by the following. Consider a task-set $\Gamma_N = \{\tau_1, \tau_2, ..., \tau_n\}$ and all tasks are set to fully non-preemptive. The assignment starts with the lowest priority, P_n. Each task is considered at P_n, with other tasks having a higher priority. Afterwards, the binary search for the DA model is used to calculate the final non-preemptive region and it is schedulable under the response time analysis using Equation (5.10). A set of values for P_n is calculated and a schedulable task with the bigger AR region is picked. This task is given priority P_n. The next assignment then occurs at $P_n - 1$ considering only those tasks that have not had their priority permanently assigned, and so on. The task-set is deemed to be unschedulable if there is no schedulable task in the set. Backtracking is not needed as the tasks with a bigger AR region are assigned to lower priorities. Although it is not an optimal scheme, it is guaranteed to avoid backtracking. In this thesis, we call this heuristic algorithm the *MAXAR* algorithm.

To illustrate, Figure 5.9 shows five tasks, τ_1, τ_2, τ_3, τ_4 and τ_5, regardless of the parameters of the tasks. The symbol X means the task is not schedulable at that level. The example starts in Figure 5.9a. The assignment starts with the lowest priority and the next column is a set of values for each task. Those values are calculated based on the lowest priority level. τ_3 and τ_4 are not schedulable with the lowest priority. τ_5 with AR $= 10$ is picked and permanently assigned the lowest priority. The size of the final non-preemptive region is irrelevant as a biggest AR region is considered to avoid backtracking. After the lowest priority level is assigned, the example continues to Figure 5.9b; it shows a set of values for the second lowest priority level. This time τ_1 is 1. In Figure 5.9c, τ_2 is picked; in Figure 5.9d: τ_4 is picked. Finally, τ_3 has the highest priority. Hence, the final priority assignment is $\{\tau_3, \tau_4, \tau_2, \tau_1, \tau_5\}$. The comparison of the EUM algorithm and the MAXAR algorithm will be shown in the section of experimental evaluation.

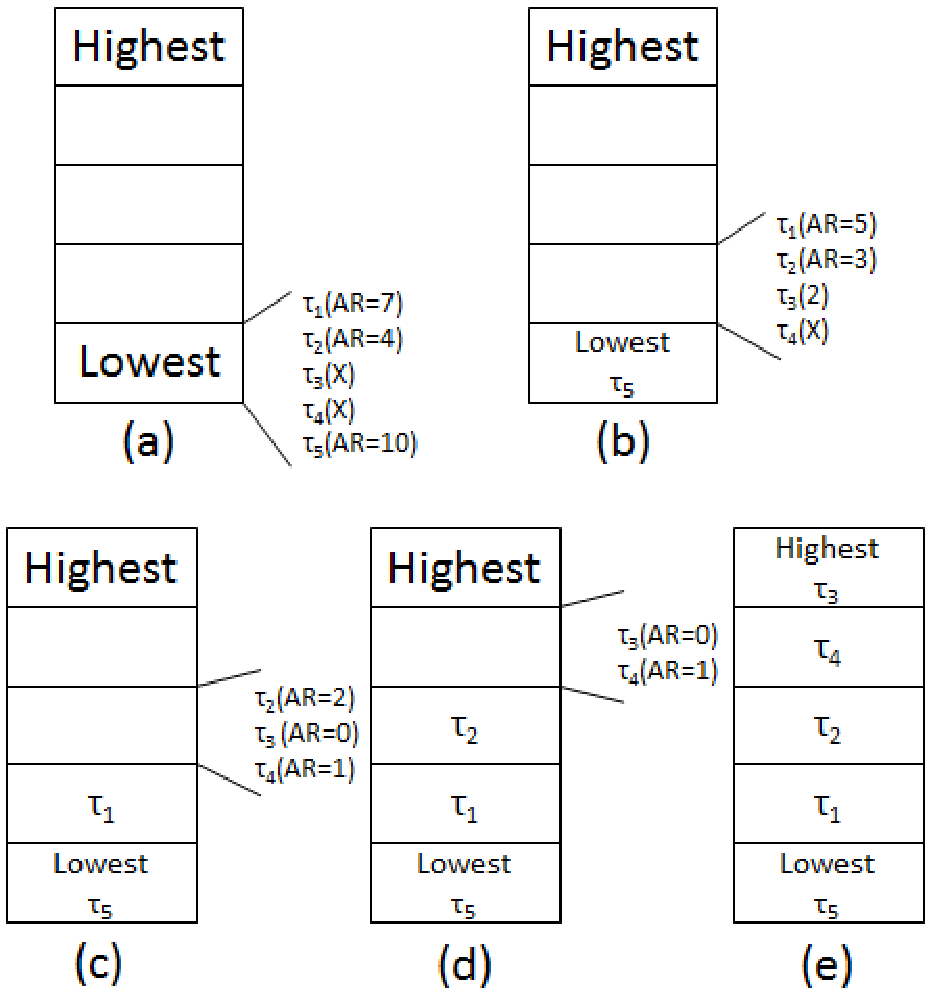

Figure 5.9: The concept of the MAXAR algorithm.

5.4 Experimental Evaluation

This section shows the results from the experiments on the comparison of EUM and MAXAR algorithms under the DA model. DA tasks can be fully AR, fully non-preemptive or DA, so there are four different tests which are 1) fully AR with EUM assignment, 2) fully non-preemptive with EUM assignment, 3) DA with EUM assignment and 4) DA with MAXAR assignment. The schedulability test is using the equations developed in this chapter. It includes the response time analysis for the DA model, the $\tilde{C}_{i,j}^{DA}$ approach and the binary search for the final non-preemptive region assignment. The parameters of the experiments are:

- Deadline is equal to period.
- All tasks are periodic.
- A set of N utilisation values U_i was generated by the UUniFast Algorithm [13].
- Task periods were generated between 500 and 5000 according to a log-uniform distribution[3]. And the computed value T_i is assigned to τ_i.
- Task execution times are: $C_i = U_i \cdot T_i$
- Utilisation for task-sets are ranged between 30% and 60% because some results reach 0% schedulability.
- Each utilisation rate generates 10000 different task-sets, i.e. $U = 30\%$ generates 10000 task-sets, $U = 31\%$ generates another 10000 task-sets, and so on.
- The numbers of tasks are 5, 10 and 15.

For all diagrams, the X-axis is Utilisation rate and the Y-axis is the Schedulability rate, i.e. the percentage of task-sets that were deemed schedulable.

[3]The log-uniform distribution of a variable x is such that ln(x) has a uniform distribution.

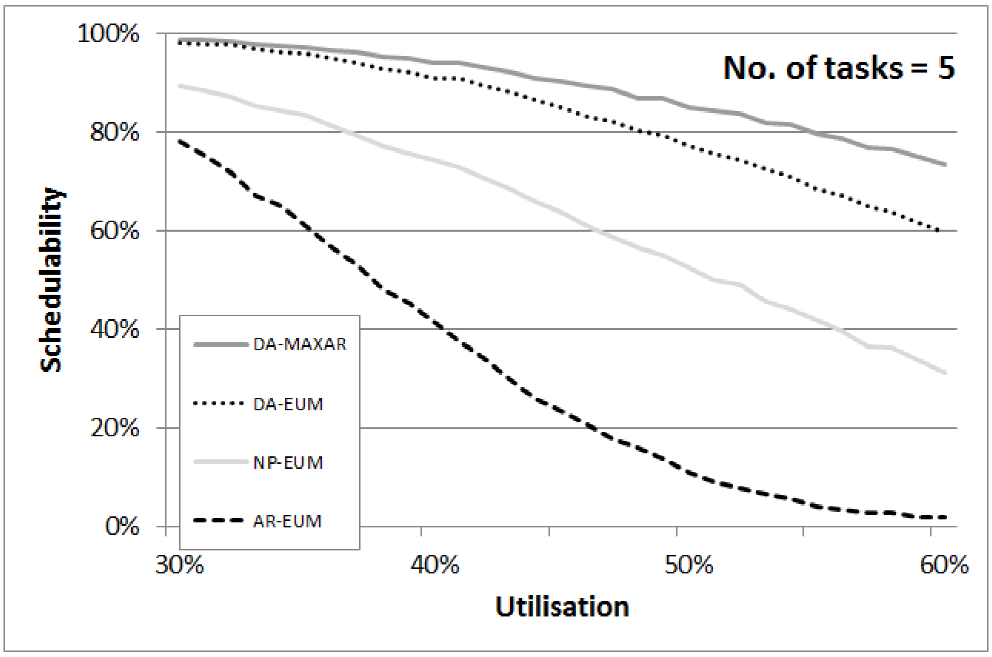

Figure 5.10: 5-tasks task-sets under the DA model.

In Figure 5.10, the number of tasks is 5, which is small, and the schedulability is expected to be higher. The line of AR-EUM is the result of the schedulability rate for fully AR with the EUM algorithm. It is similar to the result from Section 4.4, and it has poor performance compared to others. The line of NP-EUM is the result of fully non-preemptive with the EUM algorithm. Although EUM is not an optimal priority assignment for the NP model, it is still better than AR-EUM. The line of DA-EUM is the result of DA with the EUM algorithm. It is far better than both AR-EUM and NP-EUM. Indeed, DA dominates fully AR and fully non-preemptive. Lastly, the line of DA-MAXAR has the best performance. Although the MAXAR algorithm is a heuristic, the result shows a big improvement at 60% utilisation.

In Figure 5.11, the number of tasks is 10. The line of AR-EUM is dropped as expected, and it reaches 0% schedulability at 50% utilisation. The line of NP-EUM is degraded too but it has not reached 0% schedulability before 60% utilisation. The line of DA-EUM has less schedulability than before. Now the line of DA-MAXAR has better schedulability than the result from 5-tasks

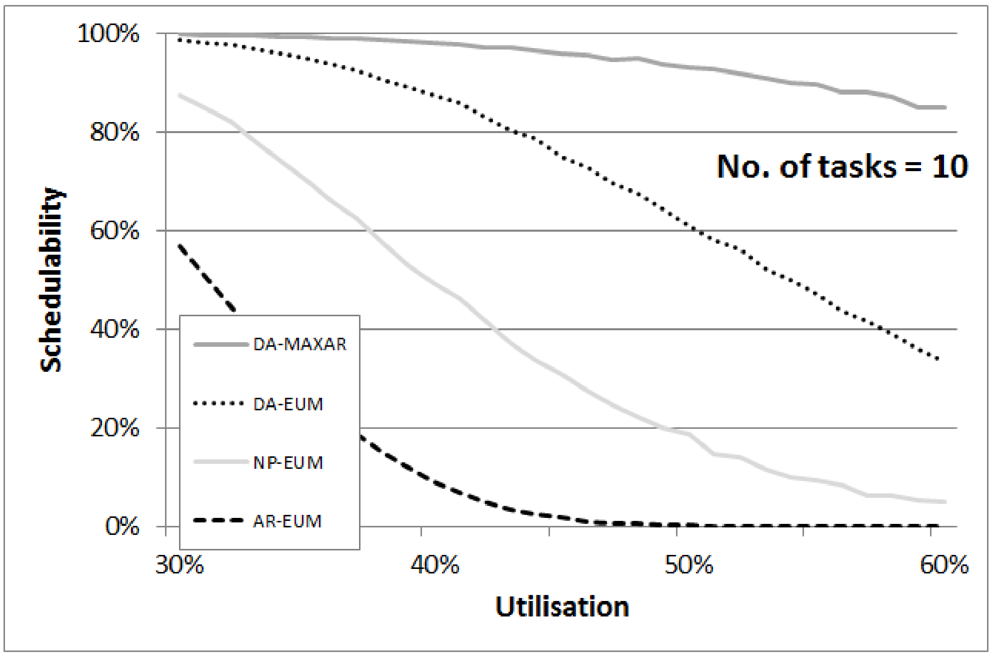

Figure 5.11: 10-tasks task-sets under the DA model.

task-sets. The result is not expected, so further analysis and discussion will follow.

In Figure 5.12, the number of tasks is 15 which is bigger. The line of AR-EUM has very poor performance which also reaches 0% schedulability at 40% utilisation. The line of NP-EUM finally reaches 0% schedulability at 60% utilisation. The line of DA-EUM has dropped a bit than before. The line of DA-MAXAR is again improved.

The results of DA-MAXAR from above are not expected. When the number of tasks is increased, the schedulability also increases. In the meantime, the execution times of tasks will be smaller relatively. The NP model gets the benefit of small execution times because the blocking times will be smaller.

To illustrate, Figure 5.13 shows three cases of different numbers of tasks. The utilisation of the system is set to 60%. When the number of tasks is 3, each task has five boxes for its execution time. When the number of tasks is 5, the execution time of each task is three boxes. When the number of tasks is 15, only one box represent the execution time for each task. By this example, the blocking time is very small in the task-set with N = 15. To

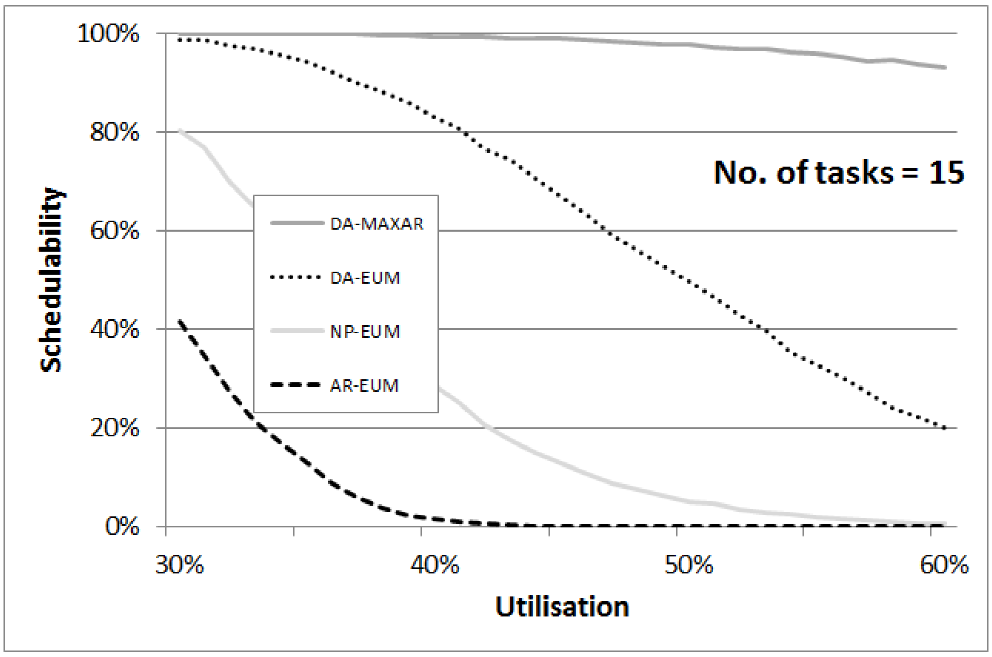

Figure 5.12: 15-tasks task-sets under the DA model.

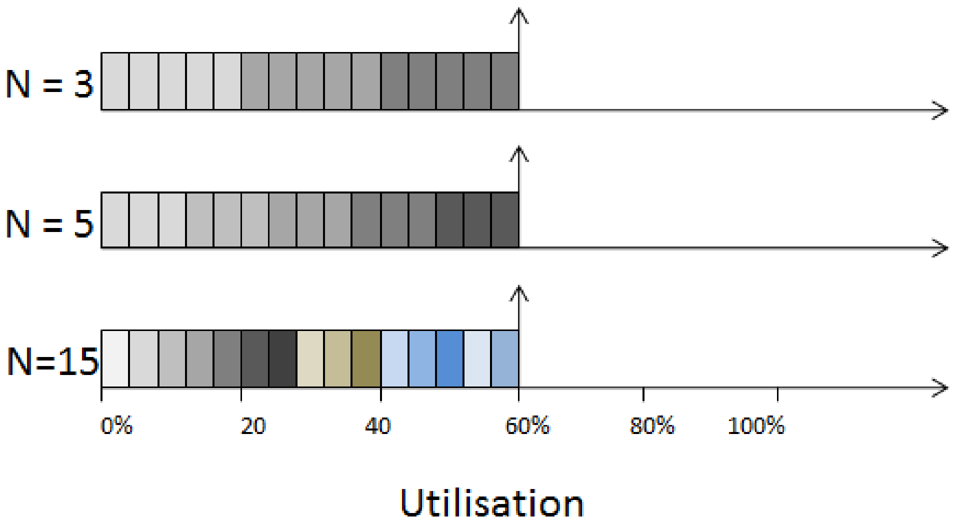

Figure 5.13: The relationship between different numbers of tasks.

show this is correct, a further experiment is done and it is given below.

The experiment below is to show that the NP model has an advantage with a bigger number of tasks. The test of fully non-preemptive with Audsley's algorithmm is added. The schedulability is the same as above.

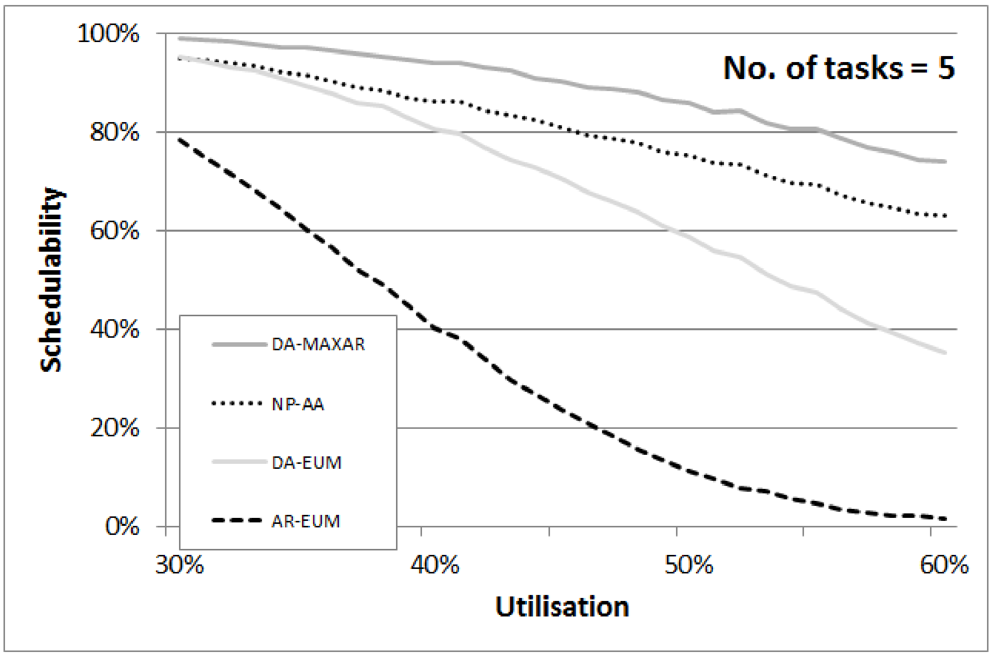

Figure 5.14: The number of tasks is 5.

In Figure 5.14 for 5 tasks, the line of NP-AA is better than the line of DA-EUM because of the optimal priority assignment for fully non-preemptive. The line of DA-MAXAR is still the best.

In Figure 5.15, the number of tasks is 10. Both lines of NP-AA and DA-MAXAR are increased for their schedulability.

Lastly, Figure 5.16 shows the result for the 15-tasks task-set. The lines of NP-AA and DA-MAXAR indeed show get better improvement with a bigger number task.

5.5 Summary

To summarise, this chapter has introduced an alternative scheme, deferred abort, to improve the schedulability of the AR model. The DA model com-

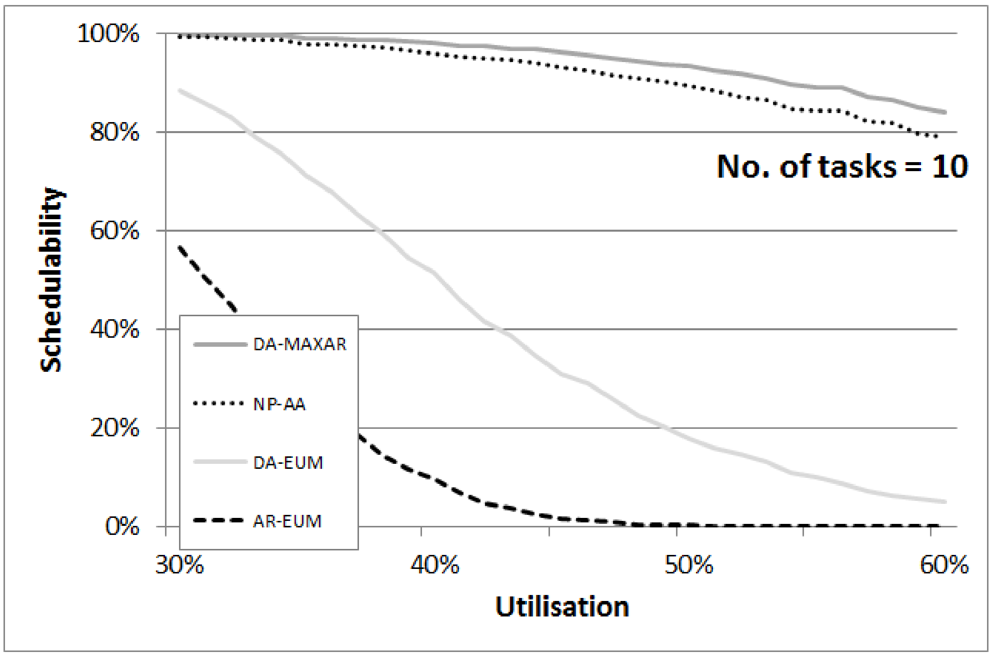

Figure 5.15: The number of tasks is 10.

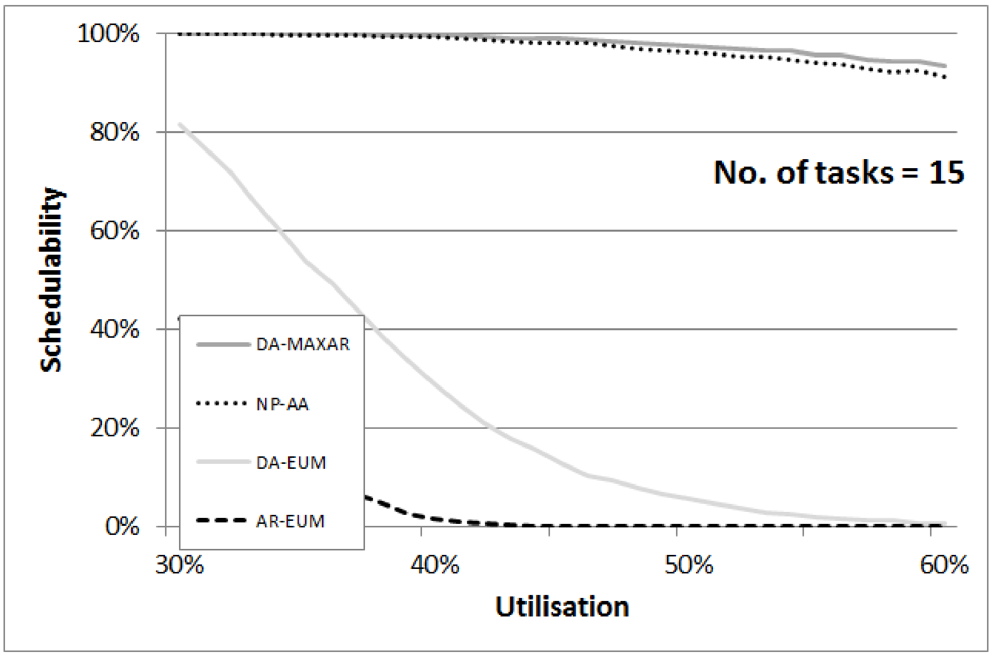

Figure 5.16: The number of tasks is 15.

bines the techniques of non-preemptive and AR. To implement the DA model, new equations are introduced, and an experiment evaluation is provided. During the discussion of the results from the experiments, the advantage of a bigger number task for the NP model is observed. A heuristic is developed for priority assignment. It performs well; however, it is not optimal and it is possible that an improvement to the heuristic is possible.

Chapter 6

A Tighter Analysis for the AR Model

This chapter introduces a tighter analysis of schedulability tests for the AR model. Chapter 3 has introduced a simple approach, $\tilde{C}^i_j$, to deal with the response time analysis for the AR model, and it simplified the calculations on the analysis used with EUM priority assignment. In reality, higher priority tasks does not always abort the lower priority task with the biggest execution time on every release; therefore the higher priority task aborts the lower priority task with the second biggest execution time.

To illustrate, Figure 6.1 shows the case that the second job of τ_1 aborts τ_3 which has a smaller C. The task-set has three tasks, τ_1, τ_2 and τ_3. The top task, τ_1, has the highest priority. The pale grey box means abort cost. The dark grey box means fully executed. The upward arrow means a point of release. The downward arrow means a point of abort. For τ_1, the lower priority task with the biggest execution time is τ_2. For τ_2, the lower priority task with the biggest execution time is τ_3. The first job of τ_1 aborts τ_2. Before the second job of τ_1 releases, τ_2 is done. Therefore, τ_1 cannot abort τ_2 a second time and has to choose τ_3 to abort.

The above example shows that the current response time analysis for the AR model can be less pessimistic if the analysis of choosing an abort task is done on each release. Now we consider an approach to deal with the analysis

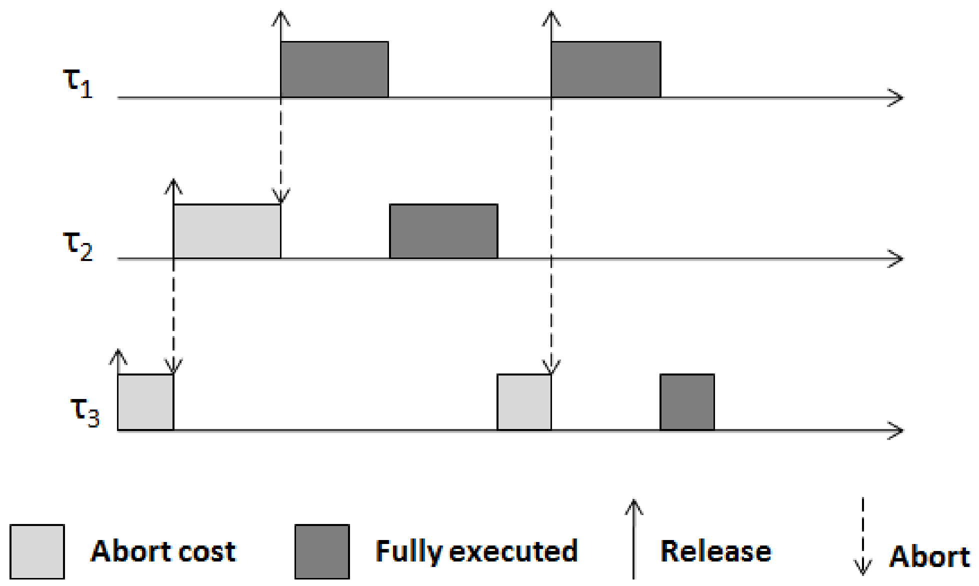

Figure 6.1: The second job of τ_1 aborts τ_3 which has a smaller C.

.

of each release. The following section introduces an analysis on cache related preemption delay which will be adapted in Section 6.2 to the AR model.

6.1 Cache Related Preemption Delay Analysis

This section briefly introduces an analysis on Cache Related Preemption Delay (CRPD), where each preemption follows a cache delay and the cost of the preemption is related to the preempted tasks. Different tasks have different preemption costs. To assume that all preemptions have the worst-case cost is pessimistic. To deal with the analysis of preemption costs, the multi-set approach [2, 3] has been introduced.

The response time with the preemption costs can be calculated [19] by the following equation.

$$R_i = C_i + \sum_{\forall j \in hp_{(i)}} \left(\left\lceil \frac{R_i + J_j}{T_j} \right\rceil C_j + \gamma_{i,j} \right) \tag{6.1}$$

$\gamma_{i,j}$ represents the total preemption cost of τ_j when scheduling τ_i. The preemption cost may be different on each release so the equation cannot simply apply the sum of C_j and the first preemption cost. In this equation, the release jitter J_j for τ_j is considered. The computation of $\gamma_{i,j}$ depends on the approach used. For the AR model, a new approach is required. In the paper written by Altmeyer et al. [3], the authors used a multi-set M to contain all the possible costs. As this approach is used for cache related preemption delay analysis and cannot apply to the AR model, the explanations of the above equations are not given in detail. In the following section, we adapt this approach to the AR model.

6.2 A New Approach for the AR Model

This section introduces how to adapt the multi-set approach to the AR model. When scheduling a task τ_i, a higher priority task τ_j can abort any task $\tau_k \in hep_{(i)} \bigcap lp_{(j)}$ up to $E_j(R_i)$ times, where $E_j(R_i)$ is the number of release of τ_j within the response time of τ_i. Here, a term *multi-bag* is used instead of *multi-set.* A bag $M_{i,j}$ contains a series of abort costs which comes from available tasks $k \in hep_{(i)} \bigcap lp_{(j)}$ within the response time R_i. To calculate the total abort cost, τ_j sums up $E_j(R_i)$ largest values from $M_{i,j}$. A task-set has a number of bags for each task. Therefore, we call this approach the *multi-bag*, in this thesis.

To implement the multi-bag approach, the equations from the section of CRPD analysis are adapted as below.

$$R_i = C_i + \sum_{\forall j \in hp_{(i)}} \left(\left\lceil \frac{R_i}{T_j} \right\rceil C_j + \gamma_{i,j} \right) \tag{6.2}$$

The response time analysis with abort cost for the AR model can be computed by the above equation. $\gamma_{i,j}$ is the abort cost of τ_j. The jitter J_j is removed from this equation as it is not considered in this thesis.

In the $\tilde{C}_j^i$ approach, the interference cost from τ_j is the sum of its original execution time, C_j, and the maximum abort cost of τ_j. The equation is shown below.

$$\left\lceil \frac{R_i}{T_j} \right\rceil \tilde{C}_j^i \Rightarrow \left\lceil \frac{R_i}{T_j} \right\rceil C_j + \left\lceil \frac{R_i}{T_j} \right\rceil \max_{\forall k \in hep_i \bigcap lp_j} C_k \tag{6.3}$$

For the multi-bag approach, all possible abort costs are considered and are needed to be contained in a bag. The equation of the maximum abort cost, to the left, is required to change to the equation of all possible abort costs, to the right.

$$\left\lceil \frac{R_i}{T_j} \right\rceil \max_{\forall k \in hep_i \bigcap lp_j} C_k \Rightarrow \left\lceil \frac{R_i}{T_j} \right\rceil \left(\bigcup_{E_j(R_k)E_k(R_i)} C_k \right) \tag{6.4}$$

Now the equation for finding out possible abort costs is shown below.

$$cost = \left(\bigcup_{E_j(R_k)E_k(R_i)} C_k \right) \tag{6.5}$$

To apply Equation (6.5) to the multi-bag approach, a new equations is given by:

$$M_{i,j} = \bigcup_{k \in hep_{(i)} \bigcap lp_{(j)}} \left(\bigcup_{E_j(R_k)E_k(R_i)} C_k \right) \tag{6.6}$$

In this equation, $M_{i,j}$ is a bag for τ_j when scheduling τ_i. It contains a series of abort costs which comes from available tasks $k \in hep_{(i)} \bigcap lp_{(j)}$ within the response time R_i. τ_j represents aborting task τ_k $E_j(R_k)E_k(R_i)$ times for each task $\tau_k \in hep_{(i)} \bigcap lp_{(j)}$, and those abort costs C_k are added to $M_{i,j}$.

$$\gamma_{i,j} = \sum_{l=1}^{E_j(R_i)} M_{i,j}^l \tag{6.7}$$

$\gamma_{i,j}$ is then given by the $E_j(R_i)$ largest values in $M_{i,j}$, where $M_{i,j}^l$ is the $l-th$ largest value in $M_{i,j}$. To illustrate, a task-set is given in Table 6.1.

The task-set has three tasks, τ_1, τ_2 and τ_3. The priority ordering is $P_1 > P_2 > P_3$. Firstly, the approach of $\tilde{C}_j^i$ and the standard response time analysis in Equation (2.5) are used, but the task-set cannot be schedulable. The values for $\tilde{C}_1^3, \tilde{C}_2^3, \tilde{C}_3^3$ are 13, 13 and 3. The calculation is shown below.

Task	T=D	C
τ_1	25	3
τ_2	35	10
τ_3	45	3

Table 6.1: An example task-set.

1. $R_3^0 = 3$
2. $R_3^1 = 3 + (\lceil \frac{3}{25} \rceil \cdot 13 + \lceil \frac{3}{35} \rceil \cdot 13) = 29$
3. $R_3^2 = 3 + (\lceil \frac{29}{25} \rceil \cdot 13 + \lceil \frac{29}{35} \rceil \cdot 13) = 42$
4. $R_3^3 = 3 + (\lceil \frac{42}{25} \rceil \cdot 13 + \lceil \frac{42}{35} \rceil \cdot 13) = 55$

.

It ends here because the response time is bigger than the deadline.

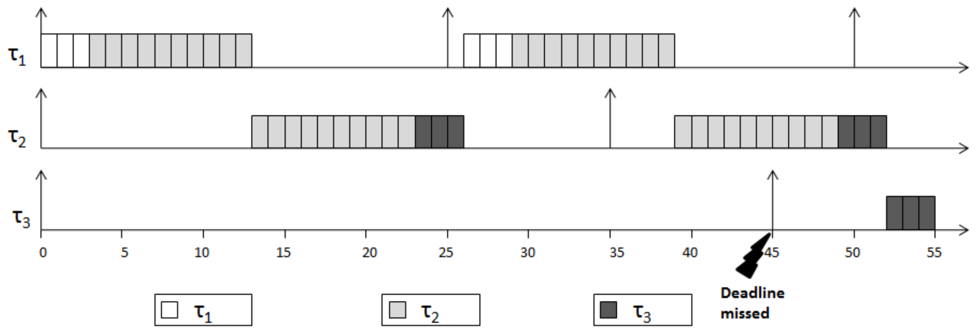

Figure 6.2: The task-set cannot pass with the $\tilde{C}_j^i$ approach.

Figure 6.2 shows the three tasks: the white box is τ_1, the grey box is τ_2 and the black box is τ_3. One box represents 1 time unit. On the time line of τ_1, the execution boxes combine 3 white boxes and 10 grey boxes because $\tilde{C}_1^3$ is 3 + 10. The figure only depicts the analysis of the standard response time. On the time line of τ_2, there are 10 grey boxes and 3 black boxes. On the lowest time line, τ_3 missed its deadline after a few iterations. The task-set is deemed to be not schedulable using the $\tilde{C}_j^i$ approach.

Now the multi-bag approach is used and the procedures are shown in detail. Firstly, τ_1 is analysed and the calculation is given by:

1. $R_1^0 = 3$

Obviously, the response time for the highest priority task is its execution time C_1 but the value $R_1 = 3$ is needed for the next calculation.

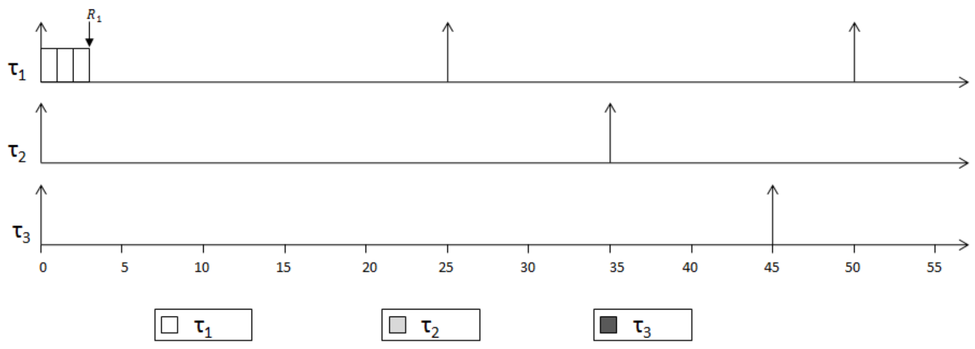

Figure 6.3: The response time analysis for τ_1.

Figure 6.3 shows the response time R_1 of τ_1 is 3. This simple diagram is only a reference. Below, τ_2 is analysed and R_2^0 starts with 10. After two more iterations, the response time is computed then R_2 is 23. The calculation is given by:

1. $R_2^0 = 10$

2. $R_2^1 = 10 + (\lceil \frac{10}{25} \rceil \cdot 3 + \gamma_{2,1}^{l=1}\{10\}) = 23$

3. $R_2^2 = 10 + (\lceil \frac{23}{25} \rceil \cdot 3 + \gamma_{2,1}^{l=1}\{10\}) = 23$

Figure 6.4 shows the analysis of the response time for τ_2. On the time line of τ_1, the execution time combines 3 white boxes and 10 grey boxes because the abort cost for τ_1 is $\gamma_{2,1}^{l=1}\{M_{2,1}^l\}$ and the bag $M_{2,1}$ only contains 10 within the response time $R_2 = 23$.

After the response time R_2 is found, the response time for τ_3 can be computed as below.

1. $R_3^0 = 3$

2. $R_3^1 = 3 + (\lceil \frac{3}{25} \rceil \cdot 3 + \gamma_{3,1}^{l=1}\{10, 3\}) + (\lceil \frac{3}{35} \rceil \cdot 10 + \gamma_{3,2}^{l=1}\{3\}) = 29$

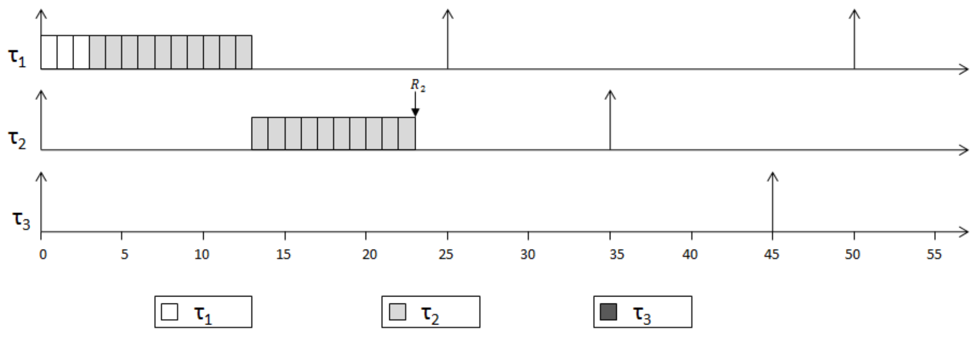

Figure 6.4: The response time analysis for τ_2.

3. $R_3^2 = 3 + (\lceil \frac{29}{25} \rceil \cdot 3 + \gamma_{3,1}^{l=1,2}\{10, 3, 3\}) + (\lceil \frac{29}{35} \rceil \cdot 10 + \gamma_{3,2}^{l=1}\{3\}) = 35$

4. $R_3^3 = 3 + (\lceil \frac{35}{25} \rceil \cdot 3 + \gamma_{3,1}^{l=1,2}\{10, 3, 3\}) + (\lceil \frac{35}{35} \rceil \cdot 10 + \gamma_{3,2}^{l=1}\{3\}) = 35.$

R_3^0 starts with 3 and the first iteration shows the R_3 is 29. R_3 is bigger than T_1 so there is another release for τ_1. On the second iteration, R_3 is now 35 and the calculation stops after the third iteration as there is no another release. The response time for τ_3 is 35 and the task-set is deemed to be schedulable with the multi-bag approach.

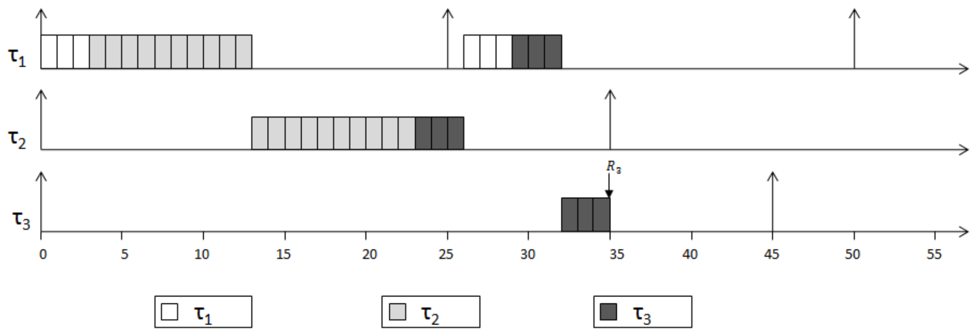

Figure 6.5: The response time analysis for τ_3.

Figure 6.5 shows the analysis of the response time for τ_3. On the time line of τ_1, the execution time of the first job combines 3 white boxes and 10 grey boxes because the abort cost for τ_1 is $\gamma_{3,1}^{l=1}\{M_{3,1}^l\}$ and the bag $M_{3,1}$ contains 10 and 3. On the time line of τ_2, the execution time of the first job combines 10 grey boxes and 3 black boxes because the abort cost for τ_1 is $\gamma_{3,2}^{l=1}\{M_{3,2}^l\}$ and the bag $M_{3,2}$ only contains a value, 3.

On the time line of τ_1, the execution time of the second job combines 3 white boxes and 3 black boxes this time because R_2 is 23 (see Figure 6.4) and T_2 is 35 and that means no task can abort τ_2 between 23 and 35. The abort cost for the second job of τ_1 is 3 as $\gamma_{3,1}^{l=2}\{M_{3,1}^{l}\}$ and the bag $M_{3,1}$ contains 10, 3 and 3. τ_3 just meets the deadline at 35. Please note that the figure only depicts the analysis and it does not mean non-preemptive is used.

As the above example shows, the multi-bag approach can improve the schedulability for the AR model. The following section shows the experiments on comparison of the $\tilde{C}_j^i$ and the multi-bag approaches. The new approach clearly dominates the original, as the original is equal to a bag with maximum values.

6.3 Experimental Evaluation

This section shows the results from the experiments of the comparison of the $\tilde{C}_j^i$ and the multi-bag approaches with DM and EUM priority assignment. To be consistent, the structure of the experiment is similar to the previous experiments. The parameters of the experiments are:

- Deadline is equal to period.
- All tasks are periodic.
- A set of N utilisation values U_i was generated by the UUniFast Algorithm [13].
- Task periods were generated between 5000 and 50000 according to a log-uniform distribution[1]. And the computed value T_i is assigned to τ_i.
- Task execution times are: $C_i = U_i \cdot T_i$
- Utilisation for task-sets are ranged between 10% and 60%.

[1]The log-uniform distribution of a variable x is such that ln(x) has a uniform distribution.

- Each utilisation rate generates 10000 different task-sets, i.e. $U = 30\%$ generates 10000 task-sets, $U = 31\%$ generates another 10000 task-sets, and so on.
- The numbers of tasks are 5, 10, 15 and 20. An additional 8-tasks task-set for an exhaustive search is included at the end.

For all diagrams, the X-axis is Utilisation rate and the Y-axis is the Schedulability rate, i.e. the percentage of task-sets that were deemed schedulable.

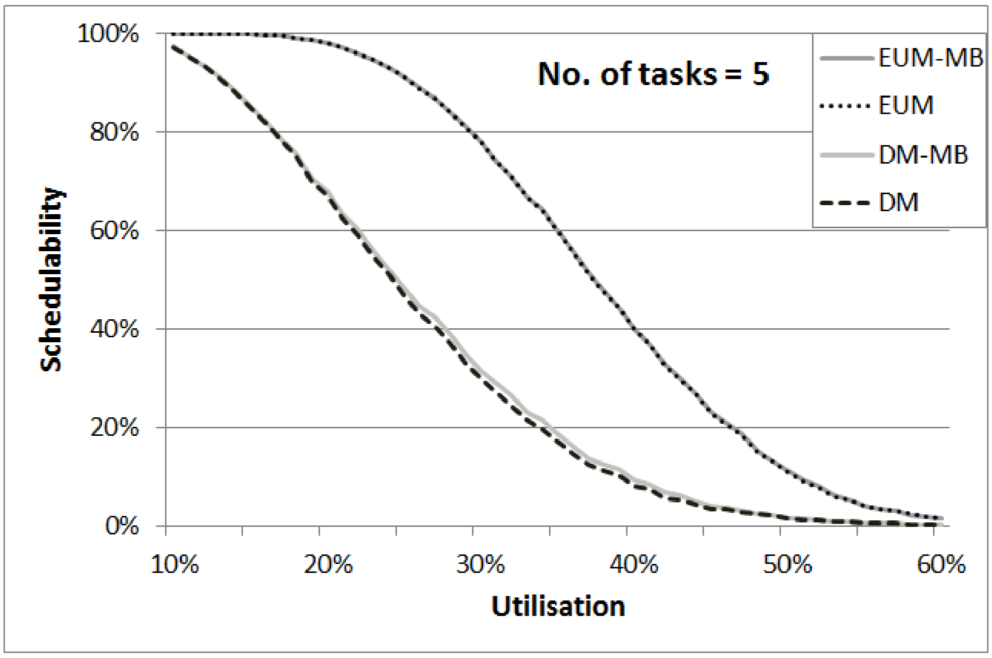

Figure 6.6: Comparison of $\tilde{C}_j^i$ and multi-bag approaches with $n = 5$.

Firstly, Figure 6.6 shows the result of the 5-tasks task-set. The line of DM represents the result of the $\tilde{C}_j^i$ approach with DM ordering; it has the worst performance. The line of DM-MB represents the result of the multi-bag approach with DM ordering; there is an improvement compared to the line of DM. The line of EUM is the result of the $\tilde{C}_j^i$ approach with EUM ordering; it has much better performance than both results of DM and DM-MB. The line of EUM-MB has slight improvement. Although the figure does not show it clearly, the difference can be found from the exact result of the data of this experiment.

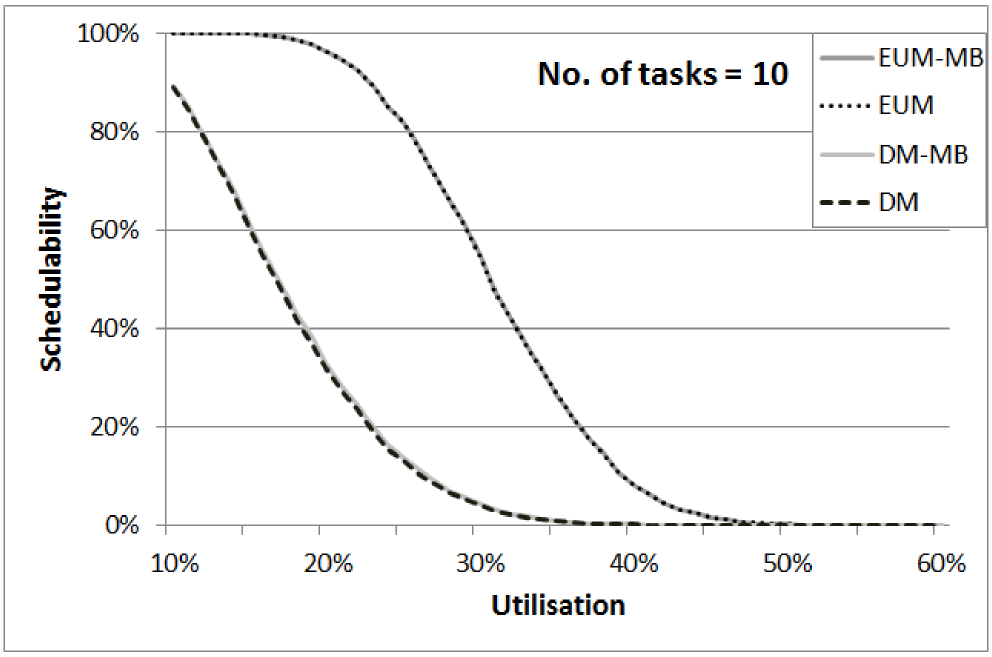

Figure 6.7: Comparison of $\tilde{C}_j^i$ and multi-bag approaches with $n = 10$.

Secondly, Figure 6.7 shows the result of the 10-tasks task-set. As expected, the schedulability of all lines is reduced. The pattern of the result is similar to the previous result. The difference between the lines of DM and DM-EUM is apparently shown on the diagram. The lines of EUM and EUM-MB are still stuck together but the improvement still exists from the exact data.

Figure 6.8 shows the result of 15-tasks task-set. The schedulability is just scaled down from the previous result as the number of tasks is increased. A gap between the lines of DM and DM-EUM still shows on the diagram. The lines of EUM and EUM-MB are still stuck together but the improvement still exists from the exact data.

Lastly, Figure 6.9 shows the result of the 20-tasks task-set. The gap between the lines of DM and DM-EUM becomes narrow. By this trend, the multi-bag approach offers less improvement when the number of tasks increases. At this level of number of tasks, EUM-MB still has improvement

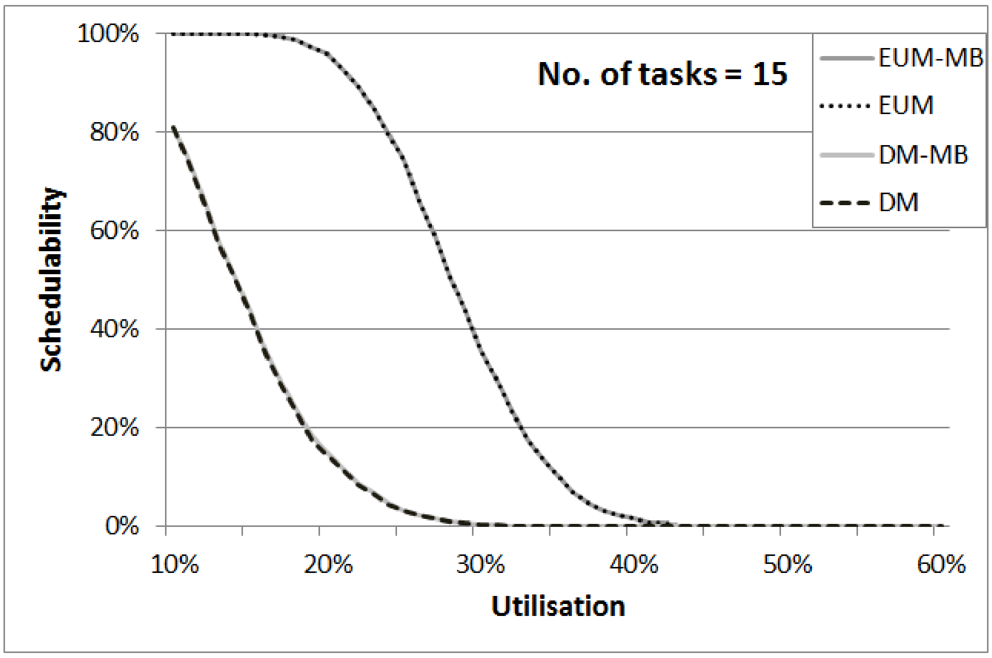

Figure 6.8: Comparison of $\tilde{C}_j^i$ and multi-bag approaches with $n = 15$.

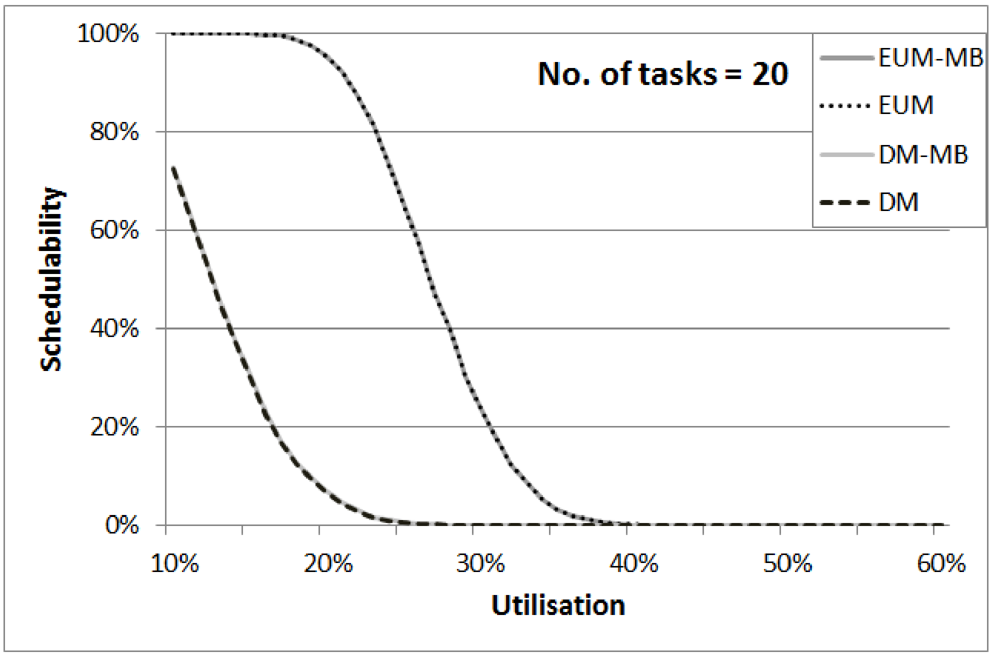

Figure 6.9: Comparison of $\tilde{C}_j^i$ and multi-bag approaches with $n = 20$.

as shown by the exact experiment data.

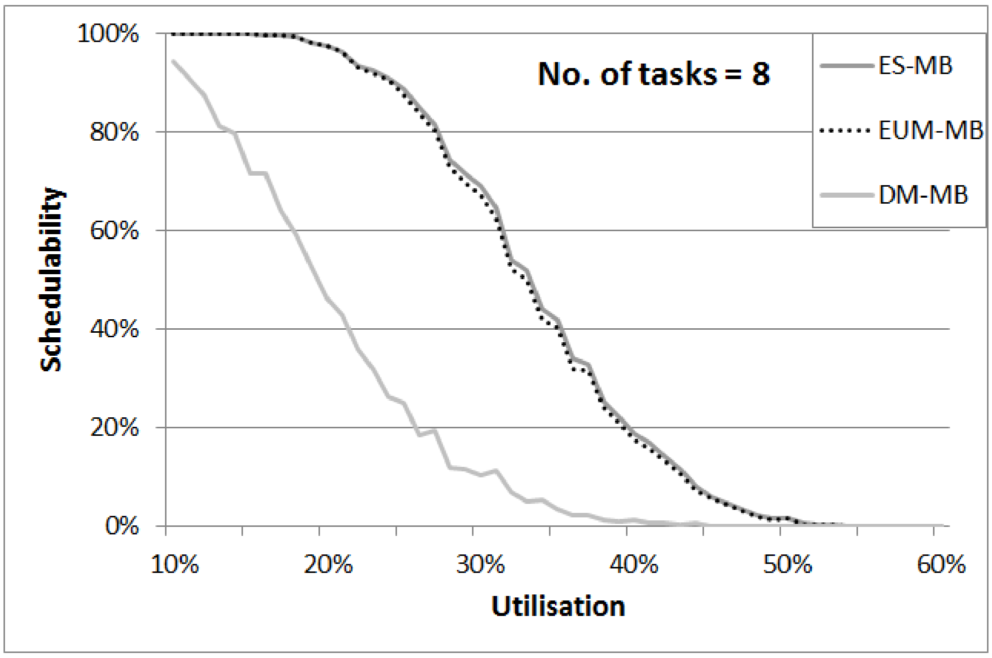

Figure 6.10: Comparison of EUM and ES with the multi-bag approach and $n = 8$.

This is an additional experiment where the number of tasks is 8 and the tests are DM-MB, EUM-MB and ES-MB (exhaustive search, i.e. optimal). This experiment aims at the performance of the multi-bag approach with different priority assignments. As the computation of exhaustive search is huge, the number of tests for each utilisation is reduced to 1000 times. The line of DM-MB has the worst performance, as expected. The line of EUM-MB is close to the line of ES-MB. As the previous chapter shows, EUM is close to optimal; the result from this experiment is further evidence that EUM is close to optimal with even the multi-bag approach.

To evaluate the results, the multi-bag approach offers an obvious improvement if the priority assignment is far from optimal. For better priority assignment, the schedulability is slightly better. The reason is that the goal of the multi-bag approach provides a tighter analysis on the decision of aborts. In other words, a mid-level priority task with the biggest execution time and a bigger period cannot be aborted by higher priority tasks between its completion time and the next release time. This experiment only generated unbiased task-sets, and this did not put emphasis on the improvement of using the multi-bag approach.

As the task-set generator is general, it does not create particular task-sets to show the performance of using the multi-bag approach. We created another set of diagrams using the same data from the above experiment. For the below diagrams, the X-axis is Utilisation rate and the Y-axis is the Average response time.

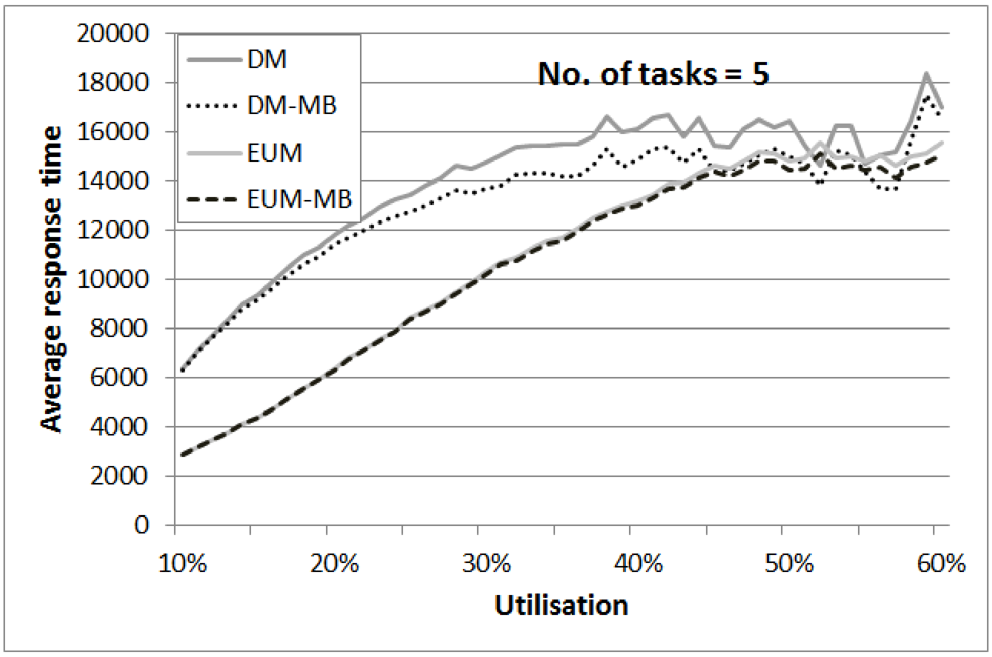

Figure 6.11: Compare average response time of 5-tasks task-sets.

Figure 6.11 with 5-tasks task-sets shows the results that EUM and EUM-MB have shorter response times. A short response time means a higher possibility of meeting the deadline. The difference between DM and DM-MB is obvious as DM priority assignment is worst for the AR model in most cases. The multi-bag approach shows its advantage clearly. The lines DM and DM-MB become jaggy after 40% utilisation because the number of schedulable task-sets is reduced rapidly for the DM priority assignment.

Figure 6.12 with 10-tasks task-sets shows a similar pattern to the results of the 5-tasks task-sets, but this time all lines become jaggy at the end. The

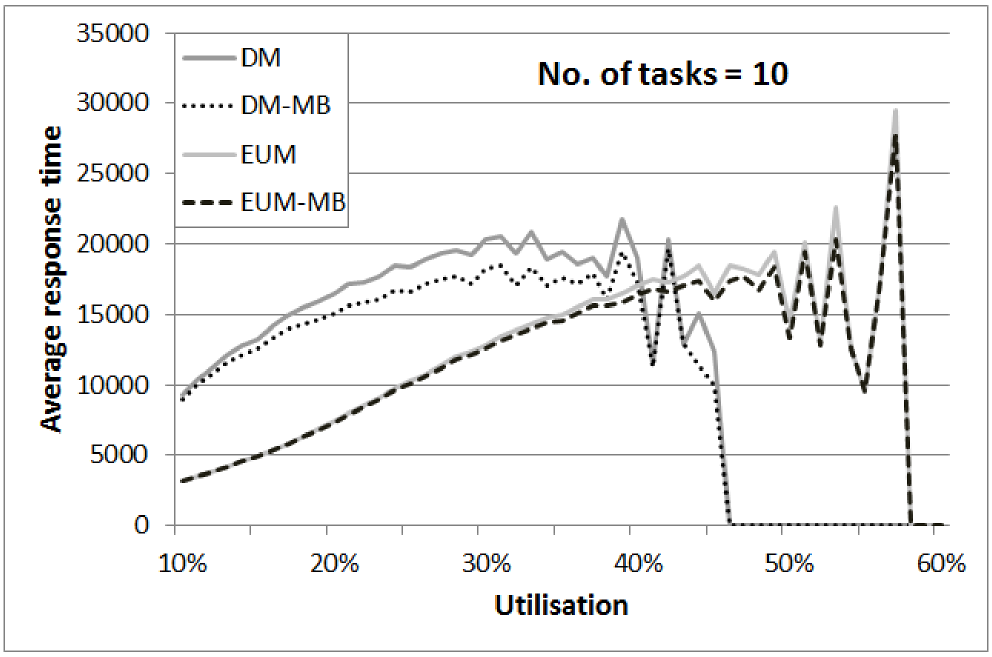

Figure 6.12: Compare average response time of 10-tasks task-sets.

results of DM and DM-MB reach fully unschedulable at 46%, and the results of EUM and EUM-MB at 58%.

Figure 6.13 with 15-tasks task-sets has a similar pattern to the results of the 10-tasks task-sets. Now the difference between with and without the multi-bag approach becomes clear.

Lastly, Figure 6.14 with 20-tasks task-sets is shown as expected. The result is scaled down from the result of the 15-tasks task-sets. The figures have a better presentation than before although the same data is used.

6.4 Summary

This chapter introduced a tighter analysis on schedulability tests for the AR model. A new approach, *multi-bag*, is introduced to give a tighter analysis on the decision of each abort. Firstly, the previous analysis is pessimistic in that higher priority tasks cannot always abort the lower priority task with the biggest execution time. Figure 6.1 shows a case which is improved. Afterwards, the analysis of CRPD was studied to adapt its equations for the

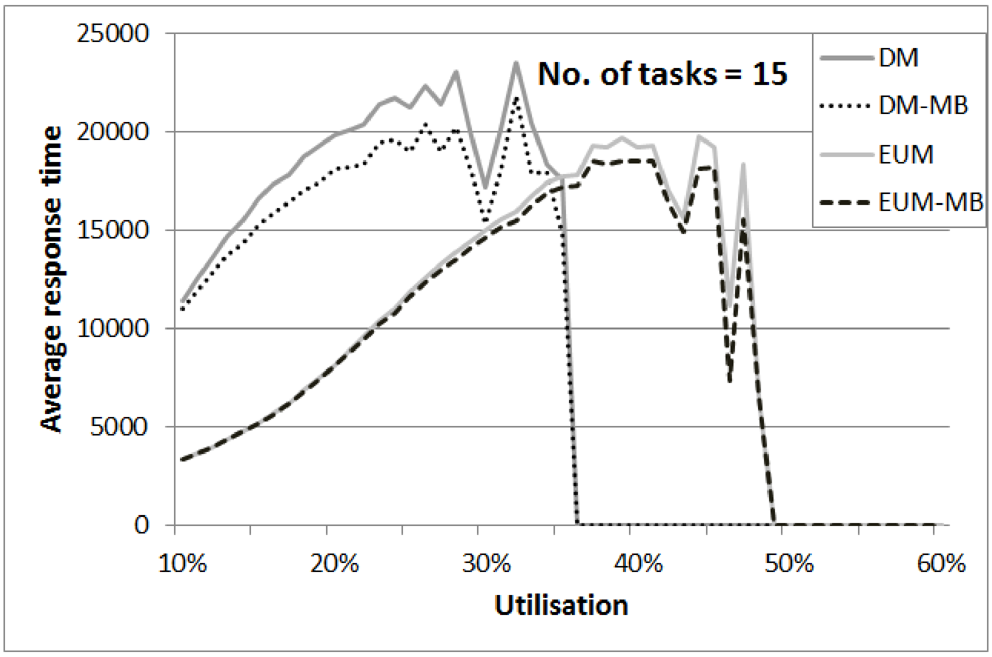

Figure 6.13: Compare average response time of 15-tasks task-sets.

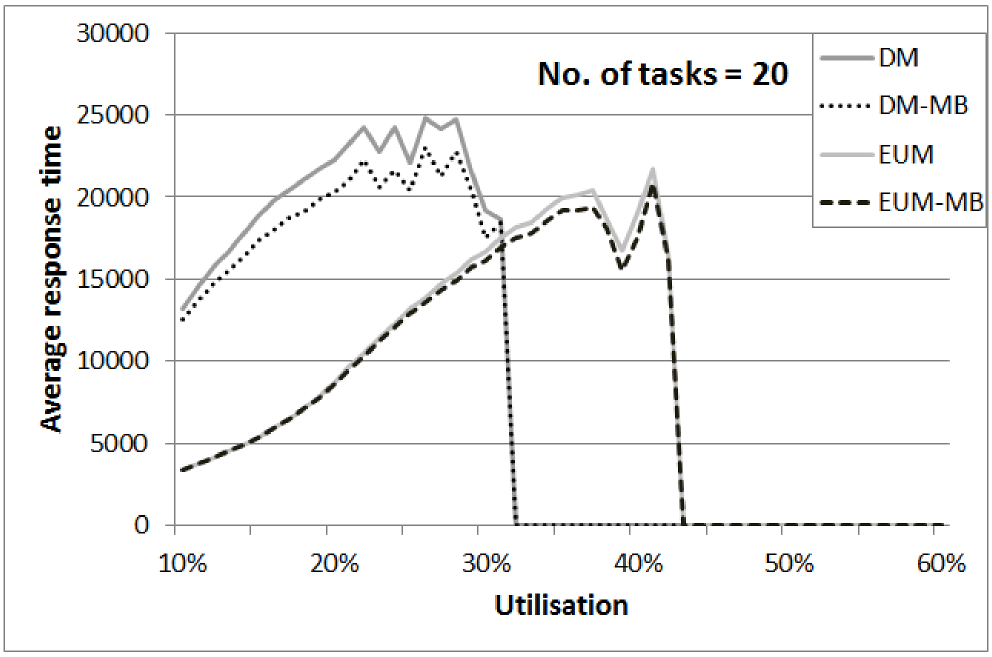

Figure 6.14: Compare average response time of 20-tasks task-sets.

AR model. Finally, the multi-set approach from CRPD was adapted to the AR model and we called it the *multi-bag* approach. Some calculations and some figures are presented. The section of experimental evaluation consists of five results. The $\tilde{C}_j^i$ approach with DM and EUM orderings and the multi-bag approach with DM and EUM orderings are compared with each other. Lastly, an additional experiment is to compare with the exhaustive search priority assignment. In the beginning of this thesis, the simple $\tilde{C}_j^i$ approach was introduced to be heuristic for the AR model. Now the multi-bag approach deals with a tighter analysis on the decision of each abort. Although the evaluations show that the multi-bag approach shows an improvement in terms of schedulability, for randomly generated task-sets it is not major. The next chapter will apply the multi-bag approach to the DA model.

Chapter 7

Multi-bag approach for the DA Model

This chapter considers how to apply the multi-bag approach to the DA model. As Chapter 6 showed, the AR model with the multi-bag approach has an improvement on the response time analysis; the DA model should benefit from this advantage as well. The multi-bag approach can be used to deal with the abort cost for the AR model. Intuitively, the DA model can also use the multi-bag approach to deal with the abort cost if AR regions are adapted to the multi-bag approach.

To illustrate, Figure 7.1 is an example to show how the multi-bag approach applies to the DA model. The grey box is the abort cost. The white box is the AR region and the black box is the non-preemptive region. The upward arrow is a release and the downward arrow is an abort. There are three tasks: τ_1, τ_2 and τ_3 with the priority levels, $P_1 > P_2 > P_3$. τ_1 is a fully non-preemptive task. Both τ_2 and τ_3 have 50% of execution time as AR region and 50% of execution time as non-preemptive region. Firstly, the first job of τ_2 aborts τ_3 before it enters the final non-preemptive region. Afterwards, the first job of τ_1 aborts the first job of τ_2 before it enters the final non-preemptive region and then the first job of τ_2 is finished before the second job of τ_1 releases. For this reason, the second job of τ_1 can only abort τ_3, not τ_2 for a second time.

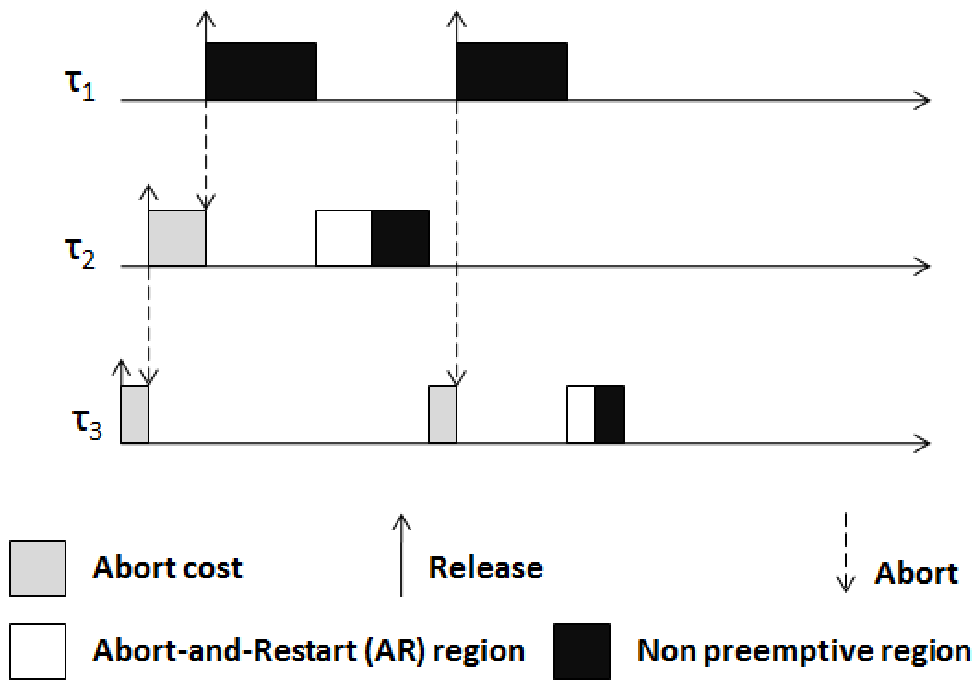

Figure 7.1: The DA model using multi-bag approach.

The above example shows that a higher priority task can only abort a lower priority task before the final non-preemptive region of the lower priority task. By using the multi-bag approach, a higher priority job only aborts a lower priority job which is still waiting for execution. Now we consider new equations to apply the multi-bag approach to the DA model. The following section is the implementation of using the multi-bag approach for the DA model.

7.1 Implementation

This section introduces new equations for the multi-bag approach and the DA model. The equations for the multi-bag approach in Chapter 6 do not consider the AR and the non-preemptive regions, and the equations for the DA model in Chapter 5 are adapted to the $\tilde{C}^i_j$ approach. Firstly, the equations of the multi-bag approach for the DA model are introduced below.

The equation to calculate the abort cost can be directly reused but the symbol $\gamma_{i,j}^{DA}$ is used to indicate this equation is for the DA model. $\gamma_{i,j}^{DA}$ is the abort cost of τ_j when scheduling τ_i.

$$\gamma_{i,j}^{DA} = \sum_{l=1}^{E_j(R_i)} M_{i,j}^{l} \tag{7.1}$$

where $M_{i,j}^{l}$ is the $l-th$ largest value in $M_{i,j}$. To compute a bag, $M_{i,j}$, the equation is given by:

$$M_{i,j} = \bigcup_{k \in hep_{(i)} \bigcap lp_{(j)}} \left(\bigcup_{E_j(R_k) E_k(R_i)} (C_k - F_k) \right) \tag{7.2}$$

As the abort cost may not be the entire execution time in the DA model, the abort cost can be calculated by $C_k - F_k$. C_k is the entire execution time and F_k is the final non-preemptive region. The result of $C_k - F_k$ is actually the AR region of τ_k. Now all the equations for the multi-bag approach are introduced.

For the equations of the DA model using the multi-bag approach, we firstly start with the equation of the priority level-i active period. The equation (in Chapter 5) can be reused.

$$A_i = B_i + \sum_{\forall j \in hep(i)} \left\lceil \frac{A_i}{T_j} \right\rceil \tilde{C}_{i,j}^{DA} \tag{7.3}$$

We note that Equations (7.1) and (7.2) require the value of R_i, and the priority level-i active period cannot compute the response time so the value of $\tilde{C}_{i,j}^{DA}$ is still used here. A bigger priority level-i active period does not affect the schedulability.

For computing the blocking time, the equation does not require any change. The equation is given by:

$$B_i = \max_{\forall l \in lp(i)} (F_l - 1) \tag{7.4}$$

For the calculation of the number of jobs, the equation does not relate to any execution time and abort cost so the same equation can be used.

$$G_i = \left\lceil \frac{A_i}{T_i} \right\rceil \tag{7.5}$$

The equation of starting time of the final non-preemptive region is given by:

$$W_{i,g}^{DA} = B_i + (g+1)C_i - F_i + \sum_{\forall j \in hp_{(i)}} \left(\left\lfloor \frac{W_{i,g}}{T_j} \right\rfloor + 1 \right) C_j + \gamma_{i,j}^{DA} \tag{7.6}$$

The symbol of DA is the indication of the DA model. $W_{i,g}^{DA}$ is the starting time of g-th job of τ_i. The equation uses the value $\gamma_{i,j}^{DA}$ as the total abort cost.

Finally, the worst-case response time can be computed by the below equation.

$$R_i = max_{\forall g=0,1,2...G_i}(W_{i,g}^{DA} + F_i - gT_i) \tag{7.7}$$

The following section will give an example to illustrate the application of these equations.

7.2 Example

This section illustrates how the above equations apply to a task-set and show that the multi-bag approach can schedule the task-set but the $\tilde{C}_{i,j}^{DA}$ approach cannot.

Task	T=D	C	AR	F
τ_1	90	6	0	6
τ_2	240	120	36	84
τ_3	300	4	0	4

Table 7.1: An example task-set.

In Table 7.1, there are three tasks and the top task has the highest priority. The values of the final non-preemptive regions are assigned. The values of

AR regions are included to make the calculation clear.

Task	T=D	C	AR	F	$\tilde{C}^{DA}_{3,j}$
τ_1	90	6	0	6	42
τ_2	240	120	36	84	120
τ_3	300	4	0	4	4

Table 7.2: An example task-set with $\tilde{C}^{DA}_{3,j}$.

Firstly, the $\tilde{C}^{DA}_{i,j}$ approach is applied to this task-set. Table 7.2 is the task-set with the values of $\tilde{C}^{DA}_{3,j}$. For computing the DA task-set with the $\tilde{C}^{DA}_{i,j}$, the calculation for τ_3 is shown as below.

1. $B_3 = \max_{\forall l \in lp(3)} (F_l - 1)$
2. $B_3 = 0$
3. $W_{3,g} = B_3 + (g+1)C_3 - F_3 + \sum_{\forall j \in hp(3)} \left(\left\lfloor \frac{W_{3,g}}{T_j} \right\rfloor + 1 \right) \tilde{C}^{DA}_{3,j}$
4. $W^0_{3,0} = B_3 + (g+1)C_3 - F_3 = 0 + (0+1)4 - 4 = 0$
5. $W^1_{3,0} = 0 + (0+1)4 - 4 + \left(\left\lfloor \frac{0}{90} \right\rfloor + 1 \right) 42 + \left(\left\lfloor \frac{0}{240} \right\rfloor + 1 \right) 120 = 162$
6. $W^2_{3,0} = 0 + (0+1)4 - 4 + \left(\left\lfloor \frac{162}{90} \right\rfloor + 1 \right) 42 + \left(\left\lfloor \frac{162}{240} \right\rfloor + 1 \right) 120 = 204$
7. $W^3_{3,0} = 0 + (0+1)4 - 4 + \left(\left\lfloor \frac{204}{90} \right\rfloor + 1 \right) 42 + \left(\left\lfloor \frac{204}{240} \right\rfloor + 1 \right) 120 = 246$
8. $W^4_{3,0} = 0 + (0+1)4 - 4 + \left(\left\lfloor \frac{246}{90} \right\rfloor + 1 \right) 42 + \left(\left\lfloor \frac{246}{240} \right\rfloor + 1 \right) 120 = 366$

The above calculation shows that the blocking time B_3 is 0. Only the first job of τ_3 is considered so the value of g is 0 and it then computes the starting time of the non-preemptive region for τ_3. The initial value of the $W^0_{3,0}$ is 0 as the task is fully non-preemptive. The calculation ends at the fourth iteration where the value is 366. It is already bigger than the deadline. The task-set is deemed to be not schedulable using the $\tilde{C}^{DA}_{i,j}$ approach.

Now the multi-bag approach is applied to this task-set. As the approach uses the multi-bag to deal with the abort costs, the values of $\tilde{C}^{DA}_{3,j}$ are not necessary. The calculations are given below.

1. $B_1 = \max_{\forall l \in lp(1)} (F_l - 1)$

2. $B_1 = 84 - 1 = 83$

3. $W_{1,g} = B_1 + (g+1)C_1 - F_1 + \sum_{\forall j \in hp(1)} \left(\left\lfloor \frac{W_{1,g}}{T_j} \right\rfloor + 1 \right) C_j + \gamma_{1,j}$

4. $W_{1,0}^0 = B_1 + (g+1)C_1 - F_1 = 83 + (0+1)6 - 6 = 83$

5. $W_{1,0}^1 = 83 + (0+1)6 - 6 = 83$

6. $R_1 = max_{\forall g=0,1,2...G_3}(W_{1,g} + F_1 - gT_1)$

7. $R_1 = max_{\forall g=0}(83 + 6 - (0 \cdot 90)) = 89$

The computation starts with τ_1 that the blocking time is 83, and the starting time is 83. The response time R_1 is 89 and it meets the deadline.

1. $B_2 = \max_{\forall l \in lp(2)} (F_l - 1)$

2. $B_2 = 4 - 1 = 3$

3. $W_{2,g} = B_2 + (g+1)C_2 - F_2 + \sum_{\forall j \in hp(2)} \left(\left\lfloor \frac{W_{2,g}}{T_j} \right\rfloor + 1 \right) C_j + \gamma_{2,j}$

4. $W_{2,0}^0 = B_2 + (g+1)C_2 - F_l = 3 + (0+1)120 - 84 = 39$

5. $W_{2,0}^1 = 39 + \left(\left\lfloor \frac{39}{90} \right\rfloor + 1 \right) 6 + \gamma_{2,1}^{l=1}\{36\} = 81$, see footnote [1]

6. $W_{2,0}^2 = 39 + \left(\left\lfloor \frac{81}{90} \right\rfloor + 1 \right) 6 + \gamma_{2,1}^{l=1}\{36\} = 81$

7. $R_2 = max_{\forall g=0,1,2...G_3}(W_{1,g} + F_1 - gT_1)$

8. $R_2 = max_{\forall g=0}(81 + 84 - (0 \cdot 240)) = 165$

The above steps are the calculation of τ_2. The blocking time is 3 and the starting time of the final non-preemptive region is 81. The value of $\gamma_{2,1}^{l=1}\{36\}$ contains one value, 36, which is the abort from τ_1 to τ_2. After, there is no other release from τ_1. The calculation has stopped at the value 81. The response time of τ_2 is 165.

[1]where $\gamma_{2,1} = M_{2,1}\{AR_2\}, AR_2 = C_2 - F_2 = 36$.

1. $B_3 = \max_{\forall l \in lp(3)} (F_l - 1)$
2. $B_3 = 0$
3. $W_{3,g} = B_3 + (g+1)C_3 - F_3 + \sum_{\forall j \in hp(3)} \left(\left\lfloor \frac{W_{3,g}}{T_j} \right\rfloor + 1 \right) C_j + \gamma_{3,j}$
4. $W^0_{3,0} = B_3 + (g+1)C_3 - F_3 = 0 + (0+1)4 - 4 = 0$
5. $W^1_{3,0} = 0 + \left(\left\lfloor \frac{0}{90} \right\rfloor + 1 \right) 6 + \gamma^{l=1}_{3,1}\{36, 0\} + \left(\left\lfloor \frac{0}{240} \right\rfloor + 1 \right) 120 + \gamma^{l=1}_{3,2}\{0\} = 162$, see footnote [2]
6. $W^2_{3,0} = 0 + \left(\left\lfloor \frac{162}{90} \right\rfloor + 1 \right) 6 + \gamma^{l=\{1,2\}}_{3,1}\{36, 0, 0\} + \left(\left\lfloor \frac{162}{240} \right\rfloor + 1 \right) 120 + \gamma^{l=1}_{3,2}\{0\} = 168$, see footnote [3]
7. $W^3_{3,0} = 0 + \left(\left\lfloor \frac{168}{90} \right\rfloor + 1 \right) 6 + \gamma^{l=\{1,2\}}_{3,1}\{36, 0, 0\} + \left(\left\lfloor \frac{168}{240} \right\rfloor + 1 \right) 120 + \gamma^{l=1}_{3,2}\{0\} = 168$
8. $R_3 = max_{\forall g=0,1,2\ldots G_3}(W_{3,g} + F_1 - gT_3)$
9. $R_3 = max_{\forall g=0}(168 + 4 - (0 \cdot 300)) = 172$

Lastly, the blocking time of τ_3 is 0 and the starting time of the non-preemptive region is 168. At the first iteration, $\gamma_{3,1}$ consists of 36 and 0, and $\gamma_{3,2}$ consists of 0. τ_1 can abort τ_2 and τ_3 so the values of AR_2 and AR_3 are stored into a bag, $M_{3,1}$. τ_2 can only abort τ_3 but τ_3 is fully non-preemptive and then the bag $M_{3,2}$ consists of one value, 0. At the second iteration, there is another release for τ_1 at 90 but τ_2 has already entered its final non-preemptive region. At that point, the values of $\gamma_{3,1}$ contained are 36, 0 and 0. The final value of the starting time is 168 and the worst-case response time is 172.

The above calculations show that the multi-bag approach can schedule the example task-set but the $\tilde{C}^{DA}_{i,j}$ approach cannot. The following section will give some experiments based on this analysis.

[2]where $\gamma_{3,1} = M_{3,1}\{AR_2, AR_3\}, AR_2 = C_2 - F_2 = 36$ and $AR_3 = 0$.

[3]where $l = \{1, 2\}$ because there are two releases and τ_1 can abort two jobs. $\gamma_{3,1} = M_{3,1}\{AR_2, AR_3, AR_3\}$. τ_1 cannot abort τ_2 twice because the second job of τ_1 is released at 90 and τ_2 has entered its final non-preemptive region.$R_2 - F2 = 165 - 84 = 81$.

7.3 Experimental Evaluation

This section shows the results from the experiments of the comparison of the $\tilde{C}_{i,j}^{DA}$ and the multi-bag approaches with EUM and MAXAR priority assignment for the DA model. To be consistent, the structure of the experiment is similar to the previous experiments. The parameters of the experiments are:

- Deadline is equal to period.
- All tasks are periodic.
- A set of N utilisation values U_i was generated by the UUniFast Algorithm [13].
- Task periods were generated between 5000 and 50000 according to a log-uniform distribution[4]. And the computed value T_i is assigned to τ_i.
- Task execution times are: $C_i = U_i \cdot T_i$
- Utilisation for task-sets are ranged between 30% and 60%.
- Each utilisation rate generates 1000 different task-sets, i.e. $U = 30\%$ generates 1000 task-sets, $U = 31\%$ generates another 1000 task-sets, and so on.
- The numbers of tasks are 5, 10, 15 and 20.

For each diagram, the X-axis is Utilisation rate and the Y-axis is the Schedulability rate, i.e. the percentage of task-sets that were deemed schedulable.

In Figure 7.2, there are 4 lines: MAXAR-MB, MAXAR, EUM-MB and EUM. The line of MAXAR-MB means the test used the multi-bag approach with the MAXAR priority assignment for the DA model. The line of MAXAR is the test using the $\tilde{C}_{i,j}^{DA}$ approach with the same MAXAR priority assignment. The line of EUM-MB represents the test using the multi-bag approach

[4]The log-uniform distribution of a variable x is such that ln(x) has a uniform distribution.

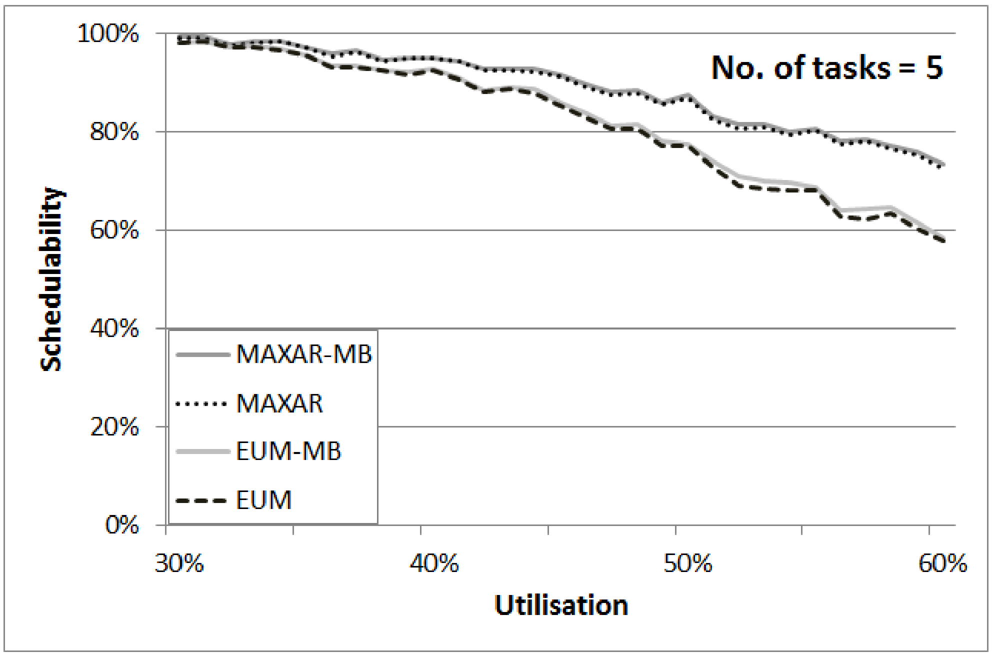

Figure 7.2: The DA model using multi-bag approach with $n = 5$.

with the EUM-MB priority assignment. The line of EUM presents the test using the $\tilde{C}_{i,j}^{DA}$ approach with the EUM priority assignment. A higher schedulability rate means better performance in the experiment. The number of tasks is 5, and the result of MAXAR-MB is slightly better than MAXAR and then EUM-MB is better than EUM.

In Figure 7.3, the number of tasks is 10. The results of MAXAR-MB and MAXAR are much better than the results of EUM-MB and EUM. The difference between MAXAR-MB and MAXAR is less. For the results of EUM-MB and EUM, the difference is still obvious.

Figure 7.4 is the test with 15-tasks task-sets, and the results of MAXAR-MB and MAXAR are better than the results with 10-tasks task-set because increasing the number of tasks improves the schedulability for the DA model with the MAXAR priority assignment (see Chapter 5). The difference between EUM-MB and EUM is less now.

Lastly, Figure 7.5 is the result of 20-tasks task-sets. The results of MAXAR-MB and MAXAR are almost 100% at 50%. The difference between EUM-MB and EUM is decreased.

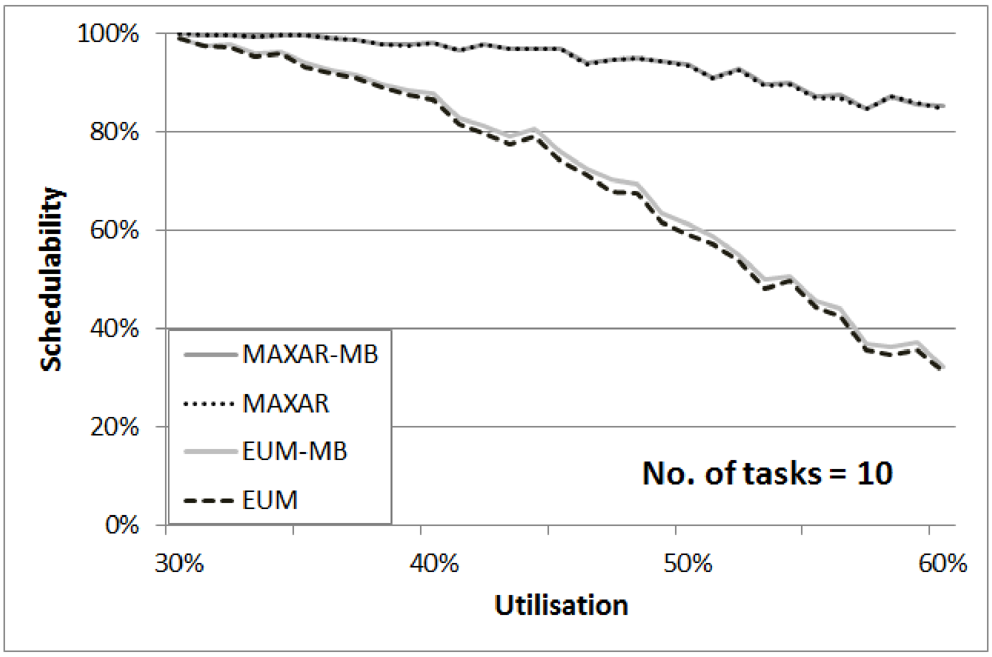

Figure 7.3: The DA model using multi-bag approach with $n = 10$.

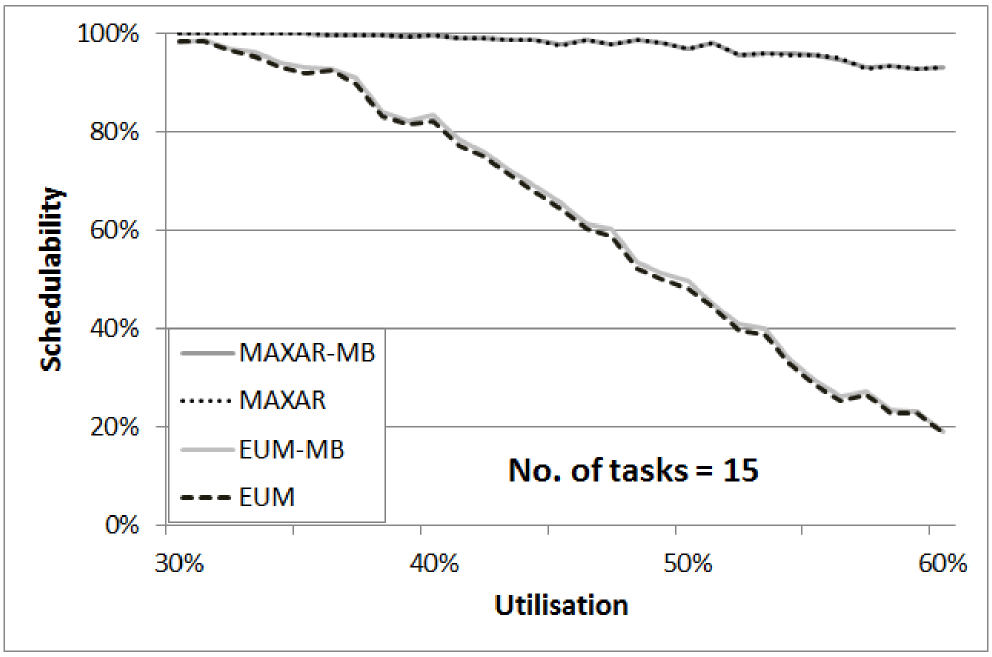

Figure 7.4: The DA model using multi-bag approach with $n = 15$.

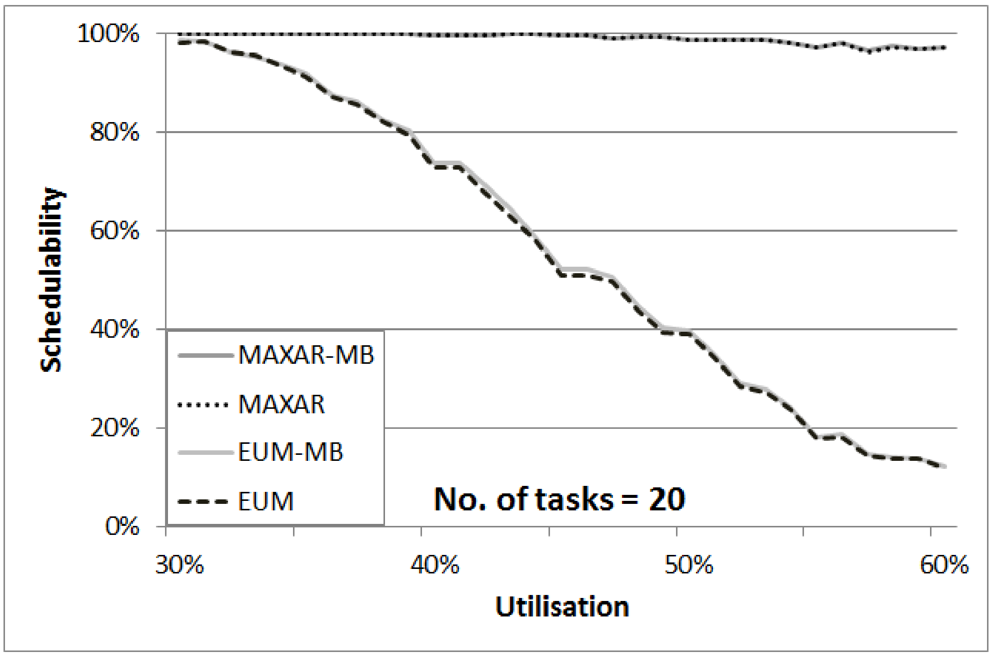

Figure 7.5: The DA model using multi-bag approach with $n = 20$.

As the task-set generator is general, it does not create particular task-sets to show the performance of using the multi-bag approach. As we did in the last chapter, we created another set of diagrams using the same data from the above experiment. For the below diagrams, the X-axis is Utilisation rate and the Y-axis is the Average response time.

In Figure 7.6, there are 4 lines: MAXAR-MB, MAXAR, EUM-MB and EUM. The line of MAXAR-MB means the test using the multi-bag approach with the MAXAR priority assignment for the DA model. The line of MAXAR is the test using the $\tilde{C}^{DA}_{i,j}$ approach with the MAXAR priority assignment. The line of EUM-MB represents the test using the multi-bag approach with the EUM-MB priority assignment. The line of EUM presents the test used the $\tilde{C}^{DA}_{i,j}$ approach with the EUM priority assignment. A smaller average response time has better performance; the best result is MAXAR-MB; the second is MAXAR; the third is EUM; the worst is EUM. Now the difference is much more obvious.

In Figure 7.7, the number of tasks is 10. The ordering of the performance is the same as above, but the difference between EUM and EUM-MB is

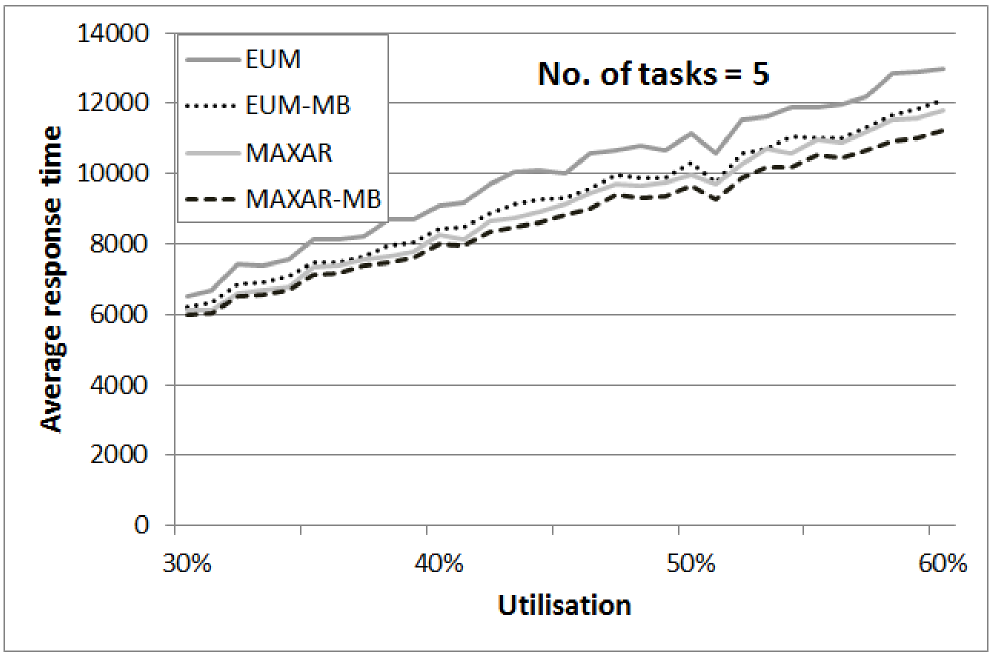

Figure 7.6: DA with multi-bag against average response time with $n = 5$.

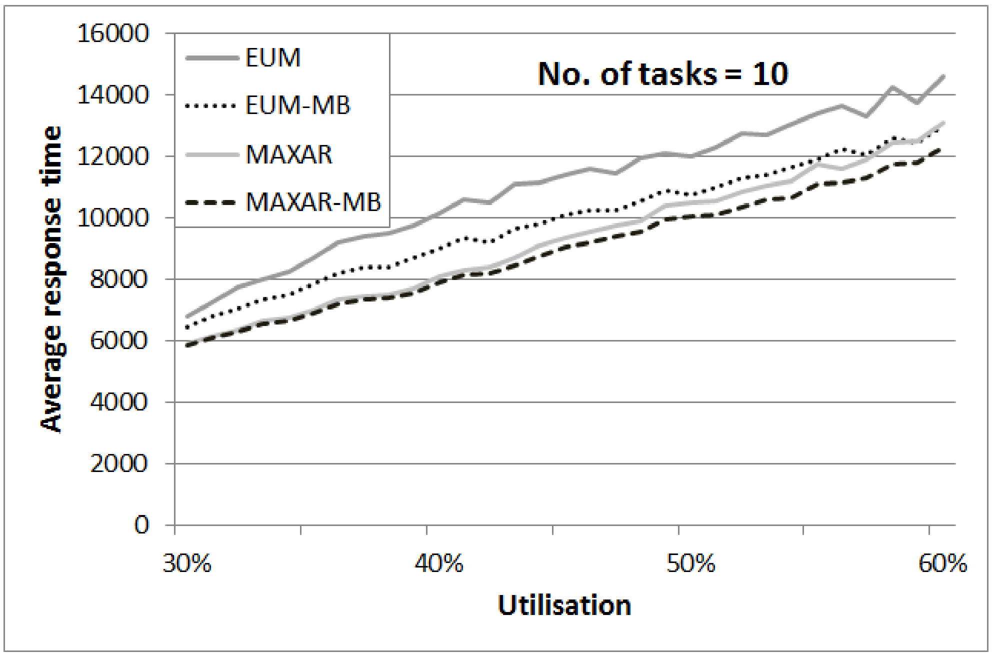

Figure 7.7: DA with multi-bag against average response time with $n = 10$.

bigger.

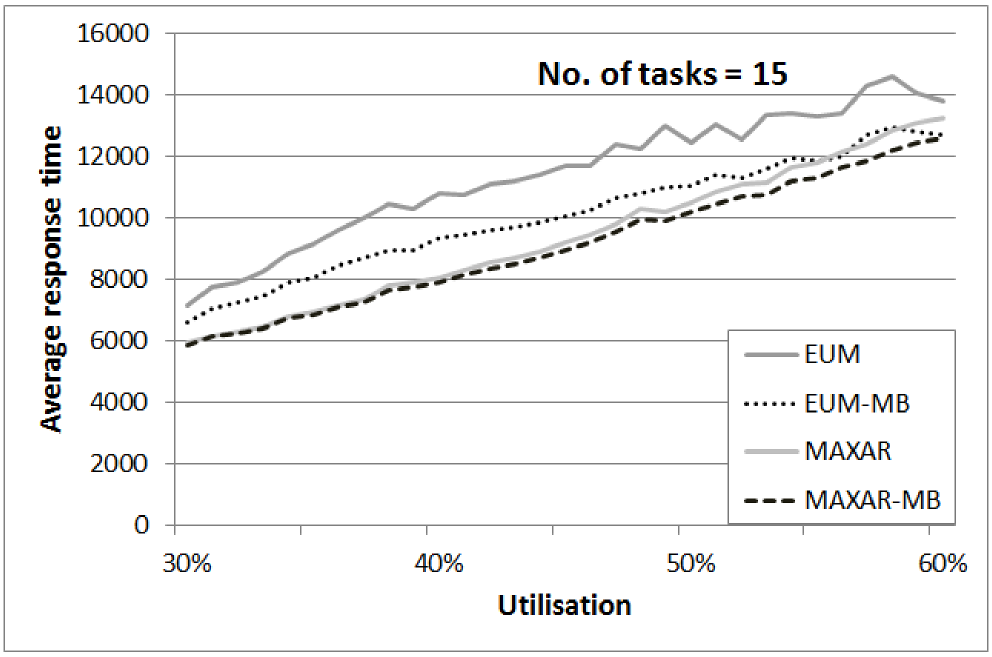

Figure 7.8: DA with multi-bag against average response time with $n = 15$.

Figure 7.8 is the result of the 15-tasks task-set and there is a big gap between EUM and EUM-MB. Overall response times are higher at 60% utilisation. The improvement of MAXAR with the multi-bag approach is obvious.

Lastly, Figure 7.9 shows the result of the 20-tasks task-set. EUM is better than MAXAR after 55% utilisation that is a tolerance as the number of tests for each utilisation is reduced to 1000 times.

To evaluate the results, the multi-bag approach provides visibly better results for the DA model. Although the diagrams with the schedulability rate against utilisation rate do not illustrate this strongly, the latter diagrams show that there are improvements in the response time. For task-sets with a deadline less than period this reduction in response time could be significant.

7.4 Summary

This chapter has completed the research by applying the multi-bag approach to the DA model. Now two techniques have been applied and tested against

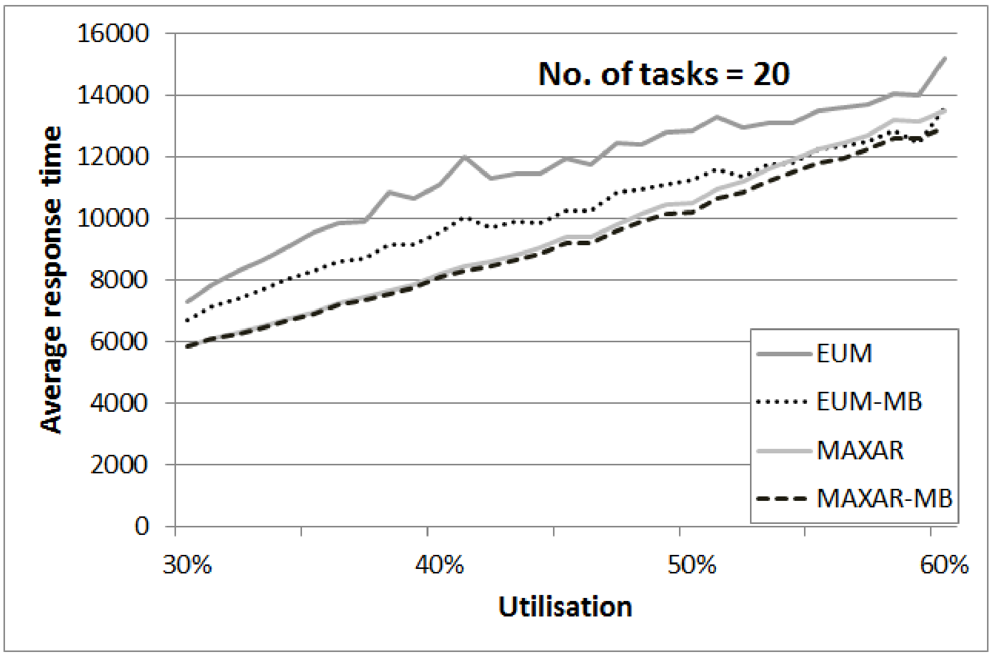

Figure 7.9: DA with multi-bag against average response time with $n = 20$.

each other, and the experiment has shown a better solution for P-FRP. The DA model with multi-bag approach and the MAXAR priority assignment have the best performance regardless of the time complexity of those algorithms.

Chapter 8

Conclusions and Future Work

In this thesis, we have considered the problem that while the AR model can deal effectively with P-FRP in terms of the problems of resource usage, applicable schedulability analysis has not been demonstrated for this model. This thesis contends that it is possible to derive an appropriate scheduling approach for the model.

8.1 Summary and Conclusions

Chapter 1 introduced real-time systems, priority-based functional reactive programming, the AR model and the motivation for this research. It addressed the problem of the AR model mentioned above.

Chapter 2 was the related work of this research. The system model was introduced, and real-time system scheduling and the AR model have been studied. According to the related work, the AR model could not be scheduled effectively for P-FRP.

Chapter 3 analysed the schedulability of the AR model, which consists of finding the critical instant and developing schedulability tests. For the critical instant, the thesis has shown that finding the critical instant for the AR model with periodic and sporadic tasks is intractable. For schedulability tests, a new formulation was introduced and we called it the $\tilde{C}_j^i$ approach. It reduced the complexity for the further analysis.

Chapter 4 introduced an improved priority assignment for the AR model. We called the priority assignment *Execution-time-towards-Utilisation-time (EUM)*. The analysis took the benefit of the $\tilde{C}^i_j$ approach that EUM requires less computation time and provides the priority assignment close to the exhaustive search algorithm which is optimal but intractable.

Chapter 5 introduced an alternative scheme to improve the schedulability for the AR model, and we called it the *deferred abort (DA)* model. The NP model can be used for P-FRP but it does not dominate the AR model. By adapting the technique of deferred preemption, the combination of AR regions and non-preemptive regions has defined the DA model. For the priority assignment, a heuristic algorithm was introduced and we called it the *MAXAR* algorithm. Lastly, the experimental evaluation showed that the DA model has a big improvement in reducing the number of aborts.

Chapter 6 introduced a tighter analysis on schedulability tests for the AR model as the $\tilde{C}^i_j$ approach was too pessimistic sometimes. The technique of CRPD analysis was studied as the multi-set approach could be applied to the AR model. The technique of the multi-set approach was adapted to a new approach; we called it the *multi-bag* approach. This approach can analyse each abort of individual jobs of higher priority tasks. In experimental evaluation, the results showed the multi-bag approach could improve the response time analysis.

Chapter 7 applied the multi-bag approach to the DA model. New equations were introduced for the combination of both techniques. An example task-set showed the multi-bag approach dominated the $\tilde{C}^{DA}_{i,j}$ approach in the DA model. In experimental evaluation, the results showed an improvement after using the multi-bag approach for both schedulability rate and worst-case response time. Contributions are summarised as follows.

Critical Instant — Finding the critical instant for the AR model with periodic and sporadic tasks is intractable.

New formulation for scheduling — This is introduced and can be applied to the standard response time analysis for the AR model.

New priority assignment schemes — New priority assignment schemes are developed for both the AR and DA models, and they have good performance and are tractable for large systems.

Deferred Abort (DA) model — This model provides better schedulability and dominates the non-preemptive model.

Multi-bag approach — This approach offers a tighter analysis on scheduling task-sets under both AR and DA models.

These contributions combine to demonstrate the correctness of the thesis hypothesis.

To conclude all the results, we created three more figures to show how the research improved the schedulability at each stage. The best result from each chapter is extracted. There are five lines: 1) DA-MB represents the result of the DA model using the multi-bag approach and the MAXAR priority assignment. 2) DA-MAXAR represents the result of the DA model using the $\tilde{C}_{i,j}^{DA}$ approach and the MAXAR priority assignment. 3) AR-MB represents the result of the AR model using the multi-bag approach and the EUM priority assignment. 4) AR-EUM represents the result of the AR model using the $\tilde{C}_j^i$ approach and the EUM priority assignment. 5) AR-RM represents the result of the AR model using the $\tilde{C}_j^i$ approach and the RM priority assignment.

Figures 8.1, 8.2 and 8.3 show the results of 5-tasks, 10-tasks and 15-tasks task-sets. The lines of DA-MB and DA-MAXAR are always at the top. The lines of AR-MB and AR-EUM are at the middle and the line of AR-RM is at the bottom. The result of AR-RM is from Chapter 2 and represents the state-of-the-art when this research commenced. The schedulability is very low even in the 5-tasks task-set. In Chapter 4, a better priority assignment was introduced so the result of AR-EUM is improved. The result of AR-MB is from Chapter 6 and it is difficult to show the improvement from this diagram. Chapter 5 introduced the technique of DA and then the schedulability had a big improvement, as shown on the line of DA-MAXAR. Finally, Chapter 7 applies the multi-bag approach to the DA model. The result of DA-MB

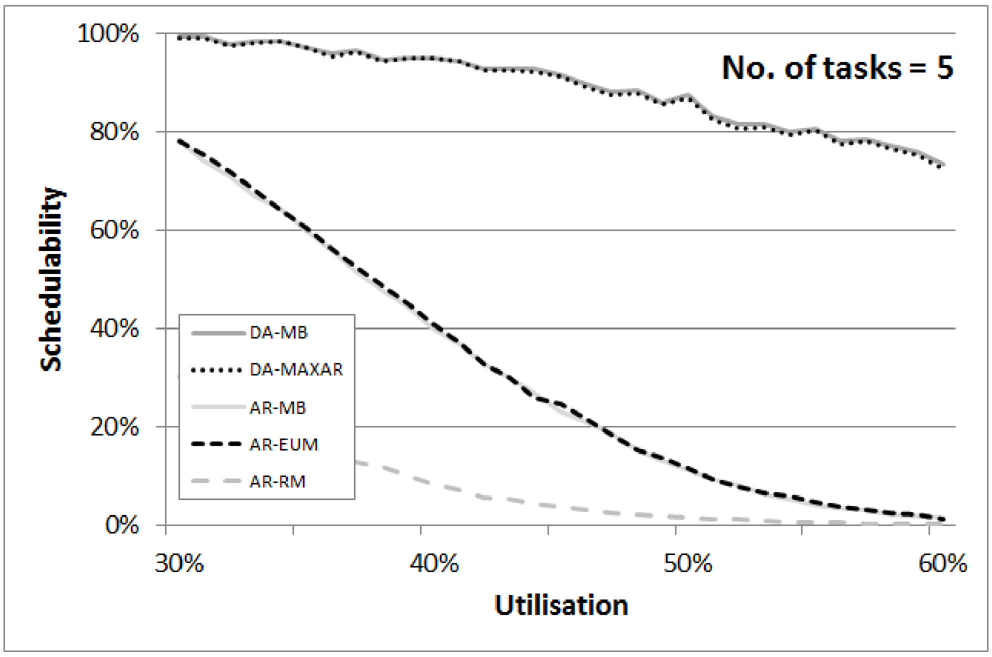

Figure 8.1: Compare overall improvement with $n = 5$.

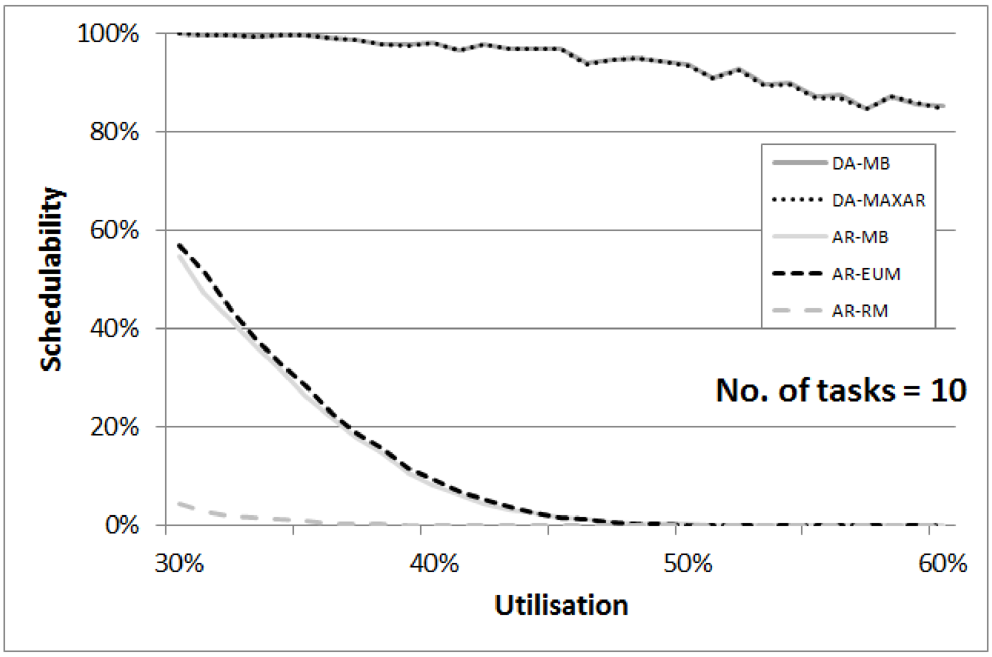

Figure 8.2: Compare overall improvement with $n = 10$.

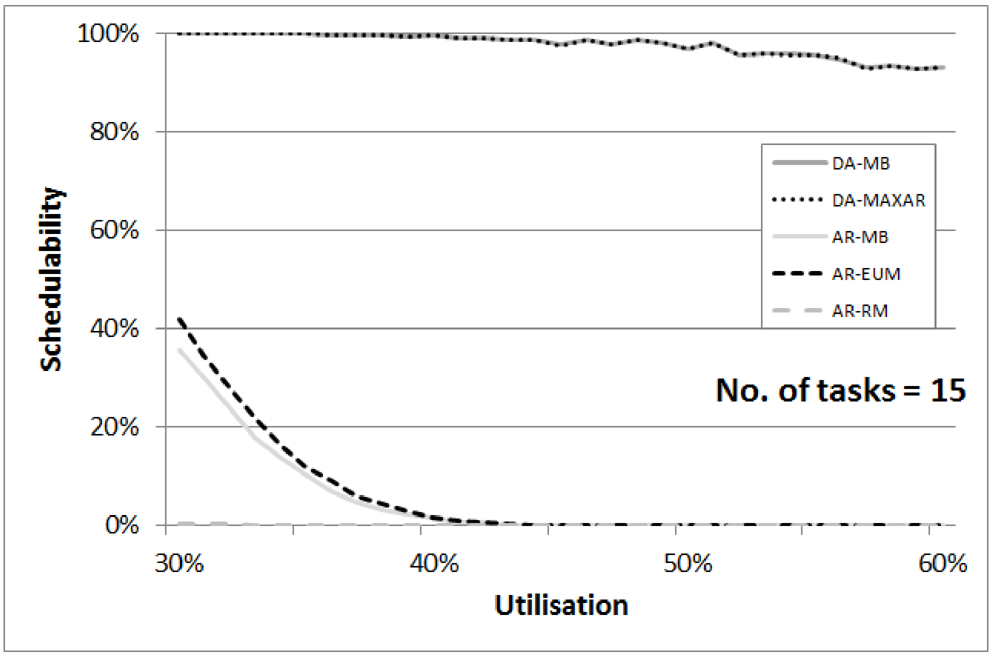

Figure 8.3: Compare overall improvement with $n = 15$.

is the best. Again, the results of this thesis combine to demonstrate the correctness of the thesis hypothesis.

8.2 Future Work

This section lists the possible future work items for this thesis. This research is based on the fixed priority scheduling on one processor systems with non-shared resources. The technique of AR can apply to other scheduling, such as Earliest Deadline First (EDF) and First-in First-out (FIFO). For shared resources systems, there are many solutions for the classical system but it causes bigger response times. The technique of AR does not face this problem. It can provide better schedulability if better scheduling methods are found. To employ P-FRP, the technique of AR is a scheme because the preemptive model cannot be used. The preemptive model is designed for non-atomic required programming. If the objective is reversed, the technique of AR can apply to both types of programming.

EDF scheduling

Earliest Deadline First (EDF) is dynamic scheduling where the priorities of tasks are according to their absolute deadlines. A task with a short absolute deadline executes first. Theoretically, EDF can reach 100% utilisation in a uniprocessor system. If the AR model uses EDF scheduling, it should improve the schedulability.

FIFO scheduling

First-in First-out (FIFO) scheduling executes tasks in the order they arrive, and other tasks have to wait (non-preemptive). In general, there is no priority in FIFO scheduling. For improving the schedulability for the AR model, fixed priority FIFO (FP/FIFO) scheduling can be used. Sometimes two tasks cannot suffer abort costs from each other. In this case, FP/FIFO scheduling can remove the problem by assigning the same priority to both tasks.

Multiprocessor

This thesis has only considered single processor systems. For multiprocessor systems, the standard AR model aborts all current executing tasks (in different processors) when a higher priority task (in one of processors) is release. A new model for multiprocessor systems is required to consider task allocations for different processors, then other processors do not interfere with an abort from a processor.

Glossary of Terms

Abort cost In the AR model, a higher priority task aborts a lower priority task and if the lower priority has already executed for some time, the time is wasted and is termed as an abort cost.

Abort-and-restart A lower priority task is aborted by a higher priority task. When the higher priority task is completed, the lower priority task restarts the execution from the beginning.

Arbitrary deadline The deadline of a task can be less than, equal to or larger than its period.

Blocking time The length of time a higher priority task is delayed by a lower priority task.

Constrained deadline The deadline of a task is no larger than its period.

Deferred abort If a higher priority task is released after the lower priority task has entered its final non-preemptive non-abort region, the higher priority task cannot abort.

Deferred preemption If a higher priority task is released after the lower priority task has entered its final non-preemptive region, the higher priority task cannot preempt.

Utilisation Monotonic (UM) Assigns a higher priority to a task which has a higher utilisation rate.

Execution-time Monotonic (EM) Assigns a higher priority to a task which has a bigger worst-case execution time.

Execution-time-toward-Utilisation Monotonic (EUM) Starts with EM ordering and moves towards to UM.

Multi-bag A series of bags where a bag contains a series of values.

Final non-preemptive region A task is assigned a region where it is non-preemptive.

Implicit deadline The deadline of a task is equal to its period.

Notations

τ_i an arbitrary task, 25

C_i the worst-case execution time for τ_i (also referred to as WCET), 31

P_i priority for τ_i,27

B_i blocking time for τ_i, 33

N the number of tasks, 31

U_i the utilisation of τ_i, 31

R_i response time for τ_i, 32

$hp(i)$ any task has higher priority than τ_i, 33

W_i^n the n-th step of iterations for τ_i. 33

α_i the total abort cost for τ_i, 40

$\max\limits_{k=i}^{j-1} C_k$ the biggest execution time task τ_k, 40

Γ_N a task-set with N tasks, 71

$hep(i)$ any task has priority higher than or equal to τ_i, 73

$lp(i)$ any task has priority lower than τ_i, 73

$\tilde{C}_j^i$ the new value for the WCET of τ_j, the biggest abort cost is picked between τ_j and τ_i, 73

A_i the priority level-i active period, 95

F_i the length of the final non-preemptive region for τ_i, 95

G_i the number of jobs for τ_i, 95

$W_{i,g}$ the starting time of the final non-preemptive region of g-th job for τ_i, 95

$W_{i,g}^{NP}$ the starting time of the final non-preemptive region of g-th job for τ_i for the non-preemptive model, 95

$\tilde{C}_{i,j}^{DA}$ the new value for the WCET of τ_j, the biggest abort cost is picked between τ_j and τ_i, for the deferred abort model, 99

J_i the jitter for τ_i, 118

$\gamma_{i,j}$ the total abort cost of τ_j when scheduling τ_i, 118

$M_{i,j}$ a bag contains a series of abort costs for τ_j when scheduling τ_i, 120

$E_j(R_i)$ the number of release for τ_j within the response time R_i, 120

$\gamma_{i,j}^{DA}$ the total abort cost of τ_j when scheduling τ_i for the DA model, 135

$W_{i,g}^{DA}$ the starting time of the final non-preemptive region of g-th job for τ_i for the DA model, 136

Bibliography

[1] L. Almeida and J.A. Fonseca. Analysis of a Simple Model for Non-Preemptive Blocking-Free Scheduling. In *Proceedings ECRTS*, Washington, DC, USA, 2001. IEEE Computer Society.

[2] S. Altmeyer, R.I. Davis, and C. Maiza. Cache related pre-emption delay aware response time analysis for fixed priority pre-emptive systems. In *Proceedings RTSS*, pages 261–271, Nov 2011.

[3] S. Altmeyer, R.I. Davis, and C. Maiza. Improved cache related pre-emption delay aware response time analysis for fixed priority pre-emptive systems. *Real-Time Systems*, 48(5):499–526, 2012.

[4] J.H. Anderson, S. Ramamurthy, and K. Jeffay. Real-time computing with lock-free shared objects. In *Proceedings RTSS*, pages 28–37, December 1995.

[5] N. Audsley. Optimal priority assignment and feasibility of static priority tasks with arbitrary start times. Technical report, University of York, 1991.

[6] N.C. Audsley. On Priority Assignment in Fixed Priority Scheduling. *Information Processing Letters*, 79(1):39–44, 2001.

[7] C. Belwal and A.M.K. Cheng. On Priority Assignment in P-FRP. In *Proceedings RTAS (WIP)*, pages 45–48, 2010.

[8] C. Belwal and A.M.K Cheng. Reducing the Number of Preemptions in P-FRP. In *Proceedings RTSS (WIP)*, 2010.

[9] C. Belwal and A.M.K. Cheng. Determining Actual Response Time in P-FRP. In Ricardo Rocha and John Launchbury, editors, *Practical Aspects of Declarative Languages*, volume 6539 of *Lecture Notes in Computer Science*, pages 250–264. Springer Berlin / Heidelberg, 2011.

[10] C. Belwal and A.M.K. Cheng. Determining Actual Response Time in P-FRP Using Idle-Period Game Board. In *Proceedings Object-Oriented Real-Time Distributed Computing*, pages 136–143, Los Alamitos, CA, USA, 2011. IEEE Computer Society.

[11] C. Belwal and A.M.K. Cheng. Feasibility Interval for the Transactional Event Handlers of P-FRP. In *Proceedings Trust, Security and Privacy in Computing and Communications (TrustCom)*, pages 966–973, November 2011.

[12] A.A. Bertossi and M.A. Bonuccelli. Preemptive Scheduling of Periodic Jobs in Uniform Multiprocessor Systems. *Information Processing Letters*, 16(1):3–6, January 1983.

[13] E. Bini and G. Buttazzo. Measuring the Performance of Schedulability Tests. *Real-Time Systems*, 30:129–154, 2005.

[14] K. Bletsas and N. Audsley. Optimal priority assignment in the presence of blocking. *Inf. Process. Lett. 99*, 3:83–86, Aug 2006.

[15] R.J. Bril, J.J. Lukkien, and W.F.J. Verhaegh. Worst-case response time analysis of real-time tasks under fixed-priority scheduling with deferred preemption revisited. In *Proceedings ECRTS*, pages 269–279, Washington, DC, USA, 2007. IEEE Computer Society.

[16] R.J. Bril, J.J. Lukkien, and W.F.J. Verhaegh. Worst-case response time analysis of real-time tasks under fixed-priority scheduling with deferred preemption. *Real-Time Systems*, 42(1-3):63–119, August 2009.

[17] A. Burns. Is Audsley's Scheme the Most Expressive Optimal Priority Assignment Algorithm? In *Proceedings RTSOPS (ECRTS)*, pages 8–11, 2013.

[18] A. Burns and A.J. Wellings. *Real-time systems and programming languages*. Pearson Education, 4th edition, 2009.

[19] J.V. Busquets-Mataix, J.J. Serrano, R. Ors, P. Gil, and A.J. Wellings. Adding instruction cache effect to schedulability analysis of preemptive real-time systems. In *Proceedings Real-Time Technology and Applications Symposium*, pages 204–212, June 1996.

[20] G.C. Buttazzo, M. Bertogna, and Gang Yao. Limited preemptive scheduling for real-time systems. a survey. *IEEE Transactions on Industrial Informatics*, 9(1):3–15, Feb 2013.

[21] J. Byun, A. Burns, and A.J. Wellings. A Worst-Case Behavior Analysis for Hard Realtime transactions. pages 144–149, 1996.

[22] M. Caccamo and L. Sha. Aperiodic Servers with Resource Constraints. In *Proceedings RTSS*, page 161, Los Alamitos, CA, USA, 2001. IEEE Computer Society.

[23] P. Chen and W. Wonham. Real-Time Supervisory Control of a Processor for Non-Preemptive Execution of Periodic Tasks. *Real-Time Systems*, 23:183–208, Mar 2002.

[24] T.H. Cormen, C.E. Leiserson, R.L. Rivest, and C. Stein. *Introduction to Algorithms*, chapter Red-Black Trees, pages 273–301. MIT Press and McGraw-Hill, second edition, 2001.

[25] A. Courtney. Frappé: Functional reactive programming in Java. In *Proceedings PADL*, March 2001.

[26] R.I. Davis and M. Bertogna. Optimal fixed priority scheduling with deferred pre-emption. In *Proceedings RTSS*, pages 39–50, December 2012.

[27] R.I. Davis and A. Burns. Priority Assignment for Global Fixed Priority Pre-Emptive Scheduling in Multiprocessor Real-Time Systems. In *Proceedings RTSS*, pages 398–409, 2009.

[28] R.I. Davis and A. Burns. Priority assignment for global fixed priority pre-emptive scheduling in multiprocessor real-time systems. In *Proceedings RTSS*, pages 398–409, December 2009.

[29] R.I. Davis and A. Burns. Improved priority assignment for global fixed priority pre-emptive scheduling in multiprocessor real-time systems. *Real-Time Systems*, pages 1–40, 2010.

[30] R.I. Davis, K.W. Tindell, and A. Burns. Scheduling Slack Time in Fixed Priority Pre-emptive Systems. In *Proceedings RTSS*, pages 222–231, December 1993.

[31] D. de Niz, L. Abeni, S. Saewong, and R.R. Rajkumar. Resource sharing in reservation-based systems. In *Proceedings RTSS*, Washington, DC, USA, 2001. IEEE Computer Society.

[32] R. Dobrin and G. Fohler. Reducing the Number of Preemptions in Fixed Priority Scheduling. In *Proceedings ECRTS*, pages 144–152, Washington, DC, USA, 2004. IEEE Computer Society.

[33] C. Ekelin. Clairvoyant Non-Preemptive EDF Scheduling. In *Proceedings ECRTS*, pages 23–32, Washington, DC, USA, 2006. IEEE Computer Society.

[34] C. Elliott. Modeling interactive 3d and multimedia animation with an embedded language. In *Proceedings of the Conference on Domain-Specific Languages on Conference on Domain-Specific Languages (DSL)*, page 22. USENIX Association, 1997.

[35] C. Elliott and P. Hudak. Functional reactive animation. In *Proceedings of the Second ACM SIGPLAN International Conference on Functional Programming*, pages 263–273. ACM, 1997.

[36] G. Fohler. Joint scheduling of distributed complex periodic and hard aperiodic tasks in statically scheduled systems. In *Proceedings RTSS*, pages 152–161, December 1995.

[37] L. Georges, P. Mühletahler, and N. Rivierre. A Few Results on Non-Preemptive Real-Time Scheduling. Research Report RR-3926, INRIA, 2000.

[38] T.M. Ghazalie and T.P. Baker. Aperiodic Servers in a Deadline Scheduling Environment. *Real-Time Systems*, 9(1):31–67, July 1995.

[39] R. R. Howell and M. K. Venkatrao. On Non-Preemptive Scheduling of Recurring Tasks Using Inserted Idle Times. *Information and Computation*, 117(1):50–62, Feb 1995.

[40] P. Hudak. *The Haskell School of Expression – Learning Functional Programming through Multimedia.* Cambridge University Press, New York, 2000.

[41] D. Isovic and G. Fohler. Handling sporadic tasks in off-line scheduled distributed hard real-time systems. In *Proceedings ECRTS*, pages 60–67, 1999.

[42] D. Isovic and G. Fohler. Online handling of hard aperiodic tasks in time triggered systems. In *Proceedings ECRTS*, June 1999.

[43] D. Isovic and G. Fohler. Efficient scheduling of sporadic, aperiodic, and periodic tasks with complex constraints. In *Proceedings RTSS*, pages 207–216, Washington, DC, USA, 2000. IEEE Computer Society.

[44] K. Jeffay. Analysis of a Synchronization and Scheduling Discipline for Real-Time Tasks with Preemption Constraints. In *Real Time Systems Symposium, 1989., Proceedings.*, pages 295–305, December 1989.

[45] K. Jeffay. Scheduling sporadic tasks with shared resources in hard real-time systems. In *Proceedings RTSS*, pages 89–99, December 1992.

[46] K. Jeffay, D.F. Stanat, and C.U. Martel. On Non-Preemptive Scheduling of Periodic and Sporadic Tasks. In *Proceedings RTSS*, pages 129–139, December 1991.

[47] M. Joseph and P. Pandya. Finding Response Times in a Real-Time System. *BCS Computer Journal*, 29(5):390–395, 1986.

[48] R. Kaiabachev, W. Taha, and A. Yun Zhu. E-frp with priorities. In *Proceedings of the 7th ACM &Amp; IEEE International Conference on Embedded Software*, pages 221–230. ACM, 2007.

[49] S. Kim, S. Hong, and T.H. Kim. Integrating real-time synchronization schemes into preemption threshold scheduling. In *Proceedings Object-Oriented Real-Time Distributed Computing*, pages 145–152, 2002.

[50] S. Kim, S. Hong, and T.H. Kim. Perfecting preemption threshold scheduling for object-oriented real-time system design: From the perspective of real-time synchronization. In *Proceedings of the Joint Conference on Languages, Compilers and Tools for Embedded Systems: Software and Compilers for Embedded Systems*, pages 223–232. ACM, 2002.

[51] C.M. Krishna. *Real-Time Systems*. McGraw-Hill Higher Education, 1st edition, 1996.

[52] T. Kuo, W. Yang, and K. Lin. EGPS: A Class of Real-Time Scheduling Algorithms Based on Processor Sharing. In *Proceedings ECRTS*, volume 0, page 27, Los Alamitos, CA, USA, 1998. IEEE Computer Society.

[53] C.G. Lee, J. Hahn, Y.M. Seo, S.L. Min, R. Ha, S. Hong, C.Y. Park, M. Lee, and C.S. Kim. Analysis of cache-related preemption delay in fixed-priority preemtive scheduling. *IEEE Transactions on Computers*, 47(6):700–713, June 1998.

[54] C.G. Lee, K. Lee, J. Hahn, Y.M. Seo, S.L. Min, R. Ha, S. Hong, C.Y. Park, M. Lee, and C.S. Kim. Bounding cache-related preemption delay for real-time systems. *Software Engineering*, 27(9):805–826, Sep 2001.

[55] J. Lehoczky, L. Sha, and Y. Ding. The Rate-Monotonic Scheduling Algorithm: Exact Characterization and Average Case Behaviour. In *Proceedings RTSS*, pages 166–171, December 1989.

[56] J.P. Lehoczky and S. Ramos-Thuel. An Optimal Algorithm for Scheduling Soft-Aperiodic Tasks in Fixed-Priority Preemptive Systems. *Proceedings Real-Time System Symposium*, pages 110–123, December 1992.

[57] J.P. Lehoczky, L. Sha, and J.K. Strosnider. Enhanced Aperiodic Responsiveness in Hard Real-Time Environments. *Proceedings IEEE Real-Time System Symposium*, pages 261–270, 1987.

[58] J.Y.T. Leung and M.L. Merrill. A Note on Preemptive Scheduling of Periodic, real-time tasks. *Information Processing Letters*, 11(3):115–118, November 1980.

[59] C.L. Liu and J.W. Layland. Scheduling Algorithms for Multiprogramming in a Hard Real-Time Environment. *JACM*, 20(1):46–61, 1973.

[60] C.D. Locke, T.J. Mesler, and D.R. Vogel. Replacing passive tasks with Ada9X protected records. *Ada Lett.*, XIII:91–96, March 1993.

[61] C.D. Locke, D.R. Vogel, and T.J. Mesher. Building a Predictable Avionics Platform in Ada: A Case Study. In *Proceedings RTSS*, pages 181–189, December 1991.

[62] J. Manson, J. Baker, A. Cunei, S. Jagannathan, M. Prochazka, B. Xin, and J. Vitek. Preemptible atomic regions for real-time Java. In *Proceedings RTSS*, pages 10–71, 2005.

[63] M. Marouf and Y. Sorel. Scheduling non-preemptive hard real-time tasks with strict periods. In *Emerging Technologies Factory Automation (ETFA), 2011 IEEE 16th Conference on*, pages 1–8, Sept 2011.

[64] O. Ulusoy and G. Belford. Real-time transaction scheduling in database systems. *Information Systems*, 18(8):559–580, 1993.

[65] J. Peterson, G.D. Hager, and P. Hudak. A language for declarative robotic programming. In *Proceedings 1999 IEEE International Conference on Robotics and Automation*, volume 2, pages 1144–1151, 1999.

[66] J. Peterson, P. Hudak, and C. Elliott. Lambda in motion: Controlling robots with haskell. In *Proceedings PADL (workshop)*, pages 91–105. Springer-Verlag, 1998.

[67] P. Puschner and C. Koza. Calculating the maximum execution time of real-time programs. *Journal of Real-Time Systems*, 1(2):159–176, Sep 1989.

[68] J. Ras and A.M.K. Cheng. An evaluation of the dynamic and static multiprocessor priority ceiling protocol and the multiprocessor stack resource policy in an smp system. In *Real-Time and Embedded Technology and Applications Symposium*, pages 13–22, April 2009.

[69] J. Ras and A.M.K. Cheng. Response Time Analysis for the Abort-and-Restart Task Handlers of the Priority-Based Functional Reactive Programming (P-FRP) Paradigm. In *IEEE International Conference on Embedded and Real-Time Computing Systems and Applications*, pages 305–314, August 2009.

[70] J. Ras and A.M.K. Cheng. Response Time Analysis of the Abort-and-Restart Model under Symmetric Multiprocessing. In *Proceedings CIT*, pages 1954–1961, 2010.

[71] A. Reid, J. Peterson, P. Hudak, and G.D. Hager. Prototyping Real-Time Vision Systems: An Experiment in DSL Design. In *Proceedings ICSE*, May 1999.

[72] M. Sage. FranTk — a Declarative GUI Language for Haskell. In *Proceedings of the Fifth ACM SIGPLAN International Conference on Functional Programming*, pages 106–117. ACM, 2000.

[73] J. Schneider. Cache and pipeline sensitive fixed priority scheduling for preemptive real-time systems. In *Proceedings RTSS*, pages 195–204, 2000.

[74] L. Sha, J.P. Lehoczky, and R. Rajkumar. Solutions for some Practical Problems in Prioritised Preemptive Scheduling. In *Proceedings RTSS*, pages 181–191, 1986.

[75] L. Sha, R. Rajkumar, and J.P. Lehoczky. Priority inheritance protocols: an approach to real-time synchronization. *IEEE Transactions on Computers*, 39(9):1175–1185, September 1990.

[76] L. Sha, B. Sprunt, and J.P. Lehoczky. Aperiodic Task Scheduling for Hard Real-Time Systems. *Real-Time Systems*, 1:27–69, 1989.

[77] J.A. Stankovic. Misconceptions about real-time computing: a serious problem for next-generation systems. *Computer*, 21(10):10–19, October 1988.

[78] J.K. Strosnider, J.P. Lehoczky, and L. Sha. The deferrable server algorithm for enhanced aperiodic responsiveness in hard real-time environments. *IEEE Transactions on Computers*, 44(1), January 1995.

[79] H. Takada and K. Sakamura. Real-time synchronization protocols with abortable critical sections. In *Proceedings of International Workshop on Real-time Computing Systems and Application*, pages 48–52, 1994.

[80] K. Tindell, A. Burns, and A.J. Wellings. An Extendible Approach for Analyzing Fixed Priority Hard Real-Time Tasks. *Real-Time Systems*, 6(2):133–151, March 1994.

[81] Z. Wan and P. Hudak. Functional reactive programming from first principles. In *Proceedings of the ACM SIGPLAN 2000 Conference on Programming Language Design and Implementation*, pages 242–252. ACM, 2000.

[82] Z. Wan, W. Taha, and P. Hudak. Event-driven FRP. In *Proceedings PADL*, January 2002.

[83] Y. Wang and M. Saksena. Scheduling fixed priority tasks with preemption threshold. *IEEE International Conference on Real-Time Computing Systems and Applications*, pages 328–335, December 1999.

[84] H.C. Wong and A. Burns. Improved Priority Assignment for the Abort-and-Restart (AR) Model. Technical Report YCS-2013-481, University of York, Department of Computer Science, 2013.

[85] H.C. Wong and A. Burns. Improved Priority Assignment for the Abort-and-Restart (AR) Model. In *Proceedings JRWRTC (RTNS)*, 2013.

[86] H.C. Wong and A. Burns. Schedulability Analysis for the Abort-and-Restart (AR) Model. In *Proceedings RTNS*. ACM, 2014.

[87] W. Zhao, K. Ramamritham, and J.A. Stankovic. Scheduling Tasks with Resource Requirements in a Hard Real-Time System. *IEEE Transactions on Software Engineering*, 13(5):564–577, May 1987.

[88] K.M. Zuberi and K.G. Shin. Non-Preemptive Scheduling of Messages on Controller Area Network for Real-Time Control Applications. In *Proceedings Real-Time Technology and Applications Symposium*, pages 240–249, May 1995.