Latency, energy, and schedulability of real-time embedded systems

Liu, D.; Liu D.

Chapter 6

Schedulability Analysis of Imprecise Mixed-Criticality Systems

Di Liu, Jelena Spasic, Nan Guan, Gang Chen, Songran Liu, Todor Stefanov, Wang Yi,
"EDF-VD Scheduling of Mixed-Criticality Systems with Degraded Quality Guarantees",
"IEEE International Real-Time Systems Symposium (RTSS'16)", Porto, Portugal, Nov. 29 - Dec. 02, 2016.

As explained in Section 1.1.3, real-time applications with different criticality levels are being implemented on a shared computing platform in order to reduce Size, Weight, and Power (SWaP). We refer to this kind of integrated systems as Mixed-Criticality (MC) systems.

One of the core issues of MC systems stems from the certification authorities (CAs), as explained in Section 1.2 (Problem 3). Vestal in [Ves07] proposes a new model to specify real-time applications in MC systems. The new model captures the core features of MC systems and has received considerable attention since 2007. However, this classical MC model also receives some criticism from system designers [BB13], who complain that the model is too pessimistic in dropping off all low-criticality application tasks when any high-criticality application task overruns. Such an approach seriously disturbs the service of low-criticality tasks and influences the effectiveness of the whole system [BB13][SZ13].

To cope with the criticism and concerns from system designers, Burns and Baruah in [BB13] improve the classical MC model by introducing reduced WCETs for low-criticality tasks. Then, if any high-criticality task overruns its high-criticality WCET, instead of discarding low-criticality tasks, the improved MC model schedules low-criticality tasks with their reduced WCETs. Since the idea of reducing execution bud-

gets to keep tasks running is conceptually similar to the *imprecise computation model* [LLS+91][LSL+94], such MC systems we call *imprecise mixed criticality* (IMC) systems in [LSG+16].

Even though the IMC model is deemed to be a generalization of the classical MC model, it has not received sufficient attention. Only two works investigate the scheduling analysis of the IMC model. In [BB13], Burns and Baruah consider preemptive fixed-priority scheduling for the IMC model and extend the adaptive mixed criticality (AMC) [BBD11] approach to provide a schedulability test for the IMC model. Recently, Barauh *et al* in [BBG16] study the schedulability of the IMC model under Mixed-Criticality fluid scheduling (MC-fluid) [LPG+14].[1] Another widely-studied MC scheduling algorithm, EDF-VD [BBD+12], which has shown strong competence by both theoretical and empirical evaluations on the classical MC model [BBD+12, EY14, Eas13], has not been investigated for the IMC model. Therefore, in this chapter, we analyze the scheduability of the IMC model under EDF-VD scheduling. The novel technical contributions of our work include

- We propose a sufficient test for the IMC model under EDF-VD, - see Theorem 6.3.3 in Section 6.3;
- For the IMC model under EDF-VD, we derive a speedup factor function with respect to the utilization ratios of high criticality tasks and low criticality tasks - see Theorem 6.4.1 in Section 6.4. The derived speedup factor function enables us to quantify the suboptimality of EDF-VD and evaluate the impact of the utilization ratios on the speedup factor. We also compute the maximum value $4/3$ of the speedup factor function, which is equal to the speedup factor bound for the classical MC model [BBD+12].
- With extensive experiments, we show that for the IMC model, by using our proposed sufficient test, in most cases EDF-VD outperforms AMC [BB13] in terms of the number of schedulable task sets. Moreover, the experimental results validate the observations we have obtained for the speedup factor.

6.1 Related Work

Burns and Davis in [BD15] give a comprehensive review of the work on real-time scheduling for MC systems. Many of these works, e.g., [BBD+12] [EY14][Eas13], consider the classical MC model in which all low criticality tasks are discarded if the system switches to the high-criticality mode. In [BB13], Burns and Baruah discuss three approaches to keep some low criticality tasks running in *high*-criticality

[1] This work got public after our RTSS paper [LSG+16] was accepted.

mode. The first approach is to change the priority of low criticality tasks. However, for fixed-priority scheduling, de-prioritizing low criticality tasks cannot guarantee the execution of the low criticality tasks with a short deadline after the mode switches [BB13]. Similarly, for EDF, lowering the priority of low criticality tasks leads to a degraded service [HGST14]. In our work, we consider the IMC model which improves the schedulability of low criticality tasks in *high*-criticality mode by reducing their execution time. The IMC model can guarantee the regular service of a system by trading off the quality of the produced results. For some applications given in [LLS+91][LSL+94][RKKK14b], such trade-off is preferred.

The second approach in [BB13] is to extend the periods of low criticality tasks when the system mode changes to *high*-criticality mode such that the low criticality tasks execute less frequently to ensure their schedulability. Su *et al.* [SZ13][SGZ14] and Jan *et al.* [J+13] both consider this model. However, some applications might prefer an on-time result with a degraded quality rather than a delayed result with a perfect quality. Some example applications can be seen in [CLL90][LLS+91][LSL+94]. Then, the approach of extending periods is less useful for this kind of applications.

The last approach proposed in [BB13] is to reduce the execution budget of low criticality tasks when the system mode switches, i.e., the use of the IMC model studied in this chapter. In [BB13], the authors extend the AMC [BBD11] approach to test the schedulability of an IMC task set under fixed-priority scheduling. Recently, MC-fluid (MCF) scheduling of the IMC model is studied in [BBG16]. In practice, the MCF scheduling suffers from extremely high context switch overhead due to its very fine-grained scheduling units and thus it is difficult to be implemented on a real platform, whereas the EDF-VD scheduling considered in this chapter that is devised based on the the original EDF algorithm does not introduce too much scheduling overhead, thereby allowing to be implemented on a real platform. However, the schedulability problem for an IMC task set under EDF-VD [BBD+12], has not yet been addressed. Therefore, in our work, we study the schedulability of the IMC task model under EDF-VD and propose a sufficient test for it.

6.2 Preliminaries

This section first introduces the IMC task model and its execution semantics. Then, we give a brief explanation to the EDF-VD scheduling [BBD+12] and an example to illustrate the execution semantics of the IMC model under the EDF-VD scheduling.

6.2.1 Imprecise Mixed-Criticality Task Model

We use the *implicit-deadline sporadic* task model given in [BB13] where a task set Γ includes n tasks which are scheduled on a uniprocessor. Without loss of generality,

all tasks in Γ are assumed to start at time 0. Each task τ_i in Γ generates an infinite sequence of jobs $\{J_i^1, J_i^2 ...\}$ and is characterized by $\tau_i = \{T_i, D_i, L_i, \mathcal{C}_i\}$:

- T_i is the period or the minimal separation interval between two consecutive jobs;
- D_i denotes the relative task deadline, where $D_i = T_i$;
- $L_i \in \{LO, HI\}$ denotes the criticality (*low or high*) of a task. In this work, like in many previous research works [SZ13][HGST14][BBD$^+$12] [EY14][Eas13], we consider a dual-criticality MC model. Then, we split tasks into two task sets, $\Gamma_{LO} = \{\tau_i | L_i = LO\}$ and $\Gamma_{HI} = \{\tau_i | L_i = HI\}$;
- $\mathcal{C}_i = \{C_i^{LO}, C_i^{HI}\}$ is a list of WCETs, where C_i^{LO} and C_i^{HI} represent the WCET in *low*-criticality mode and the WCET in *high*-criticality mode, respectively. For a *high*-criticality task, it has $C_i^{LO} \leq C_i^{HI}$, whereas $C_i^{LO} \geq C_i^{HI}$ *for a low-criticality task, i.e., low-criticality task τ_i has a reduced WCET in high-criticality mode*.

Then each job J_i is characterized by $J_i = \{a_i, d_i, L_i, \mathcal{C}_i\}$, where a_i is the absolute release time and d_i is the absolute deadline. Note that if *low*-criticality task τ_i has $C_i^{HI} = 0$, it will be immediately discarded at the time of the switch to *high*-criticality mode. In this case, the IMC model behaves like the classical MC model.

The utilization of a task is used to denote the ratio between its WCET and its period. We define the following utilizations for an IMC task set Γ:

- For every task τ_i, it has $u_i^{LO} = \frac{C_i^{LO}}{T_i}$, $u_i^{HI} = \frac{C_i^{HI}}{T_i}$;
- For all *low*-criticality tasks, we have total utilizations

$$U_{LO}^{LO} = \sum_{\forall \tau_i \in \Gamma_{LO}} u_i^{LO}, \quad U_{LO}^{HI} = \sum_{\forall \tau_i \in \Gamma_{LO}} u_i^{HI}$$

- For all *high*-criticality tasks, we have total utilizations

$$U_{HI}^{LO} = \sum_{\forall \tau_i \in \Gamma_{HI}} u_i^{LO}, \quad U_{HI}^{HI} = \sum_{\forall \tau_i \in \Gamma_{HI}} u_i^{HI}$$

- For an IMC task set, we have

$$U^{LO} = U_{LO}^{LO} + U_{HI}^{LO}, \quad U^{HI} = U_{LO}^{HI} + U_{HI}^{HI}$$

6.2.2 Execution Semantics of the IMC Model

The execution semantics of the IMC model are similar to those of the classical MC model. The **major difference** occurs after a system switches to *high*-criticality mode. *Instead of discarding all low-criticality tasks, as it is done in the classical MC model, the IMC model tries to schedule low-criticality tasks with their reduced execution times* C_i^{HI}. The execution semantics of the IMC model are summarized as follows:

- The system starts in *low*-criticality mode, and remains in this mode as long as no *high*-criticality job overruns its *low*-criticality WCET C_i^{LO}. If any job of a *low*-criticality task tries to execute beyond its C_i^{LO}, the system will suspend it and launch a new job at the next period;
- If any job of *high*-criticality task executes for its C_i^{LO} time units without signaling completion, the system immediately switches to *high*-criticality mode;
- As the system switches to *high*-criticality mode, *if jobs of low-criticality tasks have completed execution for more than their* C_i^{HI} *but less than their* C_i^{LO}*, the jobs will be suspended till the tasks release new jobs for the next period. However, if jobs of low-criticality tasks have not completed their* C_i^{HI} *(*$\leq C_i^{LO}$*) by the switch time instant, the jobs will complete the left execution to* C_i^{HI} *after the switch time instant and before their deadlines. Hereafter, all jobs are scheduled using* C_i^{HI}. For *high*-criticality tasks, if their jobs have not completed their C_i^{LO} ($\leq C_i^{HI}$) by the switch time instant, all jobs will continue to be scheduled to complete C_i^{HI}. After that, all jobs are scheduled using C_i^{HI}.

Santy *et al.* [SGTG12] have shown that the system can switch back from the *high*-criticality mode to the *low*-criticality mode when there is an idle period and no *high*-criticality job awaits for execution. For the IMC model, we can use the same scenario to trigger the switch-back. In this work, we focus on the switch from *low*-criticality mode to *high*-criticality mode.

6.2.3 EDF-VD Scheduling

The challenge to schedule MC tasks with the EDF scheduling algorithm [LL73] is to deal with the overrun of *high*-criticality tasks when the system switches from *low*-criticality mode to *high*-criticality mode. Baruah *et al.* proposed in [BBD+12] to artificially tighten deadlines of jobs of *high*-criticality tasks in *low*-criticality mode such that the system can preserve execution budgets for the *high*-criticality tasks across mode switches. This approach is called *EDF with virtual deadlines* (EDF-VD).

Task	L	C_i^{LO}	C_i^{HI}	T_i	$\hat{D}_i$
τ_1	LO	3	2	9	
τ_2	HI	4	8	10	7

Table 6.1: Illustrative example

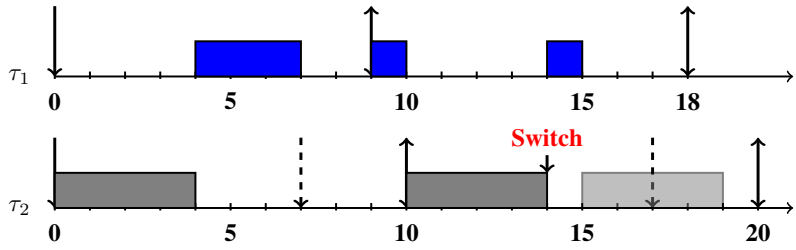

Figure 6.1: Scheduling of Example 6.1

6.2.4 An Illustrative Example

Here, we give a simple example to illustrate the execution semantics of the IMC model under EDF-VD. Table 6.1 gives two tasks, one *low*-criticality task τ_1 and one *high*-criticality task τ_2, where $\hat{D}_i$ is the virtual deadline. Figure 6.1 depicts the scheduling of the given IMC task set, where we assume that the mode switch occurs in the second period of τ_2. When the system switches to *high*-criticality mode, τ_2 will be scheduled by its original deadline 10 instead of its virtual deadline 7. Hence, τ_1 preempts τ_2 at the switch time instant. Since in *high*-criticality mode τ_1 only has execution budget of 2 , i.e., C_1^{HI}, τ_1 executes one unit and suspends. Then, τ_2 completes its left execution 4 ($C_2^{HI} - C_2^{LO}$) before its deadline.

6.3 Schedulability Analysis

In this section, we analyze the scheduability of the IMC model under EDF-VD scheduling and propose the first sufficient scheduability test. To ensure the timing correctness of the IMC model, we need to guarantee the scheduability for both *high*-criticality and *low*-criticality modes. Following, we demonstrate our analysis procedure and the formal theoretical proof.

6.3.1 Low Criticality Mode

We first ensure the schedulability of tasks when they are in *low*-criticality mode. As the task model is in *low*-criticality mode, the tasks can be considered as traditional

real-time tasks scheduled by EDF algorithm with virtual deadlines (VD). The following theorem is given in [BBD+12] for tasks scheduled in *low*-criticality mode.

Theorem 6.3.1 (Theorem 1 from [BBD+12])**.** *The following condition is sufficient for ensuring that EDF-VD successfully schedules all tasks in low-criticality mode:*

$$1 \geq \frac{U_{HI}^{LO}}{x} + U_{LO}^{LO} \tag{6.1}$$

where $x \in (0, 1)$ is used to uniformly modify the relative deadline of high-criticality tasks.

Since the IMC model behaves as the classical MC model in *low*-criticality mode, Theorem 6.3.1 holds for the IMC model as well.

6.3.2 High Criticality Mode

For *high*-criticality mode, the classical MC model discards all *low*-criticality jobs after the switch to *high*-criticality mode. In contrast, the IMC model keeps *low*-criticality jobs running but with degraded quality, i.e., a shorter execution time. So the schedulability condition in [BBD+12] does not work for the IMC model in the *high*-criticality mode. Thus, we need a new test for the IMC model in *high*-criticality mode.

To derive the sufficient test in *high*-criticality mode, suppose that there is a time interval $[0, t_2]$, where a first deadline miss occurs at t_2 and t_1 denotes the time instant of the switch to *high*-criticality mode in the time interval, where $t_1 < t_2$. Assume that $\mathcal{J}$ is the minimal set of jobs generated from task set Γ which leads to the first deadline miss at t_2. The minimality of $\mathcal{J}$ means that removing any job in $\mathcal{J}$ guarantees the schedulability of the rest of $\mathcal{J}$. Here, we introduce some notations for our later interpretation. Let variable η_i denote the cumulative execution time of task τ_i in the interval $[0, t_2]$. J_1 denotes a special *high*-criticality job which has switch time instant t_1 within its period (a_1, d_1), i.e, $a_1 < t_1 < d_1$. Furthermore, J_1 is the job with the earliest release time amongst all *high*-criticality jobs in $\mathcal{J}$ which execute in $[t_1, t_2)$. Moreover, we define a special type of job for *low*-criticality tasks which is useful for our later proofs.

Definition 6.3.1. A job J_i from *low*-criticality task τ_i is a carry-over job, if its absolute release time a_i is before and its absolute deadline d_i is after the switch time instant, i.e., $a_i < t_1 < d_i$.

With the notations introduced above, we have the following propositions,

Proposition 4 (Fact 1 from [BBD+12])**.** *All jobs in $\mathcal{J}$ that execute in $[t_1, t_2)$ have deadline $\leq t_2$.*

It is easy to observe that only jobs which have deadlines $\leq t_2$ are possible to cause a deadline miss at t_2. If a job has its deadline $> t_2$ and is still in set $\mathcal{J}$, it will contradict the minimality of $\mathcal{J}$.

Proposition 5. *The switch time instant t_1 has*

$$t_1 < (a_1 + x(t_2 - a_1)) \tag{6.2}$$

Proof. Let us consider a time instant $(a_1+x(d_1-a_1))$ which is the virtual deadline of job J_1. Since J_1 executes in time interval $[t_1, t_2)$, its virtual deadline $(a_1+x(d_1-a_1))$ must be greater than the switch time instant t_1. Otherwise, it should have completed its *low*-criticality execution before t_1, and this contradicts that it executes in $[t_1, t_2)$. Thus, it has

$$\begin{aligned} & t_1 < (a_1 + x(d_1 - a_1)) \\ \Rightarrow & t_1 < (a_1 + x(t_2 - a_1)) \quad (\text{since } d_1 \leq t_2) \end{aligned}$$

□

Proposition 6. *If a carry-over job J_i has its cumulative execution equal to $(d_i - a_i)u_i^{LO}$ and $u_i^{LO} > u_i^{HI}$, its deadline d_i is $\leq (a_1 + x(t_2 - a_1))$.*

Proof. For a carry-over job J_i, if it has its cumulative execution equal to $(d_i-a_i)u_i^{LO}$ and $u_i^{LO} > u_i^{HI}$, it should complete its C_i^{LO} execution before t_1. Otherwise, if job J_i has executed time units $C_i \in [C_i^{HI}, C_i^{LO})$ at time instant t_1, it will be suspended and will not execute after t_1.

Now, we will show that when job J_i completes its C_i^{LO} execution, its deadline is $d_i \leq (a_1 + x(t_2 - a_1))$. We prove this by contradiction. First, we suppose that J_i has its deadline $d_i > (a_1 + x(t_2 - a_1))$ and release time a_i. As shown above, job J_i completes its C_i^{LO} execution before t_1. Let us assume a time instant t^* as the latest time instant at which this carry-over job J_i starts to execute before t_1. This means that at this time instant all jobs in $\mathcal{J}$ with deadline $\leq (a_1 + x(t_2 - a_1))$ have finished their executions. This indicates that these jobs will not have any execution within interval $[t^*, t_2]$. Therefore, jobs in $\mathcal{J}$ with release time at or after time instant t^* can form a smaller job set which causes a deadline miss at t_2. Then, it contradicts the minimality of $\mathcal{J}$. Thus, carry-over job J_i with its cumulative execution time equal to $(d_i-a_i)u_i^{LO}$ and $u_i^{LO} > u_i^{HI}$ has its deadline $d_i \leq (a_1 + x(t_2 - a_1))$. □

With the propositions and notations given above, we derive an upper bound of the cumulative execution time η_i of *low*-criticality task τ_i.

Lemma 6.3.1. *For any low-criticality task τ_i, it has*

$$\eta_i \leq (a_1 + x(t_2 - a_1))u_i^{LO} + (1 - x)(t_2 - a_1)u_i^{HI} \tag{6.3}$$

Proof. If $u_i^{LO} = u_i^{HI}$, it is trivial to see that Lemma 6.3.1 holds. Below we focus on the case when $u_i^{LO} > u_i^{HI}$. If a system switches to *high*-criticality mode at t_1, then we know that *low*-criticality tasks are scheduled using C_i^{LO} before t_1 and using C_i^{HI} after t_1. To prove this lemma, we need to consider two cases, where τ_i releases a job within interval $(a_1, t_2]$ or it does not. We prove the two cases separately.

Case A (task τ_i releases a job within interval $(a_1, t_2]$): There are two sub-cases to be considered.

- **Sub-case 1 (No carry-over job):** The deadline of a job of *low*-criticality task τ_i coincides with switch time instant t_1. The cumulative execution time of *low*-criticality task τ_i within time interval $[0, t_2]$ can be bounded as follows,

$$\eta_i \leq (t_1 - 0) \cdot u_i^{LO} + (t_2 - t_1) \cdot u_i^{HI}$$

  Since $t_1 < (a_1 + x(t_2 - a_1))$ according to Proposition 5 and for *low*-criticality task τ_i it has $u_i^{LO} > u_i^{HI}$, then

$$\begin{aligned} &\eta_i < \big(a_1 + x(t_2 - a_1)\big)u_i^{LO} + \big(t_2 - \big(a_1 + x(t_2 - a_1)\big)\big)u_i^{HI} \\ \Leftrightarrow &\eta_i < (a_1 + x(t_2 - a_1))u_i^{LO} + (1 - x)(t_2 - a_1)u_i^{HI} \end{aligned}$$

- **Sub-case 2 (with carry-over job):** In this case, before the carry-over job, jobs of τ_i are scheduled with its C_i^{LO}. After the carry-over job, jobs of τ_i are scheduled with its C_i^{HI}. It is trivial to observe that for a carry-over job its maximum cumulative execution time can be obtained when it completes its C_i^{LO} within its period $[a_i, d_i]$, i.e., $(d_i - a_i)u_i^{LO}$. Considering the maximum cumulative execution for the carry-over job, we then have for *low*-criticality task τ_i,

$$\begin{aligned} &\eta_i \leq (a_i - 0)u_i^{LO} + (d_i - a_i)u_i^{LO} + (t_2 - d_i)u_i^{HI} \\ \Leftrightarrow &\eta_i \leq d_i u_i^{LO} + (t_2 - d_i)u_i^{HI} \end{aligned}$$

  Proposition 6 shows that as J_i has its cumulative execution equal to $(d_i - a_i) \cdot u_i^{LO}$, it has $d_i \leq (a_1 + x(t_2 - a_1))$. Given that $u_i^{LO} > u_i^{HI}$ for *low*-criticality task, we have

$$\begin{aligned} &\eta_i \leq d_i u_i^{LO} + (t_2 - d_i)u_i^{HI} \\ \Rightarrow &\eta_i \leq \big(a_1 + x(t_2 - a_1)\big)u_i^{LO} + \big(t_2 - \big(a_1 + x(t_2 - a_1)\big)\big)u_i^{HI} \\ \Leftrightarrow &\eta_i \leq (a_1 + x(t_2 - a_1))u_i^{LO} + (1 - x)(t_2 - a_1)u_i^{HI} \end{aligned}$$

Case B (task τ_i does not release a job within interval $(a_1, t_2]$): In this case, let J_i denote the last release job of task τ_i before a_1 and a_i and d_i are its absolute release time and absolute deadline, respectively. If $d_i \leq t_1$, we have

$$\eta_i = (a_i - 0)u_i^{LO} + (d_i - a_i) \cdot u_i^{LO} = d_i u_i^{LO}$$

If $d_i > t_1$, J_i is a carry-over job. As we discussed above, the maximum cumulative execution time of carry-over job J_i is $(d_i - a_i)u_i^{LO}$, so we have

$$\eta_i \leq (a_i - 0)u_i^{LO} + (d_i - a_i) \cdot u_i^{LO} \Leftrightarrow \eta_i \leq d_i u_i^{LO}$$

Similarly, according to Proposition 6, we obtain,

$$\begin{aligned} &\eta_i \leq d_i \cdot u_i^{LO} \leq (a_1 + x(t_2 - a_1))u_i^{LO} \\ \Rightarrow &\eta_i < (a_1 + x(t_2 - a_1))u_i^{LO} + \big(t_2 - \big(a_1 + x(t_2 - a_1)\big)\big)u_i^{HI} \\ \Leftrightarrow &\eta_i < (a_1 + x(t_2 - a_1))u_i^{LO} + (1 - x)(t_2 - a_1)u_i^{HI} \end{aligned}$$

□

Lemma 6.3.1 gives the upper bound of the cumulative execution time of a *low*-criticality task in *high*-criticality mode. In order to derive the sufficient test for the IMC model in *high*-criticality mode, we need to upper bound the cumulative execution time of *high*-criticality tasks.

Proposition 7 (Fact 3 from [BBD+12])**.** *For any high-criticality task τ_i, it holds that*

$$\eta_i \leq \frac{a_1}{x}u_i^{LO} + (t_2 - a_1)u_i^{HI} \tag{6.4}$$

Proposition 7 is used to bound the cumulative execution of the *high*-criticality tasks. Since in the IMC model the *high*-criticality tasks are scheduled as in the classical MC model, Proposition 7 holds for the IMC model as well. With Lemma 6.3.1 and Proposition 7, we can derive the sufficient test for the IMC model in *high*-criticality mode.

Theorem 6.3.2. *The following condition is sufficient for ensuring that EDF-VD successfully schedules all tasks in high-criticality mode:*

$$xU_{LO}^{LO} + (1 - x)U_{LO}^{HI} + U_{HI}^{HI} \leq 1 \tag{6.5}$$

Proof. Let N denote the cumulative execution time of all tasks in $\Gamma = \Gamma_{LO} \cup \Gamma_{HI}$ over interval $[0, t_2]$. We have

$$N = \sum_{\forall \tau_i \in \Gamma_{LO}} \eta_i + \sum_{\forall \tau_i \in \Gamma_{HI}} \eta_i$$

By using Lemma 6.3.1 and Proposition 7, N is bounded as follows

$$\begin{aligned}
N \leq & \sum_{\forall \tau_i \in \Gamma_{LO}} \left(\big(a_1 + x(t_2 - a_1)\big) u_i^{LO} + (1-x)(t_2 - a_1) u_i^{HI} \right) \\
& + \sum_{\forall \tau_i \in \Gamma_{HI}} \left(\frac{a_1}{x} u_i^{LO} + (t_2 - a_1) u_i^{HI} \right) \\
\Leftrightarrow N \leq & (a_1 + x(t_2 - a_1)) U_{LO}^{LO} + (1-x)(t_2 - a_1) U_{LO}^{HI} \\
& + \frac{a_1}{x} U_{HI}^{LO} + (t_2 - a_1) U_{HI}^{HI} \\
\Leftrightarrow N \leq & a_1 (U_{LO}^{LO} + \frac{U_{HI}^{LO}}{x}) + x(t_2 - a_1) U_{LO}^{LO} \\
& + (1-x)(t_2 - a_1) U_{LO}^{HI} + (t_2 - a_1) U_{HI}^{HI}
\end{aligned} \tag{6.6}$$

Since the tasks must be schedulable in *low*-criticality mode, the condition given in Theorem 6.3.1 holds and we have $1 \geq (U_{LO}^{LO} + \frac{U_{HI}^{LO}}{x})$. Hence,

$$\begin{aligned}
N \leq & a_1 + x(t_2 - a_1) U_{LO}^{LO} \\
& + (1-x)(t_2 - a_1) U_{LO}^{HI} + (t_2 - a_1) U_{HI}^{HI}
\end{aligned} \tag{6.7}$$

Since time instant t_2 is the first deadline miss, it means that there is no idle time instant within interval $[0, t_2]$. Note that if there is an idle instant, jobs from set $\mathcal{J}$ which have release time at or after the latest idle instant can form a smaller job set causing deadline miss at t_2 which contradicts the minimality of $\mathcal{J}$. Then, we obtain

$$\begin{aligned}
& N = \left(\sum_{\forall \tau_i \in \Gamma_{LO}} \eta_i + \sum_{\forall \tau_i \in \Gamma_{HI}} \eta_i \right) > t_2 \\
\Rightarrow & a_1 + x(t_2 - a_1) U_{LO}^{LO} + (1-x)(t_2 - a_1) U_{LO}^{HI} + (t_2 - a_1) U_{HI}^{HI} \\
& > t_2 \\
\Leftrightarrow & x(t_2 - a_1) U_{LO}^{LO} + (1-x)(t_2 - a_1) U_{LO}^{HI} + (t_2 - a_1) U_{HI}^{HI} \\
& > t_2 - a_1 \\
\Leftrightarrow & x U_{LO}^{LO} + (1-x) U_{LO}^{HI} + U_{HI}^{HI} > 1
\end{aligned}$$

By taking the contrapositive, we derive the sufficient test for the IMC model when it is in *high*-criticality mode:

$$x U_{LO}^{LO} + (1-x) U_{LO}^{HI} + U_{HI}^{HI} \leq 1$$

□

Note that if $U_{LO}^{HI} = 0$, i.e., no *low*-criticality tasks are scheduled after the system switches to *high*-criticality mode, our Theorem 6.3.2 is the same as the sufficient

test (Theorem 2 in [BBD+12]) for the classical MC model in *high*-criticality mode. Hence, our Theorem 6.3.2 actually is a generalized schedulability condition for (I)MC tasks under EDF-VD.

By combining Theorem 6.3.1 (see Section 6.3.1) and our Theorem 6.3.2, we prove the following theorem,

Theorem 6.3.3. *Given an IMC task set, if*

$$U_{HI}^{HI} + U_{LO}^{LO} \leq 1 \tag{6.8}$$

then the IMC task set is schedulable by EDF; otherwise, if

$$\frac{U_{HI}^{LO}}{1 - U_{LO}^{LO}} \leq \frac{1 - (U_{HI}^{HI} + U_{LO}^{HI})}{U_{LO}^{LO} - U_{LO}^{HI}} \tag{6.9}$$

where

$$U_{HI}^{HI} + U_{LO}^{HI} < 1 \text{ and } U_{LO}^{LO} < 1 \text{ and } U_{LO}^{LO} > U_{LO}^{HI} \tag{6.10}$$

then this IMC task set can be scheduled by EDF-VD with a deadline scaling factor x arbitrarily chosen in the following range

$$x \in \left[\frac{U_{HI}^{LO}}{1 - U_{LO}^{LO}}, \ \frac{1 - (U_{HI}^{HI} + U_{LO}^{HI})}{U_{LO}^{LO} - U_{LO}^{HI}} \right]$$

Proof. Total utilization $U \leq 1$ is the exact test for EDF on a uniprocessor system. If the condition in (6.8) is met, the given task set is *worst-case reservation* [BBD+12] schedulable under EDF, i.e., the task set can be scheduled by EDF without deadline scaling for *high*-criticality tasks and execution budget reduction for *low*-criticality tasks. Now, we prove the second condition given by (6.9). From Theorem 6.3.1, we have,

$$x \geq \frac{U_{HI}^{LO}}{1 - U_{LO}^{LO}}$$

From Theorem 6.3.2, we have

$$\begin{aligned} & xU_{LO}^{LO} + (1 - x)U_{LO}^{HI} + U_{HI}^{HI} \leq 1 \\ \Leftrightarrow & x \leq \frac{1 - (U_{HI}^{HI} + U_{LO}^{HI})}{U_{LO}^{LO} - U_{LO}^{HI}} \end{aligned}$$

Therefore, if $\frac{U_{HI}^{LO}}{1-U_{LO}^{LO}} \leq \frac{1-(U_{HI}^{HI}+U_{LO}^{HI})}{U_{LO}^{LO}-U_{LO}^{HI}}$, the schedulability conditions of both Theorem 6.3.1 and 6.3.2 are satisfied. Thus, the IMC tasks are schedulable under EDF-VD. □

6.4 Speedup Factor

The speedup factor bound is a useful metric to compare the worst-case performance of different MC scheduling algorithms. The following is the definition of the speedup factor for an MC scheduling algorithm.

Definition 6.4.1 (from [BBD+12])**.** The *speedup factor* of an algorithm $\mathcal{A}$ for scheduling MC systems is the smallest real number $f \geq 1$ such that any task system that is schedulable by a hypothetical optimal clairvoyant scheduling algorithm[2] on a unit-speed processor is correctly scheduled by algorithm $\mathcal{A}$ on a speed-f processor.

Informally speaking, by increasing the processor's speed, a non-optimal scheduling algorithm is able to schedule the task sets which are deemed to be unschedulable by the non-optimal scheduling algorithm but schedulable by an optimal scheduling algorithm on the processor without speed increase. The speedup factor actually computes how much the processor needs to speed up such that the non-optimal scheduling algorithm achieves the same scheduling performance as an optimal scheduling algorithm. The smaller speedup factor indicates a better scheduling performance for the non-optimal scheduling algorithm. The speedup factor bound for the classical MC model under EDF-VD [BBD+12] has been shown to be $4/3$.

In the following, we prove the speedup factor of the IMC model under EDF-VD scheduling. For notational simplicity, we define

$$U_{HI}^{HI} = c, \quad U_{HI}^{LO} = \alpha \times c$$
$$U_{LO}^{LO} = b, \quad U_{LO}^{HI} = \lambda \times b$$

where $\alpha \in (0, 1]$ and $\lambda \in [0, 1]$. α denotes the utilization ratio between U_{HI}^{LO} and U_{HI}^{HI}, while λ denotes the utilization ratio between U_{LO}^{HI} and U_{LO}^{LO}.

First, let us analyze the speedup factor of two corner cases. When $\alpha = 1$, i.e., $U_{HI}^{LO} = U_{HI}^{HI}$, this means that there is no mode-switch. Therefore, the task set is scheduled by the traditional EDF, i.e., the task set is schedulable if and only if $U_{LO}^{LO} + U_{HI}^{LO} \leq 1$. Since EDF is the optimal scheduling algorithm on a uniprocessor system, the speedup factor thus is 1. When $\lambda = 1$, i.e., $U_{LO}^{LO} = U_{LO}^{HI}$, if the task set is schedulable in *high*-criticality mode, it must hold $U_{HI}^{HI} + U_{LO}^{LO} \leq 1$ by Theorem 6.3.2. Then it is scheduled by the traditional EDF and thus the speedup factor is 1 as well.

In our work, instead of generating a single speedup factor bound, we derive a speedup factor function with respect to (α, λ). This speedup factor function enables

[2] A 'clairvoyant' scheduling algorithm knows all run-time information, e.g., when the mode switch will occur, prior to run-time.

us to quantify the suboptimality of EDF-VD for the IMC model in terms of speedup factor (by our proposed sufficient test) and to evaluate the impact of the utilization ratio on the schedulability of an IMC task set under EDF-VD.

First, we strive to find a minimum speed s ($\leq$1) for a clairvoyant optimal MC scheduling algorithm such that any implicit-deadline IMC task set which is schedulable by the clairvoyant optimal MC scheduling algorithm on a speed-s processor can satisfy the schedulability test given in Theorem 6.3.3, i.e., schedulable under EDF-VD on a unit-speed processor. Then, we can compute the speed-up factor by simply computing $1/s$.

Lemma 6.4.1. *Given* $b, c \in [0, 1]$, $\alpha \in (0, 1)$, $\lambda \in [0, 1)$, *and*

$$\max\{b + \alpha c, \lambda b + c\} \leq S(\alpha, \lambda) \tag{6.11}$$

where

$$S(\alpha, \lambda) = \frac{(1 - \alpha\lambda)((2 - \alpha\lambda - \alpha) + (\lambda - 1)\sqrt{4\alpha - 3\alpha^2})}{2(1 - \alpha)(\alpha\lambda - \alpha\lambda^2 - \alpha + 1)}$$

then it guarantees

$$\frac{\alpha c}{1 - b} \leq \frac{1 - (c + \lambda b)}{b - \lambda b} \tag{6.12}$$

Proof. Suppose that λ and α are constants and we have a real number $s \leq 1$, where $\max\{b + \alpha c, \lambda b + c\} \leq s$. We need to find the minimum of s which guarantees that *any* $b, c \in [0, 1]$ ensure (6.12). First, $\max\{b + \alpha c, \lambda b + c\} \leq s$ implies

$$b + \alpha c \leq s \tag{6.13}$$

$$\lambda b + c \leq s \tag{6.14}$$

Then, condition (6.12) can be written as follows,

$$\lambda b^2 + (\alpha\lambda - \alpha + 1)bc - (\lambda + 1)b - c + 1 \geq 0 \tag{6.15}$$

Inequalities (6.13)(6.14)(6.15) define a feasible space in the three-dimension space, respectively. In Figure 6.2, the space above the plane is a feasible space satisfying (6.13), where the plane corresponds to $b + \alpha c = s$. For (6.14), $\lambda b + c = s$ draws a plane and the feasible space is above the plane shown in Figure 6.3. Similarly, when (6.15) makes its right-hand-side equal to the left-hand-side, we draw a vertical curved surface seen in Figure 6.4 and the space inside the vertical surface is the feasible space (the opposite of the arrow direction). We need to find the *minimum* of s in the feasible space (above the two planes and inside the vertical surface) such that **any** b and c that meet (6.11) satisfy (6.12). Since $\max\{b + \alpha c, \lambda b + c\} = s$ is strictly increasing, to ensure that condition (6.12) hold for *any* b and c, we strive to minimize

$\max\{b+\alpha c, \lambda b+c\}$ in the feasible space. Then, this problem can be transformed into another form, where, instead of minimizing $\max\{b + \alpha c, \lambda b + c\}$ inside the vertical surface, we minimize the value of $\max\{b+\alpha c, \lambda b+c\}$ in the space outside the vertical surface[3] which is defined by

$$\lambda b^2 + (\alpha\lambda - \alpha + 1)bc - (\lambda + 1)b - c + 1 \leq 0 \tag{6.16}$$

This is equivalent to the minimization of s with the above constraint. Then, the minimization problem is formulated as follows,

$$\begin{aligned}
&\text{minimize} && s && (6.17)\\
&\text{subject to} && b + \alpha c \leq s && (6.18)\\
& && \lambda b + c \leq s && (6.19)\\
& && \lambda b^2 + (\alpha\lambda - \alpha + 1)bc - (\lambda + 1)b - c + 1 \leq 0 && (6.20)\\
& && 0 \leq b \leq 1, \quad 0 \leq c \leq 1 && (6.21)
\end{aligned}$$

where α and λ are constant and s, b, c are variables. If $S(\alpha, \lambda)$ is the optimal solution of the optimization problem (6.17), then Lemma 6.4.1 is proven.

Below, we prove that $S(\alpha, \lambda)$ is the optimal solution of the optimization problem (6.17)[4]

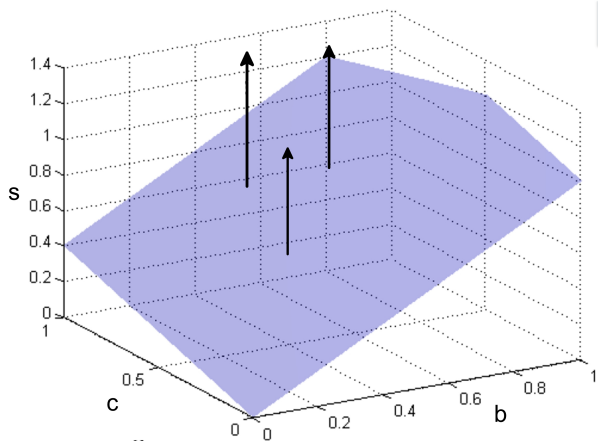

Figure 6.2: plane 1

[3] As the arrows direct

[4] This optimization problem is a non-convex problem and thus we cannot use general convex optimization techniques such as the Karush-Kuhn-Tucker (KKT) approach [KT51] to solve it.

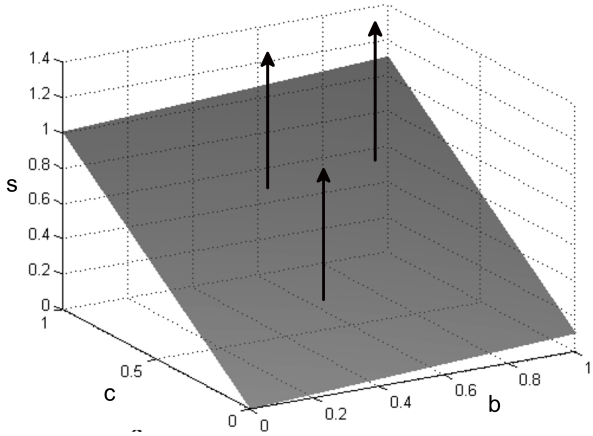

Figure 6.3: plane 2

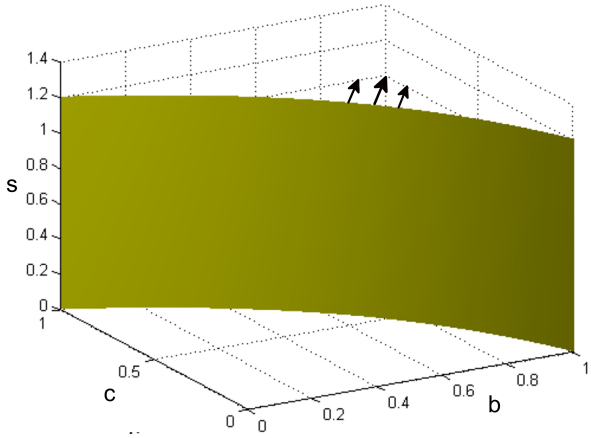

Figure 6.4: vertical surface

As stated before, the feasible solutions subject to these three constraints (6.18), (6.19) and (6.20) must be above both planes and outside the vertical curved surface. First assume that we have a point (b'_0, c'_0, s'_0) which satisfies all constraints but is not on the vertical surface. If we connect the origin $(0, 0, 0)$ and (b'_0, c'_0, s'_0), this line must have an intersection point (b^*_0, c^*_0, s^*_0) with the vertical surface. It is easy to observe that $s^*_0 < s'_0$ - see in Figure 6.5. This means that any point which is not on the vertical surface can find a point with smaller value of s on the vertical surface which satisfies all constraints. Therefore, the point with the minimum s must be on the vertical surface. Similarly, the minimum s must be on one of the two planes.

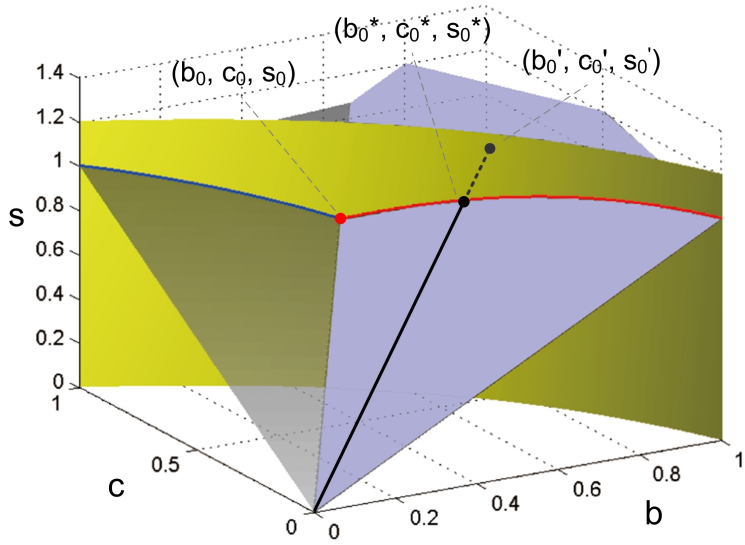

Figure 6.5: 3D space of optimization problem (6.17)

Otherwise, if it is not on any plane, we always can find a projected point on one plane which has a smaller value of s.

We have shown above that to obtain the minimum value of s the point must be on the vertical surface and one plane. Then, the two planes have an intersection line and this line intersects with the vertical surface at a point denoted by (b_0, c_0, s_0). By taking constraints (6.18)(6.19) and (6.20), we formulate a piece-wise function of s with respect to b as follows.

$$s(b) = \begin{cases} \frac{(\alpha\lambda^2-\alpha\lambda)b^2+b-1}{(\alpha\lambda-\alpha+1)b-1} & 0 < b \leq b_0 \\ \frac{(1-\alpha)b^2+(\alpha\lambda+\alpha-1)b-\alpha}{(\alpha\lambda-\alpha+1)b-1} & b_0 < b \leq 1 \end{cases} \tag{6.22}$$

This function covers all points which are on the vertical surface and one plane and at same time satisfy all constraints. By doing some calculus, we know that Equation (6.22) is monotonically decreasing in $(0, b_0]$ and monotonically increasing in $(b_0, 1]$. Therefore, the minimum value of Equation (6.22) can be obtained at (b_0, c_0, s_0). The complete proof is given by Lemma 1 in Appendix I. It means that we can obtain the optimal solution of problem (6.17) by solving the following system of equations.

$$\begin{cases} b_0 + \alpha c_0 = s_0 \\ \lambda b_0 + c_0 = s_0 \\ \lambda b_0^2+(\alpha\lambda-\alpha+1)b_0c_0-(\lambda+1)b_0-c_0+1 = 0 \end{cases} \tag{6.23}$$

By joining the first two equations we have $c_0 = \frac{1-\lambda}{1-\alpha} \times b_0$, and applying it to the last equation in (6.23) gives

$$(-\alpha\lambda^2 + \alpha\lambda - \alpha + 1)b_0^2 + (\alpha\lambda + \alpha - 2)b_0 + (1 - \alpha) = 0$$

By the well-known Quadratic Formula we get the two roots of the above quadratic equation.

$$b_0^1 = \frac{(2 - \alpha\lambda - \alpha) + (1 - \lambda)\sqrt{-3\alpha^2 + 4\alpha}}{2(-\alpha\lambda^2 + \alpha\lambda - \alpha + 1)} \tag{6.24}$$

$$b_0^2 = \frac{(2 - \alpha\lambda - \alpha) - (1 - \lambda)\sqrt{-3\alpha^2 + 4\alpha}}{2(-\alpha\lambda^2 + \alpha\lambda - \alpha + 1)} \tag{6.25}$$

We can prove that b_0^2 is larger than 1 and thus should be dropped (since we require $0 \leq b \leq 1$), while b_0^1 is in the range of $[0, 1]$. The detailed proof is given by Lemma 2 in Appendix I. As a result, we obtain the optimal solution $(b_0^1, \frac{1-\lambda}{1-\alpha}b_0^1, \frac{1-\alpha\lambda}{1-\alpha}b_0^1)$ for Equation (6.23). Thus, we have

$$\begin{aligned} S(\alpha, \lambda) &= \frac{1 - \alpha\lambda}{1 - \alpha} b_0^1 \\ &= \frac{(1 - \alpha\lambda)((2 - \alpha\lambda - \alpha) + (\lambda - 1)\sqrt{4\alpha - 3\alpha^2})}{2(1 - \alpha)(\alpha\lambda - \alpha\lambda^2 - \alpha + 1)} \end{aligned}$$

Therefore, Lemma 6.4.1 is proven. □

Lemma 6.4.1 shows that any IMC task set that is schedulable by an optimal clairvoyant MC scheduling algorithm on a speed-$S(\alpha, \lambda)$ is schedulable by EDF-VD on a unit-speed processor. Therefore, we can compute the speedup factor of EDF-VD by $1/S(\alpha, \lambda)$.

Theorem 6.4.1. *The speedup factor of EDF-VD with IMC task sets is*

$$f = \frac{2(1 - \alpha)(\alpha\lambda - \alpha\lambda^2 - \alpha + 1)}{(1 - \alpha\lambda)((2 - \alpha\lambda - \alpha) + (\lambda - 1)\sqrt{4\alpha - 3\alpha^2})}$$

Proof. Follow the explanation given above. □

The speedup factor is shown to be a function of α and λ. Figure 6.6 plots the 3D image of this function and Table 6.2 lists some of the values with different α and λ. By doing some calculus, we obtain the maximum value 1.333 $(4/3)$ of the speedup factor function when $\lambda = 0$ and $\alpha = \frac{1}{3}$, which is highlighted in Figure 6.6 and Table

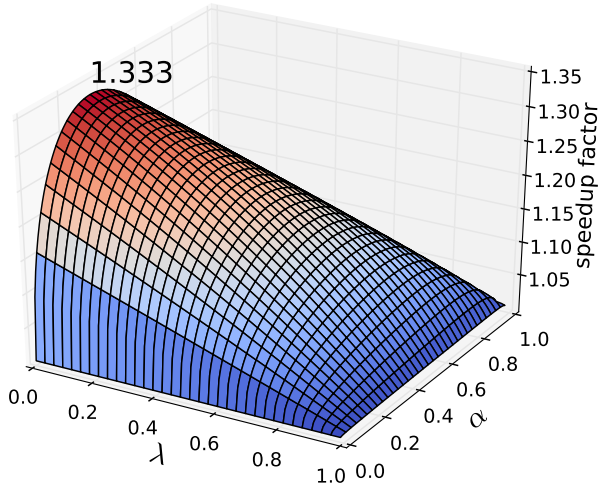

Figure 6.6: 3D image of the speedup factor w.r.t α and λ

λ \ α	0.1	0.3	1/3	0.5	0.7	0.9	1
0	1.254	1.332	**1.333**	1.309	1.227	1.091	1
0.1	1.231	1.308	1.310	1.293	1.219	1.090	1
0.3	1.183	1.256	1.259	1.254	1.201	1.087	1
0.5	1.134	1.195	1.200	1.206	1.174	1.083	1
0.7	1.082	1.126	1.130	1.143	1.133	1.074	1
0.9	1.028	1.046	1.048	1.056	1.061	1.048	1
1	1	1	1	1	1	1	1

Table 6.2: The speedup factor w.r.t α and λ

6.2. We see that the speedup factor bound is achieved when the task set is a classical MC task set. From Figure 6.6 and Table 6.2, we observe different trends for the speedup factor with respect to α and λ.

- First, given a fixed λ, the speedup factor is not a monotonic function with respect to α. The relation between α and the speedup factor draws a downward parabola. Therefore, a straightforward conclusion regarding the impact of α on the speedup factor cannot be drawn.

- Given a fixed α, the speedup factor is a monotonically decreasing function with respect to increasing λ. It is seen that increasing λ leads to a smaller value of the speedup factor. *This means that a larger λ brings a positive effect on the schedulability of an IMC task set*.

6.5 Experimental Evaluation

In this section, we conduct experiments to evaluate the effectiveness of the proposed sufficient test for the IMC model in terms of schedulable task sets (acceptance ratio). Moreover, we conduct experiments to verify the two observations stated at the end of Section 6.4 regarding the impact of α and λ on the average acceptance ratio. Our experiments are based on randomly generated MC tasks. We use a task generation approach, similar to that used in [Eas13][EY14], to randomly generate IMC task sets to evaluate the proposed sufficient test. Each task τ_i is generated based on the following procedure,

- pCriticality is the probability that the generated task is a *high*-criticality task; pCriticality$\in [0, 1]$.

- Period T_i is randomly selected from the range $[100, 1000]$.

- In order to have sufficient number of tasks in a task set, utilization u_i is randomly drawn from the range$[0.05, 0.2]$.

- For any task τ_i, $C_i^{LO} = u_i * T_i$.

- $R \geq 1$ denotes the ratio C_i^{HI}/C_i^{LO} for every *high*-criticality task. If $L_i = HI$, we set $C_i^{HI} = R * C_i^{LO}$. It is easy to see that α used in the speedup factor function is equal to $\frac{1}{R}$;

- $\lambda \in (0, 1]$ denotes the ratio C_i^{HI}/C_i^{LO} for every *low*-criticality task. If $L_i = LO$, we set $C_i^{HI} = \lambda * C_i^{LO}$.

In the experiment, we generate IMC task sets with different target utilization U. Each task set is generated as follows. Given a target utilization U, we first initialize an empty task set. Then, we generate task τ_i according to the task generation procedure introduced above and add the generated task to the task set. The task set generation stops as we have

$$U - 0.05 \leq U_{avg} \leq U + 0.05$$

where

$$U_{avg} = \frac{U^{LO} + U^{HI}}{2}$$

is the average total utilization of the generated task set. If adding a new task makes $U_{avg} > U + 0.05$, then the added task will be removed and a new task will be generated and added to the task set till the condition is met.

6.5.1 Comparison with AMC [BB13]

In the first experiment, we compare EDF-VD by using our proposed test to the AMC approach in [BB13] in terms of average acceptance ratio. In this experiment, R is randomly selected from a uniform distribution $[1.5, 2.5]$. With different λ and pCriticality settings, we vary U_{avg} from 0.4 to 0.95 with step of 0.05, to evaluate the effectiveness of the proposed sufficient test in terms of the average acceptance ratios. We generate 10,000 task sets for each given U_{avg}. Since all experimental results follow the similar trend, in this section, we only present the experimental results when pCriticality$= 0.5$. Results with different pCriticality settings can be found in Appendix III. The results are shown in Figure 6.7-6.9, where the x-axis denotes the varying U_{avg} and the y-axis denotes the acceptance ratio. In the figures, let EDF-VD and AMC denote our proposed schedulability test and the one proposed in [BB13], respectively. In most cases, EDF-VD outperforms AMC in terms of acceptance ratio. We observe the following trends:

1. When $U_{avg} \in [0.5, 0.8]$, EDF-VD always outperforms AMC in terms of acceptance ratio. However, if $U_{avg} > 0.8$ and $\lambda = 0.3$ or 0.5, AMC performs better than EDF-VD. The same trend is also found for the classical MC model under EDF-VD and AMC, see in [EY14].

2. By comparing figures in Figure 6.7-6.9, we see that the average acceptance ratio improves when λ increases. This confirms the observation for the speedup factor we stated at the end of Section 6.4. The increasing λ leads to a smaller speedup factor. As a result, it provides a better schedulability. We need to notice that when λ increases, not only EDF-VD improves its acceptance ratio but the acceptance ratio of AMC [BB13] also improves.

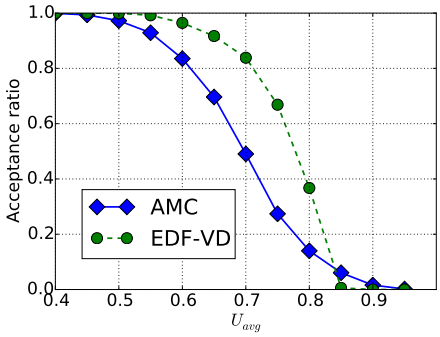

Figure 6.7: $\lambda = 0.3$

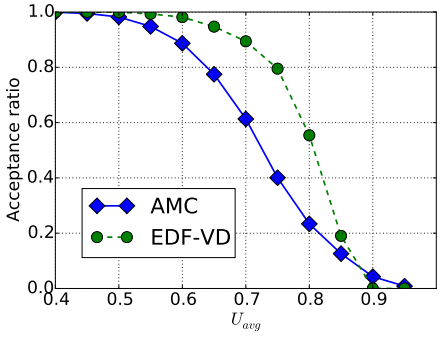

Figure 6.8: $\lambda = 0.5$

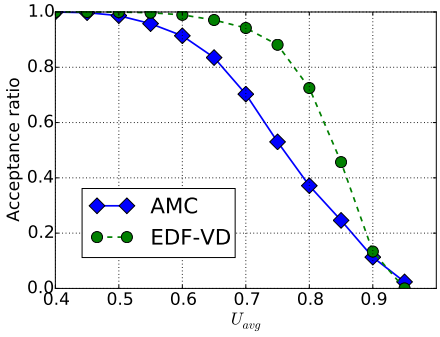

Figure 6.9: $\lambda = 0.7$

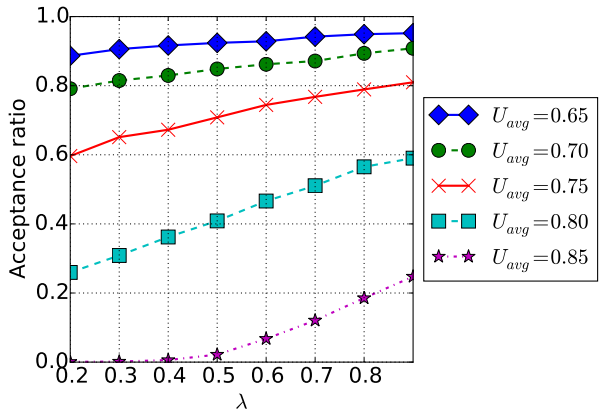

Figure 6.10: Impact of λ

6.5.2 Impact of α and λ

In the first experiment, we compare our proposed sufficient test to the existing AMC approach. In this section, we conduct experiments to further evaluate the impact of λ and α $(1/R)$ on the acceptance ratio. In this experiment, we select $U_{avg} = \{0.65, 0.7, 0.75, 0.8, 0.85\}$ to conduct experiments. We fix U_{avg} to a certain utilization and vary λ and α to evaluate the impact.

We first show the results for λ. The results are depicted in Figure 6.10, where the x-axis denotes the value of λ from 0.2 to 0.9 with step of 0.1 and the y-axis denotes the average acceptance ratio. R is randomly selected from a uniform distribution $[1.5, 2.5]$ and pCriticality$= 0.5$. Similarly, 10,000 task sets are generated for each point in the figures. A clear trend can be observed that the acceptance ratio increases as λ increases. This trend confirms the positive impact of increasing λ on the schedulability which we have observed in Section 6.4.

Next we conduct experiments to evaluate the impact of α on the schedulability. Similarly, we fix U_{avg} and vary α to carry out the experiments. Due to $\alpha = \frac{1}{R}$, if α is given, we compute the corresponding R to generate task sets. The results are depicted in Figure 6.11, where $\lambda = 0.5$. The x-axis denotes the varying α from 0.1 to 0.9 with step of 0.1. while the y-axis denotes the average acceptance ratio. First, from Table 6.2, we see that with increasing α the speedup factor first increases till a point. This means within this range the scheduling performance of EDF-VD gradually decreases. After that point, the speedup factor decreases which means the scheduling performance of EDF-VD gradually improves. The experimental results confirm what we have observed for α in Section 6.4. The acceptance ratio gradually decreases till a point and then it increases.

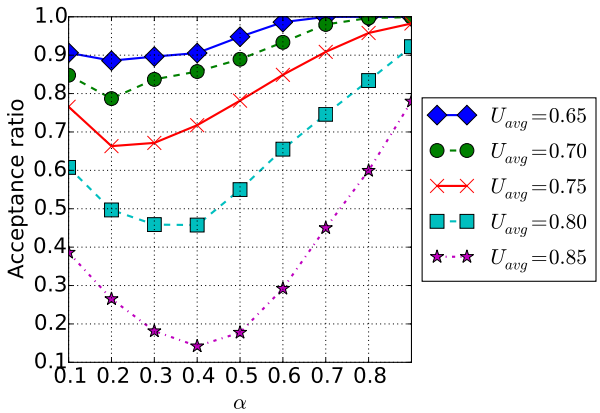

Figure 6.11: Impact of α