Exact Analysis of ZSRM Mixed-Criticality Scheduling of Sporadic Tasks

Björn Andersson, Dionisio de Niz, Hyoseung Kim, Mark Klein, and Ragunathan (Raj) Rajkumar
Carnegie Mellon University

***Abstract*—Zero-Slack Rate-Monotonic (ZSRM) is a family of mixed-criticality schedulers which are based on fixed-priority preemptive scheduling. One scheduler (which we call ZSRM-S) [5] works as follows: a job J is suspended at time t if at time t there is a higher-criticality job J' that has not finished and t minus the arrival time of J' exceeds a per-task configurable parameter (which we call zero-slack offset). ZSRM-S has two advantages compared to other mixed-criticality schedulers: (i) adaptation is local; i.e., there is no system-wide mode change needed and (ii) resumption is simple and natural. ZSRM-S has one drawback [7]: a high-criticality job J' can suffer from interference from a low-criticality job J that resumed after being suspended by another high-criticality job J'' (carry-in). Therefore, a variant of ZSRM (which we call ZSRM-SE) has been proposed [6]; it uses an enforcement mechanism to avoid carry-in. With ZSRM-SE, if a high-criticality job causes a low-criticality job to suspend and the high-criticality job has performed more execution than a certain bound then the low-criticality job shall not resume. We consider constrained-deadline sporadic tasks scheduled by ZSRM-S and present an exact schedulability test which solves a Mixed-Integer Linear Program (MILP). We also present that result for ZSRM-SE.**

I. Introduction

The problem of scheduling real-time tasks with different criticalities is not new [9, 11] but the trend towards the increasing use of embedded computers and consolidating multiple functionalities onto a single computer platform has increased the importance of this problem. For this reason, researchers have, during recent years, developed more advanced schedulers and analysis methods for systems with tasks of different criticalities. The literature is extensive — see [3] for an excellent survey. Today, most schedulers for mixed criticality systems (MCS) rely on three ideas:

I1. A task is assigned a criticality level;

I2. If it is impossible to meet all deadlines and the scheduler has to let one task miss a deadline then the scheduler should let a lower criticality task miss a deadline;

I3. The execution time of a task is characterized by multiple numbers; each number is believed to be an upper bound on the execution time of the task but the confidence one has in this belief is different for different numbers.

The research community has used these ideas in different ways. One way is to extend schedulability analysis for classic fixed-priority or Earliest-Deadline-First so that when performing schedulability analysis to determine if task τ_i meets its deadlines then execution times of other tasks must be selected to be on the same confidence level as the criticality of task τ_i. It was found that many of the optimality results in non-MCS scheduling do not apply to MCS scheduling [12, 2]. Other works use run-time monitoring and adaptation; check if a low criticality task has executed for more than it should and if so, the system switches to a high-critical mode where only high-critical tasks are allowed to execute [1]. Such an approach has two drawbacks: (i) it uses a system-wide mode and hence a system-wide mode-change is needed and (ii) it specifies how to switch from normal mode to an overload mode but typically does not specify how to switch back. We believe an alternative should be sought and hence, we consider the following idea:

I4. Before run-time, for each task τ_i, compute a parameter Z_i and at run-time, if a job of task τ_i has not finished at time Z_i after its arrival then take action to adapt.

This idea has been used for non-MCS and for this context, the action taken at Z_i is to change priorities; such use is called dual-priority scheduling [4]. This idea has been used for MCS and for this context, the action taken is to suspend jobs; such a scheduler was called ZSRM [5]. Later papers have discussed different sematics for it [7]; therefore, we let ZSRM denote a family of schedulers rather than a specific scheduler. ZSRM has been useful as witnessed by the following facts: previous work has made available implementations of a ZSRM scheduler in the Linux kernel and in VxWorks, as well as a sufficient schedulability test for it and this schedulability test is available to software practitioners in the OSATE AADL workbench. And a modification of it was used in a UAV system to ensure that an overload in vision processing does not jeopardize deadline guarantees of flight-control software [6]. Hence, we believe ZSRM is one of the most practical ideas in mixed-criticality scheduling. One scheduler (which we call ZSRM-S) [5] works as follows: a job J is suspended at time t if at time t there is a higher-criticality job J' that has not finished and t minus the arrival time of J' exceeds Z_i. Hence, the S in the name ZSRM-S means suspend. ZSRM-S has one drawback [7]: a high-criticality job J' can suffer from interference from a low-criticality job J that resumed after being suspended by another high-criticality job J'' (carry-in). Therefore, a variant of ZSRM (which we call ZSRM-SE) has been proposed [6]; it uses an enforcement mechanism to avoid carry-in. With ZSRM-SE, if a high-criticality job causes a low-criticality job to suspend and the high-criticality job has performed more execution than a certain bound then the low-criticality job shall not resume. Hence, the E in the name ZSRM-SE means execution-time monitoring. Unfortunately, no exact analysis was known for these schedulers.

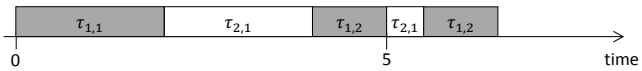
Therefore, in this paper, we consider constrained-deadline sporadic tasks scheduled by ZSRM-S and present an exact

$|\tau| = 2$
$T_1 = 4 \quad D_1 = 4 \;\; C_1 = 2 \quad C_1^o = 2 \;\; \zeta_1 = 1 \;\; \text{prio}_1 = 2 \;\; Z_1 = 2$
$T_2 = 10 \;\; D_2 = 8 \;\; C_2 = 2.5 \;\; C_2^o = 5 \;\; \zeta_2 = 2 \;\; \text{prio}_2 = 1 \;\; Z_2 = 5$

Fig. 1: An example of a taskset in our model.

$\text{nj}_1(R) = 2$
$A_{1,1}(R) = 0 \;\; c_{1,1}(R) = 2 \quad A_{1,2}(R) = 4 \;\; c_{1,2}(R) = 2$
$\text{nj}_2(R) = 1$
$A_{2,1}(R) = 0 \;\; c_{2,1}(R) = 2.5$

(a) An assignment R for the taskset in Fig. 1.



(b) ZSRM-S schedule of the assignment R for the taskset in Fig. 1.

Fig. 2: An assignment for the taskset in Fig. 1 and its ZSRM-S schedule. At time 5, the zero-slack instant of $\tau_{2,1}$ occurs so at this time, jobs from tasks with lower criticality get suspended; specifically, $\tau_{1,2}$ is suspended at this time. In (a), $c_{2,1}(R)$ is small so that when $\tau_{2,1}$ finishes, there is still time for $\tau_{1,2}$ to finish execution by its deadline (see (b)). For another assignment where $c_{2,1}$ is five, $\tau_{1,2}$ would miss its deadline.

schedulability test which solves a Mixed-Integer Linear Program (MILP). We also present that result for ZSRM-SE.

The rest of the paper is organized as follows. Section II presents the system model. Section III presents the new schedulability test for ZSRM-S. Section IV presents the new schedulability test for ZSRM-SE. Section V presents tools that perform the calculations of these schedulability tests. Section VI concludes.

II. System model

Throughout this paper, we let s.t. mean "such that" and we let : mean "it holds that" and we let $\{x|f(x)\}$ denote a set of elements so that an element x is in the set if and only if $f(x)$ is true. We let $\langle a, b\rangle$ indicate a tuple with two elements a and b. We let $[a, b]$ indicate an interval of real numbers. We let $\{a..b\}$ indicate the set of integers that are $\geq a$ and $\leq b$.

Static parameters. We consider a system comprising a taskset τ and a computer platform comprising a single processor. A task τ_i in τ is characterized by T_i, D_i, C_i, C_i^o, ζ_i, prio_i, and Z_i with the interpretation that τ_i generates a sequence of jobs with two consecutive jobs of τ_i having arrival times separated by at least T_i and each job of τ_i must finish within D_i time units. ζ_i indicates the criticality of τ_i. (If ζ_i is high then the criticality of τ_i is high.) prio_i indicates the priority of τ_i. (If prio_i is high then the priority of τ_i is high.) We assume $\forall \tau_i \in \tau$: $C_i \leq C_i^o$ and $D_i \leq T_i$. The symbol Z_i means zero-slack offset of τ_i and it is used by the scheduler to determine the time instant when jobs of lower criticality should be adapted (e.g. suspended). The symbols C_i and C_i^o are upper bounds on the execution time of a job of τ_i; the reason for having two upper bounds will be explained later in this section. For historical reasons, we refer to C_i as nominal execution time and C_i^o as overload execution time. Fig. 1 shows an example of a taskset in our model.

Run-time behavior of ZSRM-S. Let $\tau_{i,q}$ denote the q^{th} job of τ_i. Let R denote an assignment, for each task, the number of jobs it generates and for each of the jobs, an arrival time and execution time. Certain quantities that we define will be a function of the schedule and then we let sc be a schedule. Let $\text{nj}_i(R)$ denote the number of jobs that τ_i generates. Let $A_{i,q}(R)$ denote the arrival time of $\tau_{i,q}$. Let $c_{i,q}(R)$ denote the execution time of $\tau_{i,q}$. Let $f_{i,q}(\text{sc}, R)$ denote the finishing time of $\tau_{i,q}$. Let $\text{donex}_{i,q}(\text{t}, \text{sc}, R)$ denote the cumulative duration of execution of $\tau_{i,q}$ before time t. Fig. 3 shows predicates that we use. $\text{elig}(i, q, t, \tau, R, \text{sc})$ is a predicate that is true if, at time t, the job $\tau_{i,q}$ has arrived but not finished and $\tau_{i,q}$ is not suspended at time t because of higher-criticality jobs. $\text{eligZSRMS}(i, q, t, \tau, R, \text{sc})$ indicates that $\tau_{i,q}$ is eligible for execution (i.e., it is in the ready queue or it is running) at time t. Clearly, because of priority-based scheduling, an eligible job will only execute if there is no other eligible job with higher priority. We use the predicate $\text{candZSRMS}(i, q, t, \tau, R, \text{sc})$ to indicate that $\tau_{i,q}$ is a candidate for execution; i.e., $\tau_{i,q}$ is eligible and there is no eligible job of higher priority. An instant is a ZSRMSschedinst if there is a job that arrives at this instant or there is a job that finishes at this instant or there is a job that has its zero-slack instant at this instant — see Fig. 3. At each instant t such that t is a ZSRMSschedinst, the scheduler does the following: if there is at least one job $\tau_{i,q}$ such that $\text{candZSRMS}(i, q, t, \tau, R, \text{sc})$, then arbitrarily choose a job $\tau_{i,q}$ such that $\text{candZSRMS}(i, q, t, \tau, R, \text{sc})$ and execute it on the processor at time t and let it continue to execute until the next ZSRMSschedinst; if there is no job $\tau_{i,q}$ such that $\text{candZSRMS}(i, q, t, \tau, R, \text{sc})$, then keep the processor idle at time t until the next schedinst. Fig. 2 shows a schedule that the taskset in Fig. 1 can generate.

Run-time behavior of ZSRM-SE. The run-time behavior of ZSRM-SE differs from ZSRM-S only in that with ZSRM-SE, a job J is terminated if there was a time now or in the past such that at that time, there was a higher-criticality job J' that has reached its zero-slack instant and not finished and executed for more than its nominal execution time. We specify this formally with predicates in Fig. 3. The predicate $\text{candZSRMSE}(i, q, t, \tau, R, \text{sc}$ indicates that $\tau_{i,q}$ is a candidate for execution. The predicate $\text{eligZSRMSE}(i, q, t, \tau, R, \text{sc})$ indicates that $\tau_{i,q}$ is eligible for execution; if $\text{terminatedZSRMSE}(i, q, t, \tau, R, \text{sc})$ is true then $\tau_{i,q}$ is not eligible for execution. The predicate $\text{terminatedZSRMSE}(i, q, t, \tau, R, \text{sc})$ is true if there is a time t' such that $t' \leq t$ and $\text{terminatednowZSRMSE}(i, q, t', \tau, R, \text{sc})$. The predicate $\text{terminatednowZSRMSE}(i, q, t', \tau, R, \text{sc})$ is true if there is job $\tau_{i',q'}$ such that $\zeta_{i'} > \zeta_i$ and $\tau_{i',q'}$ has arrived but not finished and $\tau_{i',q'}$ has executed for more than $C_{i'}$ time units.

Schedulability and schedulability test of ZSRM-S. We say that a job $\tau_{i,q}$ is success if its finishing time is at most

$$
\begin{aligned}
\text{ZSRMSschedinst}(t,\tau,R,\text{sc}) =& (\exists \langle i,q\rangle \text{ s.t. } (\tau_i \in \tau) \wedge (q \in 1..\text{nj}_i(R)) \wedge ((A_{i,q}(R) = t) \vee (f_{i,q}(R,\text{sc}) = t) \vee (A_{i,q}(R) + Z_i = t)))\\
\text{ZSRMSEschedinst}(t,\tau,R,\text{sc}) =& (\exists \langle i,q\rangle \text{ s.t. } (\tau_i \in \tau) \wedge (q \in 1..\text{nj}_i(R)) \wedge ((A_{i,q}(R) = t) \vee (f_{i,q}(R,\text{sc}) = t) \vee (A_{i,q}(R) + Z_i = t) \vee\\
& (\text{donex}_{i',q'}(\text{t},\text{sc},R) = C_{i'})))\\
\text{arrived}(i,q,t,\tau,R,\text{sc}) =& (A_{i,q}(R) \leq t)\\
\text{Zd}(i,q,t,\tau,R,\text{sc}) =& (A_{i,q}(R) + Z_i \leq t)\\
\text{finZSRMS}(i,q,t,\tau,R,\text{sc}) =& (f_{i,q}(\text{sc},R) \leq t)\\
\text{finZSRMSE}(i,q,t,\tau,R,\text{sc}) =& ((f_{i,q}(\text{sc},R) \leq t) \vee (\text{terminated}(i,q,t,\tau,R,\text{sc})))\\
\text{arrivednotfinZSRMS}(i,q,t,\tau,R,\text{sc}) =& ((\text{arrived}(i,q,t,\tau,R,\text{sc})) \wedge (\neg \text{finZSRMS}(i,q,t,\tau,R,\text{sc})))\\
\text{arrivednotfinZSRMSE}(i,q,t,\tau,R,\text{sc}) =& ((\text{arrived}(i,q,t,\tau,R,\text{sc})) \wedge (\neg \text{finZSRMSE}(i,q,t,\tau,R,\text{sc})))\\
\text{ZdnotfinZSRMS}(i,q,t,\tau,R,\text{sc}) =& ((\text{Zd}(i,q,t,\tau,R,\text{sc})) \wedge (\neg \text{finZSRMS}(i,q,t,\tau,R,\text{sc})))\\
\text{ZdnotfinZSRMSE}(i,q,t,\tau,R,\text{sc}) =& ((\text{Zd}(i,q,t,\tau,R,\text{sc})) \wedge (\neg \text{finZSRMSE}(i,q,t,\tau,R,\text{sc})))\\
\text{lowexZSRMSE}(i,q,t,\tau,R,\text{sc}) =& (\text{donex}_{i,q}(\text{t},\text{sc},R) \leq C_i)\\
\text{ZdnotfinlowexZSRMSE}(i',q',t,\tau,R,\text{sc}) =& ((\text{ZdnotfinZSRMSE}(i,q,t,\tau,R,\text{sc})) \wedge (\text{lowexZSRMSE}(i,q,t,\tau,R,\text{sc})))\\
\text{ZdnotfinhiexZSRMSE}(i',q',t,\tau,R,\text{sc}) =& ((\text{ZdnotfinZSRMSE}(i,q,t,\tau,R,\text{sc})) \wedge (\neg \text{lowexZSRMSE}(i,q,t,\tau,R,\text{sc})))\\
\text{suspendednowZSRMS}(i,q,t,\tau,R,\text{sc}) =& (\exists \tau_{i',q'} \text{ s.t. } (\zeta_{i'} > \zeta_i) \wedge (\text{ZdnotfinZSRMS}(i',q',t,\tau,R,\text{sc})))\\
\text{suspendednowZSRMSE}(i,q,t,\tau,R,\text{sc}) =& ((\exists \tau_{i',q'} \text{ s.t. } (\zeta_{i'} > \zeta_i) \wedge (\text{ZdnotfinZSRMSE}(i',q',t,\tau,R,\text{sc}))) \wedge\\
& (\forall \tau_{i',q'} \text{ s.t. } (\zeta_{i'} > \zeta_i) : (\text{ZdnotfinZSRMSE}(i',q',t,\tau,R,\text{sc})) \Rightarrow\\
& \quad (\text{ZdnotfinlowexZSRMSE}(i',q',t,\tau,R,\text{sc}))))\\
\text{terminatednowZSRMSE}(i,q,t,\tau,R,\text{sc}) =& (\exists \tau_{i',q'} \text{ s.t. } (\zeta_{i'} > \zeta_i) \wedge (\text{ZdnotfinhiexZSRMSE}(i',q',t,\tau,R,\text{sc})))\\
\text{terminatedZSRMSE}(i,q,t,\tau,R,\text{sc}) =& (\exists t' \text{ s.t. } (t' \leq t) \wedge (\text{terminatednowZSRMSE}(i,q,t',\tau,R,\text{sc})))\\
\text{eligZSRMS}(i,q,t,\tau,R,\text{sc}) =& ((\text{arrivednotfinZSRMS}(i,q,t,\tau,R,\text{sc})) \wedge (\neg \text{suspendednowZSRMS}(i,q,t,\tau,R,\text{sc})))\\
\text{eligZSRMSE}(i,q,t,\tau,R,\text{sc}) =& ((\text{arrivednotfinZSRMSE}(i,q,t,\tau,R,\text{sc})) \wedge (\neg \text{suspendednowZSRMSE}(i,q,t,\tau,R,\text{sc})) \wedge\\
& (\neg \text{terminated}(i,q,t,\tau,R,\text{sc})))\\
\text{candZSRMS}(i,q,t,\tau,R,\text{sc}) =& ((\text{eligZSRMS}(i,q,t,\tau,R,\text{sc})) \wedge (\forall \tau_{i',q'} \text{ s.t. } \text{prio}_{i'} > \text{prio}_i : \neg \text{eligZSRMS}(i',q',t,\tau,R,\text{sc})))\\
\text{candZSRMSE}(i,q,t,\tau,R,\text{sc}) =& ((\text{eligZSRMSE}(i,q,t,\tau,R,\text{sc})) \wedge (\forall \tau_{i',q'} \text{ s.t. } \text{prio}_{i'} > \text{prio}_i : \neg \text{eligZSRMSE}(i',q',t,\tau,R,\text{sc})))\\
\text{legMCS}(R,\text{sc},\tau,\text{iD},\text{qD}) =& ((\forall \langle i,q\rangle \text{ s.t. } (\tau_i \in \tau) \wedge (q \in 2..\text{nj}_i(R)) : A_{i,q}(R) - A_{i,q-1}(R) \geq T_i) \wedge\\
& (\forall \langle i,q\rangle \text{ s.t. } (\tau_i \in \tau) \wedge (q \in 1..\text{nj}_i(R)) \wedge (\zeta_i > \zeta_{\text{iD}}) : c_{i,q}(R) \in [0, C_i]) \wedge\\
& (\forall \langle i,q\rangle \text{ s.t. } (\tau_i \in \tau) \wedge (q \in 1..\text{nj}_i(R)) \wedge (\zeta_i \leq \zeta_{\text{iD}}) : c_{i,q}(R) \in [0, C_i^o]))\\
\text{successZSRMS}(i,q,\tau,R,\text{sc}) =& (f_{i,q}(\text{sc},R) \leq A_{i,q}(R) + D_i)\\
\text{successZSRMSE}(i,q,\tau,R,\text{sc}) =& ((\neg \text{terminated}(i,q,A_{i,q}(R) + D_i,\tau,R,\text{sc})) \wedge (f_{i,q}(\text{sc},R) \leq A_{i,q}(R) + D_i))\\
\text{ZSRMSsch}(\tau) =& (\forall \langle \text{iD},\text{qD},R,\text{sc}\rangle \text{ s.t. } (\tau_{\text{iD}} \in \tau) \wedge (\text{qD} \in \{1..\text{nj}_{\text{iD}}(R)\}) \wedge\\
& (\text{legMCS}(R,\text{sc},\tau,\text{iD},\text{qD})) \wedge (\text{legZSRMSsch}(\text{sc},R,\tau)) : \text{successZSRMS}(\text{iD},\text{qD},\tau,R,\text{sc}))\\
\text{ZSRMSEsch}(\tau) =& (\forall \langle \text{iD},\text{qD},R,\text{sc}\rangle \text{ s.t. } (\tau_{\text{iD}} \in \tau) \wedge (\text{qD} \in \{1..\text{nj}_{\text{iD}}(R)\}) \wedge\\
& (\text{legMCS}(R,\text{sc},\tau,\text{iD},\text{qD})) \wedge (\text{legZSRMSEsch}(\text{sc},R,\tau)) : \text{successZSRMSE}(\text{iD},\text{qD},\tau,R,\text{sc}))
\end{aligned}
$$

Fig. 3: Predicates that we will use.

its deadline. The predicate $\text{successZSRMS}(i,q,\tau,R,\text{sc})$ indicates that. The predicate $\text{legMCS}(R,\text{sc},\tau,\text{iD},\text{qD})$ is true if R satisfies certain constraints (expressing arrival times and execution times of jobs) — see Fig. 3. Note that compared to the definition of a legal assignment used in classic fixed-priority scheduling without mixed criticalities, our definition of legal assignment differs in two ways (i) our definition takes a schedule as input whereas the classic definition does not and (ii) our definition takes a task and job index as input whereas the classic does not. The predicate $\text{legZSRMSsch}(\text{sc},R,\tau)$ indicates that schedule sc can be generated by ZSRM-S for assignment R for taskset τ. The predicate $\text{ZSRMSsch}(\tau)$ indicates that for each $\langle R,\text{iD},\text{qD},\text{sc}\rangle$ such that $\text{legMCS}(R,\text{sc},\tau,\text{iD},\text{qD})$ and $\text{legZSRMSsch}(\text{sc},R,\tau)$, it holds that $\text{successZSRMS}(\text{iD},\text{qD},\tau,R,\text{sc})$. Intuitively, the meaning of $\text{ZSRMSsch}(\tau)$ is that $\text{ZSRMSsch}(\tau)$ is true if each job J meets its deadline for the case that jobs of higher criticality than J have execution times that are bounded by the nominal execution times (not overload execution times). If $\text{ZSRSMsch}(\tau)$ is true then we say that the taskset is schedulable. Conversely, if $\text{ZSRMSsch}(\tau)$ is false then we say that the taskset is unschedulable. A schedulability test for ZSRM-S is a function that takes τ as input and outputs a boolean. For schedulability test ST associated with ZSRM-S, we say that ST is an *exact* schedulability test if $ST(\tau) \Leftrightarrow \text{ZSRMSsch}(\tau)$.

Schedulability and schedulability test of ZSRM-SE. The concepts for ZSRM-SE are analogous.

III. New Schedulability Test for ZSRM-S

Our goal in this section is to present an exact schedulability test for ZSRM-S. Traditional analysis of fixed-priority preemptive scheduling on a single processor relies on a con-

Sets :

$$\mathrm{TS} = \{i' | (\tau_{i'} \in \tau) \wedge ((\mathrm{prio}_{i'} \geq \mathrm{prio}_{\mathrm{iD}}) \vee (\zeta_{i'} \geq \zeta_{\mathrm{iD}}))\}, \mathrm{TSHC}(i) = \{i' | (\tau_{i'} \in \tau) \wedge (\zeta_{i'} > \zeta_{\mathrm{i}})\},$$
$$\mathrm{TSHP}(i) = \{i' | (\tau_{i'} \in \tau) \wedge (\mathrm{prio}_{i'} > \mathrm{prio}_{\mathrm{i}})\}, \mathrm{QS}(i') = \{1..\lceil \frac{t}{T_{i'}} \rceil\}, \mathrm{PS} = \{1..3 \times \sum_{i' \in \mathrm{TS}} \lceil \frac{t}{T_{i'}} \rceil\}$$

Constraints :

$$t^1 = 0 \qquad \forall p \in (\mathrm{PS} \setminus \{|\mathrm{PS}|\}) : t^p \leq t^{p+1}$$
$$\forall \langle i, q \rangle \text{ s.t. } (i \in \mathrm{TS}) \wedge (q \in (\mathrm{QS}(i) \setminus \{1\})) : A_{i,q} - A_{i,q-1} \geq T_i$$
$$\forall \langle i, q \rangle \text{ s.t. } (i \in \mathrm{TS}) \wedge (q \in \mathrm{QS}(i)) :$$
$$\sum_{p' \in \mathrm{PS}} \mathrm{arrives}_{i,q}^{p'} = 1 \qquad \sum_{p' \in \mathrm{PS}} \mathrm{finishes}_{i,q}^{p'} = 1 \qquad \sum_{p' \in \mathrm{PS}} \mathrm{ZS}_{i,q}^{p'} = 1$$
$$\forall \langle i, q, p \rangle \text{ s.t. } (i \in \mathrm{TS}) \wedge (q \in \mathrm{QS}(i)) \wedge (p \in \mathrm{PS}) :$$
$$(\mathrm{arrives}_{i,q}^{p} = 1) \Rightarrow (\mathrm{donex}_{i,q}^{p} = 0) \quad (\mathrm{arrives}_{i,q}^{p} = 1) \Rightarrow (A_{i,q} = t^p) \quad \mathrm{arrived}_{i,q}^{p} = \sum_{p' \in \{1..p\}} \mathrm{arrives}_{i,q}^{p'}$$
$$(\mathrm{finishes}_{i,q}^{p} = 1) \Rightarrow (\mathrm{donex}_{i,q}^{p} = c_{i,q}) \quad (\mathrm{finishes}_{i,q}^{p} = 1) \Rightarrow (f_{i,q} = t^p) \quad \mathrm{finZSRMS}_{i,q}^{p} = \sum_{p' \in \{1..p\}} \mathrm{finishes}_{i,q}^{p'}$$
$$(\mathrm{ZS}_{i,q}^{p} = 1) \Rightarrow (A_{i,q} + Z_i = t^p) \qquad \mathrm{Zd}_{i,q}^{p} = \sum_{p' \in \{1..p\}} \mathrm{ZS}_{i,q}^{p'}$$
$$(\mathrm{arrivednotfinZSRMS}_{i,q}^{p} = 1) \Leftrightarrow ((\mathrm{arrived}_{i,q}^{p} = 1) \wedge (\mathrm{finZSRMS}_{i,q}^{p} = 0))$$
$$(\mathrm{ZdnotfinZSRMS}_{i,q}^{p} = 1) \Leftrightarrow ((\mathrm{Zd}_{i,q}^{p} = 1) \wedge (\mathrm{finZSRMS}_{i,q}^{p} = 0))$$
$$(\mathrm{suspendednowZSRMS}_{i,q}^{p} = 1) \Rightarrow (\sum_{i' \in \mathrm{TSHC}(i)} \sum_{q' \in \mathrm{QS}(i')} \mathrm{ZdnotfinZSRMS}_{i',q'}^{p} \geq 1)$$
$$(\mathrm{suspendednowZSRMS}_{i,q}^{p} = 0) \Rightarrow (\sum_{i' \in \mathrm{TSHC}(i)} \sum_{q' \in \mathrm{QS}(i')} \mathrm{ZdnotfinZSRMS}_{i',q'}^{p} \leq 0)$$
$$(\mathrm{eligZSRMS}_{i,q}^{p} = 1) \Leftrightarrow ((\mathrm{arrivednotfinZSRMS}_{i,q}^{p} = 1) \wedge (\mathrm{suspendednowZSRMS}_{i,q}^{p} = 0))$$
$$(\mathrm{candZSRMS}_{i,q}^{p} = 1) \Leftrightarrow ((\mathrm{eligZSRMS}_{i,q}^{p} = 1) \wedge (\wedge_{i' \in \mathrm{TSHP}(i)} \wedge_{q' \in \mathrm{QS}(i')} (\mathrm{eligZSRMS}_{i',q'}^{p} = 0)))$$
$$\forall p \in (\mathrm{PS} \setminus \{|\mathrm{PS}|\}) :$$
$$(\mathrm{busyZSRMS}^{p} = 1) \Rightarrow (\sum_{\tau_{i'} \in \tau} \sum_{q' \in \mathrm{QS}(i')} \mathrm{candZSRMS}_{i',q'}^{p} \geq 1) \qquad (\mathrm{busyZSRMS}^{p} = 0) \Rightarrow (\sum_{\tau_{i'} \in \tau} \sum_{q' \in \mathrm{QS}(i')} \mathrm{candZSRMS}_{i',q'}^{p} \leq 0)$$
$$\forall \langle i, q, p \rangle \text{ s.t. } (i \in \mathrm{TS}) \wedge (q \in \mathrm{QS}(i)) \wedge (\mathrm{PS} \setminus \{|\mathrm{PS}|\}) :$$
$$(\mathrm{xZSRMS}_{i,q}^{p} = 1) \Rightarrow (\mathrm{donex}_{i,q}^{p+1} = \mathrm{donex}_{i,q}^{p} + t^{p+1} - t^p) \qquad \mathrm{x}_{i,q}^{p} \leq \mathrm{candZSRMS}_{i,q}^{p}$$
$$\forall p \in (\mathrm{PS} \setminus \{|\mathrm{PS}|\}) : \qquad \sum_{\tau_{i'} \in \mathrm{TS}} \sum_{q' \in \mathrm{QS}(i')} \mathrm{xZSRMS}_{i',q'}^{p} = \mathrm{busyZSRMS}^{p}$$
$$\forall p \in \mathrm{PS} : \qquad (\mathrm{finishes}_{\mathrm{iD},\mathrm{qD}}^{p} = 1) \Rightarrow (\sum_{p' \in \{1..p-1\}} \mathrm{busyZSRMS}^{p'} \geq p - 1)$$
$$\forall \langle i, q \rangle \text{ s.t. } (i \in \mathrm{TS}) \wedge (q \in \mathrm{QS}(i)) \wedge (\zeta_i > \zeta_{\mathrm{iD}}) : \qquad c_{i,q} \leq C_i$$
$$\forall \langle i, q \rangle \text{ s.t. } (i \in \mathrm{TS}) \wedge (q \in \mathrm{QS}(i)) \wedge (\zeta_i \leq \zeta_{\mathrm{iD}}) : \qquad c_{i,q} \leq C_i^o$$

Domains of variables :$\mathrm{t}^p \in \mathbb{R}_{\geq 0}, \mathrm{A}_{i,q} \in \mathbb{R}_{\geq 0}, \mathrm{c}_{i,q} \in \mathbb{R}_{\geq 0}, \mathrm{f}_{i,q} \in \mathbb{R}_{\geq 0}, \mathrm{donex}_{i,q}^{p} \in \mathbb{R}_{\geq 0}, \mathrm{arrives}_{i,q}^{p} \in \{0,1\}, \mathrm{arrived}_{i,q}^{p} \in \{0,1\},$
$\mathrm{finishes}_{i,q}^{p} \in \{0,1\}, \mathrm{finZSRMS}_{i,q}^{p} \in \{0,1\}, \mathrm{ZS}_{i,q}^{p} \in \{0,1\}, \mathrm{Zd}_{i,q}^{p} \in \{0,1\}, \mathrm{arrivednotfinZSRMS}_{i,q}^{p} \in \{0,1\}, \mathrm{ZdnotfinZSRMS}_{i,q}^{p} \in \{0,1\},$
$\mathrm{suspendednowZSRMS}_{i,q}^{p} \in \{0,1\}, \mathrm{eligZSRMS}_{i,q}^{p} \in \{0,1\}, \mathrm{candZSRMS}_{i,q}^{p} \in \{0,1\}, \mathrm{busyZSRMS}_{i,q}^{p} \in \{0,1\}, \mathrm{xZSRMS}_{i,q}^{p} \in \{0,1\}$

Fig. 4: Constraints we use for exact schedulability analysis of ZSRM-S.

dition for critical instant [10] or the concept busy period [8]. Unfortunately, these concepts cannot be used directly for exact schedulability analysis of ZSRM-S because in ZSRM-S, a job may be suspended. Thus, we will develop new ideas and put them together into an exact schedulability test for ZSRM-S.

For this discussion, let $\mathrm{feas}(X)$ denote a predicate that is true if and only if X (a set of constraints) is feasible. Let $\max\{\mathrm{myobj}|X\}$ denote the largest value of myobj subject to the constraints X. Also, let $\mathrm{ct}(t, \mathrm{iD}, \mathrm{qD})$ denote the set of constraints in Fig. 4 where (i) t in Fig. 4 is a constant which is equal to the 1^{st} parameter of ct, (ii) iD in Fig. 4 is a constant which is equal to the 2^{nd} parameter of ct, and (iii) qD in Fig. 4 is a constant which is equal to the 3^{rd} parameter of ct.

Our first lemma states certain properties of a time interval for an unschedulable taskset.

Lemma 1.

$$(\neg \mathrm{ZSRMSsch}(\tau)) \Rightarrow$$
$$(\exists \langle \mathrm{iD}, \mathrm{qD}, R, \mathrm{sc} \rangle \text{ s.t. } (\tau_{\mathrm{iD}} \in \tau) \wedge (\mathrm{qD} \in \{1..\mathrm{nj}_{\mathrm{iD}}(R)\}) \wedge$$
$$(\mathrm{legMCS}(R, \mathrm{sc}, \tau, \mathrm{iD}, \mathrm{qD})) \wedge (\mathrm{legZSRMSsch}(\mathrm{sc}, R, \tau)) \wedge$$
$$(f_{\mathrm{iD},\mathrm{qD}}(\mathrm{sc}, R) - A_{\mathrm{iD},\mathrm{qD}} > D_{\mathrm{iD}}) \wedge$$

$$\begin{aligned}&(\text{for schedule sc, it holds that at all times}\\&\text{before time } 0, \text{the processor is idle})\wedge\\&(\text{for schedule sc, it holds that at all times}\\&\text{in } [0, f_{\mathrm{iD,qD}}(\mathrm{sc}, R)], \text{the processor executes}\\&\text{a job with priority} \geq \mathrm{prio}_{\mathrm{iD}} \text{ or criticality } \geq \zeta_{\mathrm{iD}}))\end{aligned}$$

Proof: Assume that the left-hand side of the lemma is true. Then, from the definition of ZSRMSsch, it holds that:

$$\begin{aligned}&\exists\langle \mathrm{iD}, \mathrm{qD}, R, \mathrm{sc}\rangle \text{ s.t. } (\tau_{\mathrm{iD}} \in \tau) \wedge (\mathrm{qD} \in \{1..\mathrm{nj}_{\mathrm{iD}}(R)\})\wedge\\&(\mathrm{legMCS}(R, \mathrm{sc}, \tau, \mathrm{iD}, \mathrm{qD})) \wedge (\mathrm{legZSRMSsch}(\mathrm{sc}, R, \tau))\wedge\\&(\neg \mathrm{successZSRMS}(\mathrm{iD}, \mathrm{qD}, \tau, R, \mathrm{sc}))\end{aligned}$$

From the definition of successZSRMS we obtain an inequality, which applied on the above yields that:

$$\begin{aligned}&\exists\langle \mathrm{iD}, \mathrm{qD}, R, \mathrm{sc}\rangle \text{ s.t. } (\tau_{\mathrm{iD}} \in \tau) \wedge (\mathrm{qD} \in \{1..\mathrm{nj}_{\mathrm{iD}}(R)\})\wedge\\&(\mathrm{legMCS}(R, \mathrm{sc}, \tau, \mathrm{iD}, \mathrm{qD})) \wedge (\mathrm{legZSRMSsch}(\mathrm{sc}, R, \tau))\wedge\\&(f_{\mathrm{iD,qD}}(\mathrm{sc}, R) - A_{\mathrm{iD,qD}} > D_{\mathrm{iD}})\end{aligned} \tag{1}$$

In the schedule sc above, we can form a time interval that (i) ends at time $f_{\mathrm{iD,qD}}(\mathrm{sc}, R)$ and (ii) begins at the earliest time such that in this time interval, only jobs with priority $\geq \mathrm{prio}_{\mathrm{iD}}$ or criticality $\geq \zeta_{\mathrm{iD}}$ execute. One can delete all jobs arriving before this time interval. We can also set the origin of the time axis to be such that time zero is the time when this time interval begins. Applying this reasoning on (1) yields:

$$\begin{aligned}&\exists\langle \mathrm{iD}, \mathrm{qD}, R, \mathrm{sc}\rangle \text{ s.t. } (\tau_{\mathrm{iD}} \in \tau) \wedge (\mathrm{qD} \in \{1..\mathrm{nj}_{\mathrm{iD}}(R)\})\wedge\\&(\mathrm{legMCS}(R, \mathrm{sc}, \tau, \mathrm{iD}, \mathrm{qD})) \wedge (\mathrm{legZSRMSsch}(\mathrm{sc}, R, \tau))\wedge\\&(f_{\mathrm{iD,qD}}(\mathrm{sc}, R) - A_{\mathrm{iD,qD}} > D_{\mathrm{iD}})\wedge\\&(\text{for schedule sc, it holds that at all times}\\&\text{before time } 0, \text{the processor is idle})\wedge\\&(\text{for schedule sc, it holds that at all times}\\&\text{in } [0, f_{\mathrm{iD,qD}}(\mathrm{sc}, R)], \text{the processor executes}\\&\text{a job with priority} \geq \mathrm{prio}_{\mathrm{iD}} \text{ or criticality } \geq \zeta_{\mathrm{iD}})\end{aligned}$$

This is the right-hand side of the lemma. ∎

Our second lemma states that if the taskset is unschedulable then there exists a tuple $\langle \mathrm{iD}, \mathrm{qD}, t\rangle$ such that a certain problem is infeasible and t is at most a certain bound. When expressing this bound, we need to compute

$$\max_{\mathrm{qD}' \in \{1..\lceil \frac{t'}{T_{\mathrm{iD}}} \rceil\} \wedge (\mathrm{feas}(\mathrm{ct}(t', \mathrm{iD}, \mathrm{qD}')))} \max\{f_{\mathrm{iD,qD}'} | \mathrm{ct}(t', \mathrm{iD}, \mathrm{qD}')\} \tag{2}$$

This expression means the folllowing: (i) iterate over all qD' (which may be different from qD), (ii) check if the current qD' is such that $\mathrm{ct}(t', \mathrm{iD}, \mathrm{qD}')$ is feasible, (iii) if the answer to the preceding question is yes, then evaluate $\max\{f_{\mathrm{iD,qD}'} | \mathrm{ct}(t', \mathrm{iD}, \mathrm{qD}')\}$, and (iv) take the maximum of the computed values. One can see that for $\mathrm{qD}' = 1$, it holds that $\mathrm{ct}(t', \mathrm{iD}, \mathrm{qD}')$ is feasible. Hence, the evaluation in step (ii) is true for at least one iteration. And hence, the expression in (2) is well defined. With this, we can state our second lemma.

Lemma 2.

$$\begin{aligned}&(\neg \mathrm{ZSRMSsch}(\tau)) \Rightarrow\\&(\exists\langle \mathrm{iD}, \mathrm{qD}, t\rangle \text{ s.t. } (t > 0) \wedge (\tau_{\mathrm{iD}} \in \tau) \wedge (\mathrm{qD} \in \{1..\lceil \frac{t}{T_{\mathrm{iD}}} \rceil\})\wedge\\&(\mathrm{feas}(\{f_{\mathrm{iD,qD}} - A_{\mathrm{iD,qD}} > D_{\mathrm{iD}}\} \cup \mathrm{ct}(t, \mathrm{iD}, \mathrm{qD})))\wedge\\&(t \leq\\&\min\{t' | t' = \max_{\mathrm{qD}' \in \{1..\lceil \frac{t'}{T_{\mathrm{iD}}} \rceil\} \wedge (\mathrm{feas}(\mathrm{ct}(t', \mathrm{iD}, \mathrm{qD}')))} \max\{f_{\mathrm{iD,qD}'} | \mathrm{ct}(t', \mathrm{iD}, \mathrm{qD}')\}\}))\end{aligned}$$

Proof: Assume that the left-hand side of the lemma is true. Then, using Lemma 1 yields that:

$$\begin{aligned}&\exists\langle \mathrm{iD}, \mathrm{qD}, R, \mathrm{sc}\rangle \text{ s.t. } (\tau_{\mathrm{iD}} \in \tau) \wedge (\mathrm{qD} \in \{1..\mathrm{nj}_{\mathrm{iD}}(R)\})\wedge\\&(\mathrm{legMCS}(R, \mathrm{sc}, \tau, \mathrm{iD}, \mathrm{qD})) \wedge (\mathrm{legZSRMSsch}(\mathrm{sc}, R, \tau))\wedge\\&(f_{\mathrm{iD,qD}}(\mathrm{sc}, R) - A_{\mathrm{iD,qD}} > D_{\mathrm{iD}})\wedge\\&(\text{for schedule sc, it holds that at all times}\\&\text{before time } 0, \text{the processor is idle})\wedge\\&(\text{for schedule sc, it holds that at all times}\\&\text{in } [0, f_{\mathrm{iD,qD}}(\mathrm{sc}, R)], \text{the processor executes}\\&\text{a job with priority} \geq \mathrm{prio}_{\mathrm{iD}} \text{ or criticality } \geq \zeta_{\mathrm{iD}})\end{aligned} \tag{3}$$

Clearly, in the schedule sc above, the rules of dispatching (expressed in Fig. 3) applies and the assignment R is legal. Let us consider the part of schedule sc during $[0, f_{\mathrm{iD,qD}}(\mathrm{sc}, R)]$ and let us introduce t as $t = f_{\mathrm{iD,qD}}(\mathrm{sc}, R)$. We can encode this schedule with variables and constraints — indeed $\mathrm{ct}(t, \mathrm{iD}, \mathrm{qD})$, expressed in Fig. 4 does that. One can understand this encoding as follows: Clearly, for each task $\tau_{i'}$, there are at most $\lceil \frac{t}{T_{i'}} \rceil$ jobs of $\tau_{i'}$. Then we introduce variables that are direct analogs of the assignment R. The variable $A_{i',q'}$ in Fig. 4 is the arrival time of $\tau_{i',q'}$ and $c_{i',q'}$ in Fig. 4 is the execution time of $\tau_{i',q'}$. In Fig. 4, TS denotes the set of task that can generate jobs that can execute in the time interval $[0, t]$. In Fig. 4, $\mathrm{QS}(i')$ denotes the set of indices of jobs of task $\tau_{i'}$ that can execute in the time interval $[0, t]$. Recall that an instant is a schedinst if there is a job that arrives at this instant or there is a job that finishes at this instant or there is a job that has its zero-slack instant at this instant. Since we consider a time interval of duration t, and no jobs arrive before the time interval, it holds that there are at most $3 \times \sum_{i' \in \mathrm{TS}} \lceil \frac{t}{T_{i'}} \rceil$ instants that are schedinst. We can divide time into sub-time-intervals that are non-intersecting and that these instants separate the sub-time-intervals. This gives us that (i) a sub-time-interval begins at an instant that is a schedinst and (ii) if a sub-time-interval is not the last one, then it ends at an instant that is a schedinst and (iii) within a sub-time-interval, there is no instant that is a schedinst. We call these sub-time-intervals *positions* and we let t^p denote the time when the p^{th} position starts. There are at most $3 \times \sum_{i' \in \mathrm{TS}} \lceil \frac{t}{T_{i'}} \rceil - 1$ positions. We let PS denote the set $\mathrm{PS} = \{1..3 \times \sum_{i' \in \mathrm{TS}} \lceil \frac{t}{T_{i'}} \rceil\}$, i.e., if

$p' \leq |\mathrm{PS}| - 1$ then $t^{p'}$ is the beginning of the p'^{th} position and if $p' = |\mathrm{PS}|$ then $t^{p'}$ is the end of the $p' - 1^{th}$ position (the last position). Since we consider the time interval $[0, t]$, it holds that the first position starts at time 0, that is, $t^1 = 0$. For each position that is not the first position, it holds that its starting time is constrained to be at least as large as the starting time of its predecessor position; we express it as $\forall p \in (\mathrm{PS} \setminus \{1\}) : t^{p-1} \leq t^p$.

We can then express whether an event occurs in the beginning of a position. $\mathrm{arrives}^p_{i,q}$ is a variable in $\{0, 1\}$; if $\mathrm{arrives}^p_{i,q} = 1$ then it means that $\tau_{i,q}$ arrives in the beginning of position p. $\mathrm{arrived}^p_{i,q}$ is a variable in $\{0, 1\}$; if $\mathrm{arrived}^p_{i,q} = 1$ then it means that $\tau_{i,q}$ arrives in the beginning of position p or in an earlier position. $\mathrm{finishes}^p_{i,q}$ is a variable in $\{0, 1\}$; if $\mathrm{finishes}^p_{i,q} = 1$ then it means that $\tau_{i,q}$ finishes in the beginning of position p. $\mathrm{finZSRMS}^p_{i,q}$ is a variable in $\{0, 1\}$; if $\mathrm{finZSRMS}^p_{i,q} = 1$ then it means that $\tau_{i,q}$ finishes in the beginning of position p or in an earlier position. $\mathrm{ZS}^p_{i,q}$ is a variable in $\{0, 1\}$; if $\mathrm{ZS}^p_{i,q} = 1$ then it means that $\tau_{i,q}$ arrives exactly Z_i before the beginning of position p. $\mathrm{Zd}^p_{i,q}$ is a variable in $\{0, 1\}$; if $\mathrm{Zd}^p_{i,q} = 1$ then it means that there is a position $p' \leq p$ such that $\mathrm{ZS}^{p'}_{i,q} = 1$. Fig. 3 shows predicates and these predicates describe dispatching. We can introduce variables that describe if a predicate is true for a job at a time which is the beginning of a position. For example, $\mathrm{arrivednotfinZSRMS}^p_{i,q}$ is a variable in $\{0, 1\}$; if $\mathrm{arrivednotfinZSRMS}^p_{i,q} = 1$ then it means that $\tau_{i,q}$ arrives in the beginning of position p or earlier and $\tau_{i,q}$ finishes in the beginning of a position later than p. In Fig. 4, we express this as $(\mathrm{arrivednotfinZSRMS}^p_{i,q} = 1) \Leftrightarrow ((\mathrm{arrived}^p_{i,q} = 1) \wedge (\mathrm{finZSRMS}^p_{i,q} = 0))$. Other variables in Fig. 4 describe predicates in Fig. 3 analogously. In the end, we obtain a variable $\mathrm{candZSRMS}^p_{i,q}$ which describes that $\tau_{i,q}$ is a candidate for execution in the beginning of position p. Recall that a job is a candidate for execution if it is eligible and there is no other eligible job with higher priority at this time. We then introduce $\mathrm{xZSRMS}^p_{i,q}$ which is a variable in $\{0, 1\}$; if $\mathrm{xZSRMS}^p_{i,q} = 1$ then it means that $\tau_{i,q}$ executes in position p. Clearly, a job can only execute if it is a candidate. In Fig. 4, we express this as $\mathrm{xZSRMS}^p_{i,q} \leq \mathrm{candZSRMS}^p_{i,q}$. The above reasoning yields:

$$\exists \langle \mathrm{iD}, \mathrm{qD}, t \rangle \text{ s.t. } (t > 0) \wedge (\tau_{\mathrm{iD}} \in \tau) \wedge (\mathrm{qD} \in \{1..\lceil \frac{t}{T_{\mathrm{iD}}} \rceil\}) \wedge (\mathrm{feas}(\{f_{\mathrm{iD,qD}} - A_{\mathrm{iD,qD}} > D_{\mathrm{iD}}\} \cup \{f_{\mathrm{iD,qD}} = t\} \cup \mathrm{ct}(t, \mathrm{iD}, \mathrm{qD}))) \tag{4}$$

Let us now discuss the length of the busy period mentioned in (3). It can be seen that if $\tau_{\mathrm{iD,qD}}$ misses its deadline in a time interval where only jobs with priority $\geq \mathrm{prio}_{\mathrm{iD}}$ or criticality $\geq \zeta_{\mathrm{iD}}$, then

$$f_{\mathrm{iD,qD}}(\mathrm{sc}, R) \leq \min\{t' | t' = \max\{f_{\mathrm{iD,qD}} | \mathrm{ct}(t', \mathrm{iD}, \mathrm{qD})\}\}$$

Clearly, since $t = f_{\mathrm{iD,qD}}(\mathrm{sc}, R)$, we obtain:

$$t \leq \min\{t' | t' = \max\{f_{\mathrm{iD,qD}} | \mathrm{ct}(t', \mathrm{iD}, \mathrm{qD})\}\}$$

The right-hand side of this expression contains the symbol qD. We would like to find an upper bound that does not depend on qD. It can be seen that:

$$t \leq \min\{t' | t' = \max_{\mathrm{qD}' \in \{1..\lceil \frac{t'}{T_{\mathrm{iD}}} \rceil\}} \max\{f_{\mathrm{iD,qD}'} | \mathrm{ct}(t', \mathrm{iD}, \mathrm{qD}')\}\}$$

Combining it with (4) yields:

$$\exists \langle \mathrm{iD}, \mathrm{qD}, t \rangle \text{ s.t. } (t > 0) \wedge (\tau_{\mathrm{iD}} \in \tau) \wedge (\mathrm{qD} \in \{1..\lceil \frac{t}{T_{\mathrm{iD}}} \rceil\}) \wedge (\mathrm{feas}(\{f_{\mathrm{iD,qD}} - A_{\mathrm{iD,qD}} > D_{\mathrm{iD}}\} \cup \{f_{\mathrm{iD,qD}} = t\} \cup \mathrm{ct}(t, \mathrm{iD}, \mathrm{qD}))) \wedge (t \leq \min\{t' | t' = \max_{\mathrm{qD}' \in \{1..\lceil \frac{t'}{T_{\mathrm{iD}}} \rceil\} \wedge (\mathrm{feas}(\mathrm{ct}(t', \mathrm{iD}, \mathrm{qD}')))} \max\{f_{\mathrm{iD,qD}'} | \mathrm{ct}(t', \mathrm{iD}, \mathrm{qD}')\}\})$$

Dropping one constraints cannot cause infeasibility. Hence, by dropping $\{f_{\mathrm{iD,qD}} = t\}$ from the above, we have:

$$\exists \langle \mathrm{iD}, \mathrm{qD}, t \rangle \text{ s.t. } (t > 0) \wedge (\tau_{\mathrm{iD}} \in \tau) \wedge (\mathrm{qD} \in \{1..\lceil \frac{t}{T_{\mathrm{iD}}} \rceil\}) \wedge (\mathrm{feas}(\{f_{\mathrm{iD,qD}} - A_{\mathrm{iD,qD}} > D_{\mathrm{iD}}\} \cup \mathrm{ct}(t, \mathrm{iD}, \mathrm{qD}))) \wedge (t \leq \min\{t' | t' = \max_{\mathrm{qD}' \in \{1..\lceil \frac{t'}{T_{\mathrm{iD}}} \rceil\} \wedge (\mathrm{feas}(\mathrm{ct}(t', \mathrm{iD}, \mathrm{qD}')))} \max\{f_{\mathrm{iD,qD}'} | \mathrm{ct}(t', \mathrm{iD}, \mathrm{qD}')\}\})$$

This is the right-hand side of the lemma. ■

Our third lemma states how a change in t impacts certain inequalities.

Lemma 3. *If $t_a < t_b$ then it holds that:*

$$(\exists \langle \mathrm{iD}, \mathrm{qD} \rangle \text{ s.t. } (\tau_{\mathrm{iD}} \in \tau) \wedge (\mathrm{qD} \in \{1..\lceil \frac{t_a}{T_{\mathrm{iD}}} \rceil\}) \wedge (\mathrm{feas}(\{f_{\mathrm{iD,qD}} - A_{\mathrm{iD,qD}} > D_{\mathrm{iD}}\} \cup \mathrm{ct}(t_a, \mathrm{iD}, \mathrm{qD}))))$$
$$\Rightarrow$$
$$(\exists \langle \mathrm{iD}, \mathrm{qD} \rangle \text{ s.t. } (\tau_{\mathrm{iD}} \in \tau) \wedge (\mathrm{qD} \in \{1..\lceil \frac{t_b}{T_{\mathrm{iD}}} \rceil\}) \wedge (\mathrm{feas}(\{f_{\mathrm{iD,qD}} - A_{\mathrm{iD,qD}} > D_{\mathrm{iD}}\} \cup \mathrm{ct}(t_b, \mathrm{iD}, \mathrm{qD}))))$$

Proof: Assume that the left-hand side is true. Since the left-hand side is true, we know that there is a solution to the constraints. We can copy that solution to use it to satisfy the constraints on the right-hand side and then for the new variables that only exists on the right-hand side but not on the left-hand side, we can set them to zero. This yields a solution to the constraints on the right-hand side. And hence the right-hand side is true. ■

Our fourth lemma states certain inequalities for an unschedulable taskset (it differs from the second lemma only in that it uses $=$ instead of $\leq$) on the right-hand side.

Lemma 4.

$$(\neg\text{ZSRMSsch}(\tau)) \Rightarrow$$
$$(\exists\langle \text{iD}, \text{qD}, t\rangle \text{ s.t. } (t>0) \wedge (\tau_{\text{iD}} \in \tau) \wedge (\text{qD} \in \{1..\lceil \frac{t}{T_{\text{iD}}} \rceil\}) \wedge$$
$$(\text{feas}(\{f_{\text{iD,qD}} - A_{\text{iD,qD}} > D_{\text{iD}}\} \cup \text{ct}(t, \text{iD}, \text{qD}))) \wedge$$
$$(t = \min\{t' | t' = \max_{\text{qD}' \in \{1..\lceil \frac{t'}{T_{\text{iD}}} \rceil\} \wedge (\text{feas}(\text{ct}(t', \text{iD}, \text{qD}')))} \max\{f_{\text{iD,qD}'} | \text{ct}(t', \text{iD}, \text{qD}')\}\}))$$

Proof: Follows from applying Lemma 3 on Lemma 2. ■

We will now discuss another direction of implication; we will discuss $\Leftarrow$ instead of $\Rightarrow$. Our fifth lemma states that if certain inequalities are true then the taskset is unschedulable.

Lemma 5.

$$(\neg\text{ZSRMSsch}(\tau)) \Leftarrow$$
$$(\exists\langle \text{iD}, \text{qD}, t\rangle \text{ s.t. } (t>0) \wedge (\tau_{\text{iD}} \in \tau) \wedge (\text{qD} \in \{1..\lceil \frac{t}{T_{\text{iD}}} \rceil\}) \wedge$$
$$(\text{feas}(\{f_{\text{iD,qD}} - A_{\text{iD,qD}} > D_{\text{iD}}\} \cup \text{ct}(t, \text{iD}, \text{qD}))))$$

Proof: Assume that the right-hand side of the lemma is true. Then there exists a $\langle \text{iD}, \text{qD}, t\rangle$ such that the right-hand side is true. Since the constraints on the right-hand side are feasible, we have an assignment of values to the variables in $\text{ct}(t, \text{iD}, \text{qD})$ and with this assignment of values to variables we obtain an assignment R and can (based on the discussion in Lemma 2 and using R), construct a schedule during $[0, t]$ such that $\tau_{\text{iD,qD}}$ misses its deadline. Hence, it holds that

$$\exists\langle \text{iD}, \text{qD}, R, \text{sc}\rangle \text{ s.t. } (\tau_{\text{iD}} \in \tau) \wedge (\text{qD} \in \{1..\text{nj}_{\text{iD}}(R)\}) \wedge$$
$$(\text{legMCS}(R, \text{sc}, \tau, \text{iD}, \text{qD})) \wedge (\text{legZSRMSsch}(\text{sc}, R, \tau)) \wedge$$
$$(f_{\text{iD,qD}} - A_{\text{iD,qD}} > D_{\text{iD}})$$

This can be rewritten as :

$$(\neg\text{ZSRMSsch}(\tau))$$

This is the left-hand side of the lemma. ■

We can consider Lemma 5 but add additional constraints on the right-hand side.

Lemma 6.

$$(\neg\text{ZSRMSsch}(\tau)) \Leftarrow$$
$$(\exists\langle \text{iD}, \text{qD}, t\rangle \text{ s.t. } (t>0) \wedge (\tau_{\text{iD}} \in \tau) \wedge (\text{qD} \in \{1..\lceil \frac{t}{T_{\text{iD}}} \rceil\}) \wedge$$
$$(\text{feas}(\{f_{\text{iD,qD}} - A_{\text{iD,qD}} > D_{\text{iD}}\} \cup \text{ct}(t, \text{iD}, \text{qD}))) \wedge$$
$$(t = \min\{t' | t' = \max_{\text{qD}' \in \{1..\lceil \frac{t'}{T_{\text{iD}}} \rceil\} \wedge (\text{feas}(\text{ct}(t', \text{iD}, \text{qD}')))} \max\{f_{\text{iD,qD}'} | \text{ct}(t', \text{iD}, \text{qD}')\}\}))$$

Proof: Follows from Lemma 5. ■

We then present an exact condition for unschedulability.

```
1.  allOK := true
2.  for each τ_iD ∈ τ, as long as allOK do
3.    t := −1; newt := C^o_iD
4.    while (t < newt) do
5.      t := newt
6.      flag := false
7.      for each qD' ∈ {1..⌈t/T_iD⌉} do
8.        < fe, va >:= solve(max{f_iD,qD' | ct(t, iD, qD')})
9.        if fe then
10.         if flag then newt := max(newt, va)
11.         else newt := va; flag := true
12.         end if
13.       end if
14.     end for
15.   end while
16.   for each qD ∈ {1..⌈t/T_iD⌉}, as long as allOK do
17.     if feas({f_iD,qD − A_iD,qD > D_iD} ∪
            ct(t, iD, qD)) then
18.       allOK := false
19.     end if
20.   end for
21. end for
22. return allOK
```

Fig. 5: An algorithm for ZSRM-S schedulability testing.

Lemma 7.

$$(\neg\text{ZSRMSsch}(\tau)) \Leftrightarrow$$
$$(\exists\langle \text{iD}, \text{qD}, t\rangle \text{ s.t. } (t>0) \wedge (\tau_{\text{iD}} \in \tau) \wedge (\text{qD} \in \{1..\lceil \frac{t}{T_{\text{iD}}} \rceil\}) \wedge$$
$$(\text{feas}(\{f_{\text{iD,qD}} - A_{\text{iD,qD}} > D_{\text{iD}}\} \cup \text{ct}(t, \text{iD}, \text{qD}))) \wedge$$
$$(t = \min\{t' | t' = \max_{\text{qD}' \in \{1..\lceil \frac{t'}{T_{\text{iD}}} \rceil\} \wedge (\text{feas}(\text{ct}(t', \text{iD}, \text{qD}')))} \max\{f_{\text{iD,qD}'} | \text{ct}(t', \text{iD}, \text{qD}')\}\}))$$

Proof: Follows from Lemma 4 and Lemma 6 ■

We then present an exact conditions for schedulability.

Theorem 1.

$$\text{ZSRMSsch}(\tau) \Leftrightarrow$$
$$(\forall \text{iD} \text{ s.t. } (\tau_{\text{iD}} \in \tau) :$$
$$\text{for } t =$$
$$\min\{t' | t' = \max_{\text{qD}' \in \{1..\lceil \frac{t'}{T_{\text{iD}}} \rceil\} \wedge (\text{feas}(\text{ct}(t', \text{iD}, \text{qD}')))} \max\{f_{\text{iD,qD}'} | \text{ct}(t', \text{iD}, \text{qD}')\}\} :$$
$$\forall \text{qD} \in \{1..\lceil \frac{t}{T_{\text{iD}}} \rceil\} :$$
$$\neg\text{feas}(\{f_{\text{iD,qD}} - A_{\text{iD,qD}} > D_{\text{iD}}\} \cup \text{ct}(t, \text{iD}, \text{qD}))$$
$$)$$

Proof: Follows from rewriting Lemma 7. ■

Evaluating the right-hand side of Theorem 1 requires calculating $\min\{t' | t' = \max_{\text{qD}' \in \{1..\lceil \frac{t'}{T_{\text{iD}}} \rceil\}} \max\{f_{\text{iD,qD}'} | \text{ct}(t', \text{iD}, \text{qD}')\}\}$ — let tmin denote this. It can be seen that $\max_{\text{qD}' \in \{1..\lceil \frac{t'}{T_{\text{iD}}} \rceil\}} \max\{f_{\text{iD,qD}'} | \text{ct}(t', \text{iD}, \text{qD}')\}$ is non-increasing with increasing t' (follows from reasoning

```
1.  allOK := true
2.  for each τ_iD ∈ τ, as long as allOK do
3.    t := −1; newt := C^o_iD
4.    while (t < newt) and allOK do
5.      t := newt
6.      for each qD ∈ {1..⌈t/T_iD⌉}, as long as allOK do
7.        if feas({f_iD,qD − A_iD,qD > D_iD}∪
              ct(t, iD, qD)) then
8.          allOK := false
9.        end if
10.     end for
11.     if allOK then
12.       flag := false
13.       for each qD′ ∈ {1..⌈t/T_iD⌉} do
14.        < fe, va >:= solve(max{f_iD,qD′ | ct(t, iD, qD′)})
15.        if fe then
16.          if flag then newt := max(newt, va)
17.          else newt := va; flag := true
18.          end if
19.        end if
20.      end for
21.    end if
22.  end while
23.  end for
24.  return allOK
```

Fig. 6: An algorithm for ZSRM-S schedulability testing; it is optimized for detecting clearly unschedulable tasksets quickly.

analogous to the reasoning in the proof of Lemma 3). Therefore, we can evaluate tmin with standard iterative procedure. Fig. 5 is an algorithm that uses such an iterative procedure to perform the schedulability test as expressed by Theorem 1. We use the notation $< \mathrm{fe}, \mathrm{va} >:= \mathrm{solve}(\max\{\mathrm{PROB}\})$ to state that the optimization problem PROB should be solved and fe is a boolean which is true if the problem is feasible and false otherwise; if the problem is feasible then ve is the value of the objective function for an optimal solution.

For many tasksets that are unschedulable, it holds that there is a job that misses its deadline at an early time in a busy period. Unfortunately, Fig. 5 requires that we obtain $t \geq \mathrm{newt}$ before we can even start checking the existence of deadline misses. We would like to get early termination for such tasksets. By using Lemma 5, we can check a given $\langle \mathrm{iD}, \mathrm{qD}, t \rangle$ to see if it satisfies certain conditions and if this is the case, we know that the taskset is unschedulable. We can apply this condition for the t after line 5 in Fig. 5. By adding such a check, we know that lines 16 to 20 in Fig. 5 are not needed. With these observations, we can rewrite the algorithm Fig. 5 into the algorithm in Fig. 6.

IV. New Schedulability Test for ZSRM-SE

In this section, we present an exact schedulability test for ZSRM-SE. Note that ZSRM-SE differs from ZSRM-S in only two ways. First, the definition of success of a job is different; a job can be not-success if the job misses its deadline (just like in ZSRM-S) but a job can also be not-success if it is terminated. Second, the schedules that can be generated by ZSRM-SE are different from the ones that can be generated by ZSRM-S. Hence, use the constraints in Fig. 4 as a starting point and observe that ZSRM-SE is impacted by termination condition and hence, we add constraints for that and this results in the constraints in Fig. 7. Note that in Fig. 7, we have a variable $\mathrm{terminatednow}^p_{i,q}$ with the interpretation that if $\mathrm{terminatednow}^p_{i,q} = 1$ then $\tau_{i,q}$ is terminated at the beginning of position p. There is also a predicate $\mathrm{terminated}^p_{i,q}$ with the interpretation that if $\mathrm{terminated}^p_{i,q} = 1$ then $\tau_{i,q}$ is terminated at the beginning of position p or earlier. With these variables, we can define $\mathrm{eligZSRMSE}^p_{i,q}$ that describe whether $\tau_{i,q}$ is eligible at the beginning of position p; it is calculated based on $\mathrm{terminated}^p_{i,q}$. Therefore, if $\tau_{\mathrm{iD,qD}}$ is a not-success job then it holds that the there is a t such that the following constaints are feasible: $\{((\mathrm{terminated}^{\mathrm{PS}}_{\mathrm{iD,qD}} = 1) \vee (f_{\mathrm{iD,qD}} > A_{\mathrm{iD,qD}} + D_{\mathrm{iD}}))\} \cup \mathrm{ct2}(t, \mathrm{iD}, \mathrm{qD})$, where ct2 is the set of constraints in Fig. 7 and PS is the last position. We also introduce $\mathrm{ft}_{i,q}$ — meaning failure time — which is a variable that states the time that $\tau_{i,q}$ generated a failure. If $\tau_{i,q}$ is terminated then $\mathrm{ft}_{i,q}$ is the time when it got terminated. If $\tau_{i,q}$ is not terminated then $\mathrm{ft}_{i,q}$ is the time when it finished. Our formulation here also differ from the one in the previous section in that the number of scheduling instants is greater; here each job can generate four scheduling instants — the time when a job has executed exactly its nominal execution time can be a scheduling instant as well (because job termination can happen at such an instant).

Theorem 2.

$$
\begin{aligned}
&\mathrm{ZSRMSEsch}(\tau) \Leftrightarrow \\
&(\forall \mathrm{iD \text{ s.t. } } (\tau_{\mathrm{iD}} \in \tau) : \\
&\quad \text{for } t = \\
&\qquad \min\{t' | t' = \max_{\mathrm{qD}' \in \{1..\lceil \frac{t'}{T_{\mathrm{iD}}} \rceil\}} \max\{\mathrm{ft}_{\mathrm{iD,qD}'} | \mathrm{ct2}(t', \mathrm{iD}, \mathrm{qD}')\}\} : \\
&\qquad\qquad \forall \mathrm{qD} \in \{1..\lceil \frac{t}{T_{\mathrm{iD}}} \rceil\} : \\
&\quad \neg\mathrm{feas}(\{(\mathrm{terminated}^{\mathrm{PS}}_{\mathrm{iD,qD}} = 1) \vee (f_{\mathrm{iD,qD}} - A_{\mathrm{iD,qD}} > D_{\mathrm{iD}})\} \cup \\
&\qquad\qquad \mathrm{ct2}(t, \mathrm{iD}, \mathrm{qD})) \\
&)
\end{aligned}
$$

Proof: This is a direct extension of Theorem 1. ∎

Fig. 8 is an algorithm that uses such an iterative procedure to perform the schedulability test as expressed by Theorem 2.

V. Our tool

Recall Fig. 5 presented an algorithm for performing exact schedulability analysis of ZSRM-S and Fig. 7 presented an algorithm for performing exact schedulability analysis of ZSRM-SE. These algorithms have in common that they check if a set of constraints is feasible and they also solve a problem of maximizing an objective function subject to certain constaints. Some of the constraints mentioned are not MILP — they have binary variables and logical operators. We will now discuss how to convert them to MILP. In our problems, we can add, for each real variable a the constraint: $a \leq \mathrm{BIG}$ where BIG is a constant computed as $\mathrm{BIG} = \sum_{\tau_i \in \tau} \lceil \frac{t}{T_i} \rceil \times C^o_i$.

Sets :

$$\mathrm{TS} = \{i' | (\tau_{i'} \in \tau) \wedge ((\mathrm{prio}_{i'} \geq \mathrm{prio}_{\mathrm{iD}}) \vee (\zeta_{i'} \geq \zeta_{\mathrm{iD}}))\}, \mathrm{TSHC}(i) = \{i' | (\tau_{i'} \in \tau) \wedge (\zeta_{i'} > \zeta_{\mathrm{i}})\},$$
$$\mathrm{TSHP}(i) = \{i' | (\tau_{i'} \in \tau) \wedge (\mathrm{prio}_{i'} > \mathrm{prio}_{\mathrm{i}})\}, \mathrm{QS}(i') = \{1..\lceil \frac{t}{T_{i'}} \rceil\}, \mathrm{PS} = \{1..4 \times \sum_{i' \in \mathrm{TS}} \lceil \frac{t}{T_{i'}} \rceil\}$$

Constraints :

$$t^1 = 0 \qquad \forall p \in (\mathrm{PS} \setminus \{|\mathrm{PS}|\}) : t^p \leq t^{p+1} :$$
$$\forall \langle i, q \rangle \text{ s.t. } (i \in \mathrm{TS}) \wedge (q \in (\mathrm{QS}(i) \setminus \{1\})) : A_{i,q} - A_{i,q-1} \geq T_i$$
$$\forall \langle i, q \rangle \text{ s.t. } (i \in \mathrm{TS}) \wedge (q \in \mathrm{QS}(i)) : \sum_{p' \in \mathrm{PS}} \mathrm{arrives}_{i,q}^{p'} = 1 \qquad \sum_{p' \in \mathrm{PS}} \mathrm{ZS}_{i,q}^{p'} = 1$$
$$\forall \langle i, q, p \rangle \text{ s.t. } (i \in \mathrm{TS}) \wedge (q \in \mathrm{QS}(i)) \wedge (p \in (\mathrm{PS} \setminus \{|\mathrm{PS}|\})) : \sum_{p' \in \mathrm{PS}} \mathrm{finishes}_{i,q}^{p'} = 1 - \mathrm{terminated}_{i,q}^{p}$$
$$(\mathrm{arrives}_{i,q}^{p} = 1) \Rightarrow (\mathrm{donex}_{i,q}^{p} = 0) \qquad (\mathrm{arrives}_{i,q}^{p} = 1) \Rightarrow (A_{i,q} = t^p) \qquad \mathrm{arrived}_{i,q}^{p} = \sum_{p' \in \{1..p\}} \mathrm{arrives}_{i,q}^{p'}$$
$$(\mathrm{finishes}_{i,q}^{p} = 1) \Rightarrow (\mathrm{donex}_{i,q}^{p} = c_{i,q}) \qquad (\mathrm{finishes}_{i,q}^{p} = 1) \Rightarrow (f_{i,q} = t^p) \qquad \mathrm{fin}_{i,q}^{p} = \sum_{p' \in \{1..p\}} \mathrm{finishes}_{i,q}^{p'}$$
$$(\mathrm{ZS}_{i,q}^{p} = 1) \Rightarrow (A_{i,q} + Z_i = t^p) \qquad \mathrm{Zd}_{i,q}^{p} = \sum_{p' \in \{1..p\}} \mathrm{ZS}_{i,q}^{p'}$$
$$(\mathrm{doneexactly}_{i,q}^{p} = 1) \Rightarrow (\mathrm{donex}_{i,q}^{p} = C_i) \qquad \sum_{p' \in \{1..\}} \mathrm{doneexactly}_{i,q}^{p'} = \mathrm{highexeandnotterminated}_{i,q}^{p}$$
$$(\mathrm{highexeandnotterminated}_{i,q}^{p} = 1) \Leftrightarrow ((\mathrm{arrived}_{i,q}^{p} = 1) \wedge (\mathrm{atmostnom}_{i,q}^{p} = 0) \wedge (\mathrm{terminated}_{i,q}^{p} = 0))$$
$$(\mathrm{arrivednotfin}_{i,q}^{p} = 1) \Leftrightarrow ((\mathrm{arrived}_{i,q}^{p} = 1) \wedge (\mathrm{fin}_{i,q}^{p} = 0))$$
$$(\mathrm{Zdnotfin}_{i,q}^{p} = 1) \Leftrightarrow ((\mathrm{Zd}_{i,q}^{p} = 1) \wedge (\mathrm{fin}_{i,q}^{p} = 0))$$
$$(\mathrm{atmostnom}_{i,q}^{p} = 1) \Leftarrow (\mathrm{donex}_{i,q}^{p} \leq C_i)$$
$$(\mathrm{atmostnom}_{i,q}^{p} = 0) \Leftarrow (\mathrm{donex}_{i,q}^{p} > C_i)$$
$$(\mathrm{Zdnotfinandnotatmostnom}_{i,q}^{p} = 1) \Leftrightarrow ((\mathrm{Zd}_{i,q}^{p} = 1) \wedge (\mathrm{fin}_{i,q}^{p} = 0) \wedge (\mathrm{atmostnom}_{i,q}^{p} = 0))$$
$$\mathrm{terminated}_{i,q}^{p} = \sum_{p' \in \{1..p\}} \mathrm{terminatednow}_{i,q}^{p'}$$
$$(\mathrm{suspendednow}_{i,q}^{p} = 1) \Rightarrow (\sum_{i' \in \mathrm{TSHC}(i)} \sum_{q' \in \mathrm{QS}(i')} \mathrm{Zdnotfin}_{i',q'}^{p} \geq 1)$$
$$(\mathrm{suspendednow}_{i,q}^{p} = 0) \Rightarrow (\sum_{i' \in \mathrm{TSHC}(i)} \sum_{q' \in \mathrm{QS}(i')} \mathrm{Zdnotfin}_{i',q'}^{p} \leq 0)$$
$$(\mathrm{terminatednow}_{i,q}^{p} = 1) \Rightarrow (\sum_{i' \in \mathrm{TSHC}(i)} \sum_{q' \in \mathrm{QS}(i')} \mathrm{Zdnotfinandnotatmostnom}_{i',q'}^{p} \geq 1)$$
$$(\mathrm{terminatednow}_{i,q}^{p} = 0) \Rightarrow (\sum_{i' \in \mathrm{TSHC}(i)} \sum_{q' \in \mathrm{QS}(i')} \mathrm{Zdnotfinandnotatmostnom}_{i',q'}^{p} \leq 0)$$
$$(\mathrm{eligZSRMSE}_{i,q}^{p} = 1) \Leftrightarrow ((\mathrm{arrivednotfin}_{i,q}^{p} = 1) \wedge (\mathrm{suspendednow}_{i,q}^{p} = 0) \wedge (\mathrm{terminated}_{i,q}^{p} = 0))$$
$$(\mathrm{candZSRMSE}_{i,q}^{p} = 1) \Leftrightarrow ((\mathrm{eligZSRMSE}_{i,q}^{p} = 1) \wedge (\wedge_{i' \in \mathrm{TSHP}(i)} \wedge_{q' \in \mathrm{QS}(i')} (\mathrm{eligZSRMS}_{i,q}^{p} = 0)))$$
$$(\mathrm{busy}^p = 1) \Rightarrow (\sum_{\tau_{i'} \in \tau} \sum_{q' \in \mathrm{QS}(i')} \mathrm{candZSRMSE}_{i',q'}^{p} \geq 1) \qquad (\mathrm{busy}^p = 0) \Rightarrow (\sum_{\tau_{i'} \in \tau} \sum_{q' \in \mathrm{QS}(i')} \mathrm{candZSRMSE}_{i',q'}^{p} \leq 0)$$
$$(\mathrm{x}_{i,q}^{p} = 1) \Rightarrow (\mathrm{donex}_{i,q}^{p+1} = \mathrm{donex}_{i,q}^{p} + t^{p+1} - t^p) \qquad \mathrm{x}_{i,q}^{p} \leq \mathrm{candZSRMSE}_{i,q}^{p}$$
$$\forall p \in (\mathrm{PS} \setminus \{|\mathrm{PS}|\}) : \qquad \sum_{\tau_{i'} \in \tau} \sum_{q' \in \mathrm{QS}(i')} x_{i',q'}^{p} = \mathrm{busy}^p$$
$$\forall p \in \mathrm{PS} : \qquad (\mathrm{finishes}_{\mathrm{iD,qD}}^{p} = 1) \Rightarrow (\sum_{p' \in \{1..p-1\}} \mathrm{busy}^{p'} \geq p - 1)$$
$$\forall \langle i, q \rangle \text{ s.t. } (i \in \mathrm{TS}) \wedge (q \in \mathrm{QS}(i)) \wedge (\zeta_i > \zeta_{\mathrm{iD}}) : c_{i,q} \leq C_i \qquad \forall \langle i, q \rangle \text{ s.t. } (i \in \mathrm{TS}) \wedge (q \in \mathrm{QS}(i)) \wedge (\zeta_i \leq \zeta_{\mathrm{iD}}) : c_{i,q} \leq C_i^o$$
$$(\mathrm{terminated}_{\mathrm{iD,qD}} = 0) \Rightarrow (\mathrm{ft} = f_{\mathrm{iD,qD}}) \qquad \forall p \in \mathrm{PS} : \quad (\mathrm{terminatednow}_{\mathrm{iD,qD}}^{p} = 1) \Rightarrow (\mathrm{ft} = t^p)$$

Domains of variables :$\mathrm{t}^p \in \mathbb{R}_{\geq 0}, \mathrm{A}_{i,q}^{p} \in \mathbb{R}_{\geq 0}, \mathrm{arrives}_{i,q}^{p} \in \{0,1\}, \mathrm{arrived}_{i,q}^{p} \in \{0,1\}, \mathrm{f}_{i,q} \in \mathbb{R}_{\geq 0}, \mathrm{finishes}_{i,q}^{p} \in \{0,1\}$, $\mathrm{fin}_{i,q}^{p} \in \{0,1\}, \mathrm{ZS}_{i,q}^{p} \in \{0,1\}, \mathrm{Zd}_{i,q}^{p} \in \{0,1\}, \mathrm{donex}_{i,q}^{p} \in \mathbb{R}_{\geq 0}, c_{i,q} \in \{0,1\}, \mathrm{arrivednotfin}_{i,q}^{p} \in \{0,1\}, \mathrm{Zdnotfin}_{i,q}^{p} \in \{0,1\}$, $\mathrm{suspendednow}_{i,q}^{p} \in \{0,1\}, \mathrm{eligZSRMSE}_{i,q}^{p} \in \{0,1\}, \mathrm{candZSRMSE}_{i,q}^{p} \in \{0,1\}, \mathrm{busy}^p \in \{0,1\}, \mathrm{x}_{i,q}^{p} \in \{0,1\}, \mathrm{ft} \in \mathbb{R}_{\geq 0}$

Fig. 7: Constraints we use for exact schedulability analysis of ZSRM-SE.

```
1.  allOK := true
2.  for each τ_iD ∈ τ, as long as allOK do
3.    t := −1; newt := C^o_iD
4.    while (t < newt) and allOK do
5.      t := newt
6.      for each qD ∈ {1..⌈t/T_iD⌉}, as long as allOK do
7.        if feas({(terminated^PS_iD,qD = 1)∨
            (f_iD,qD − A_iD,qD > D_iD)} ∪ ct2(t, iD, qD)) then
8.          allOK := false
9.        end if
10.     end for
11.     if allOK then
12.       flag := false
13.       for each qD′ ∈ {1..⌈t/T_iD⌉} do
14.         < fe, va >:=
              solve(max{ft|ct2(t, iD, qD′)})
15.         if fe then
16.           if flag then newt := max(newt, va)
17.           else newt := va; flag := true
18.           end if
19.         end if
20.       end for
21.     end if
22.   end while
23. end for
24. return allOK
```

Fig. 8: An algorithm for ZSRM-SE schedulability testing; it is optimized for detecting clearly unschedulable tasksets quickly.

And this does not change feasibility. A constraint of the form $(x = 1) \Rightarrow (a = b)$ can be rewritten as: $((x = 1) \Rightarrow (a \leq b)) \wedge ((x = 1) \Rightarrow (a \geq b))$. Note that if x is a variable with the domain $\{0, 1\}$ and a and b are non-negative real variables and BIG is a constant selected so that $a \leq \text{BIG}$ and $b \leq \text{BIG}$, then a constraint $(x = 1) \Rightarrow (a \leq b)$ can be rewritten as

$$a - b + \text{BIG} \times x \leq \text{BIG} \tag{5}$$

Also, a constraint of the form $(a = 1) \Leftrightarrow (b = 1) \wedge (c = 1)$ can be rewritten as $(b + c - a \leq 1) \wedge (b + c - 2a \geq 0)$. Consider the constraint: $(\text{suspendednowZSRMS}^p_{i,q} = 1) \Rightarrow (\sum_{i' \in \text{TSHC}(i)} \sum_{q' \in \text{QS}(i')} \text{ZdnotfinZSRMS}^p_{i',q'} \geq 1)$. When we rewrite it, we use $\text{BIG} = 1 + (\sum_{i' \in \text{TSHC}(i)} |\text{QS}(i')|)$.

With these techniques, we can rewrite the optimization problems and feasibility checking problems are MILP. Indeed, we have done so and implemented a tool that performs these computations. Our implementation uses Gurobi 6.0.3 — a state-of-the-art MILP solver.

VI. Conclusions

Zero-Slack Rate-Monotonic (ZSRM) is a mixed-criticality scheduler which suspends a low-criticality tasks when a high-criticality tasks has not finished at a certain time. Previous work has made available implementations of a ZSRM scheduler in the Linux kernel and in VxWorks, as well as a sufficient schedulability test for it and this schedulability test is available to software practitioners in the OSATE AADL workbench. And a modification of it was used in a UAV system to ensure that an overload in vision processing does not jeopardize deadline guarantees of flight-control software [6]. Hence, we believe ZSRM is one of the most practical ideas in mixed-criticality scheduling. Unfortunately, no exact schedulability analysis was available for ZSRM schedulers. Therefore, in this paper, we presented exact schedulability tests for two ZSRM schedulers.

Acknowledgment

This material is based upon work funded and supported by the Department of Defense under Contract No. FA8721-05-C-0003 with Carnegie Mellon University for the operation of the Software Engineering Institute, a federally funded research and development center. This material has been approved for public release and unlimited distribution. DM-0002431

References

[1] S. K. Baruah, A. Burns, and R. I. Davis. Response-time analysis for mixed criticality systems. In *RTSS'11*.

[2] S. K. Baruah and S. Vestal. Schedulability analysis of sporadic tasks with multiple criticality specifications. In *ECRTS'08*.

[3] A. Burns and R. I. Davis. Mixed criticality systems — a review. Technical report, Department of Computer Science, University of York, 2015. www-users.cs.york.ac.uk/burns/review.pdf.

[4] R. I. Davis and A. J. Wellings. Dual priority scheduling. In *RTSS'95*.

[5] D. de Niz, K. Lakshmanan, and R. Rajkumar. On the scheduling of mixed-criticality real-time task sets. In *RTSS'09*.

[6] D. de Niz, L. Wrage, A. Rowe, and R. Rajkumar. Utility-based resource overbooking for cyber-physical systems. In *RTCSA'13*.

[7] H.-M. Huang, C. Gill, and C. Lu. Implementation and evaluation of mixed criticality scheduling approaches for periodic tasks. In *RTAS'12*.

[8] J. Lehoczky, L. Sha, and Y. Ding. The rate monotonic scheduling algorithm: Exact characterization and average case behavior. In *RTSS'90*.

[9] J. P. Lehoczky, L. Sha, and J. K. Strosnider. Enhanced aperiodic responsiveness in hard real-time environments. In *RTSS'87*.

[10] C. L. Liu and J. W. Layland. Scheduling algorithms for multiprogramming in a hard-real-time environment. *J. ACM*, 20(1):46–61, 1973.

[11] L. Sha, J. P. Lehoczky, and R. Rajkumar. Solutions for some practical problems in prioritized preemptive scheduling. In *RTSS'86*.

[12] S. Vestal. Preemptive scheduling of multi-criticality systems with varying degrees of execution time assurance. In *RTSS'07*.