Erratum for "Scheduling real-time mixed-criticality jobs"

Sanjoy Baruah, Vincenzo Bonifaci, Gianlorenzo D'Angelo, Pontus Ekberg, Haohan Li, Alberto Marchetti-Spaccamela, Nicole Megow, and Leen Stougie

***Abstract*—This is an erratum for "Scheduling real-time mixed-criticality jobs" (IEEE Transactions on Computers 2012 [1]) pointing out an error concerning the complexity status, namely the membership in the class NP of deciding schedulability of mixed-criticality (MC) instances. Membership was based on a lemma that appeared not to be true if jobs have release dates. In this erratum we repair the result by a direct argument.**

This note repairs a flaw in [1]. The paper considers the mixed-criticality real-time scheduling of a set of n jobs on a single machine. Each job $j = 1, \dots, n$ is characterized by a release date r_j, a deadline d_j, a criticality level χ_j and for each criticality level $\ell \leq \chi_j$ a worst-case execution time $P_j(\ell)$. A system is MC-schedulable if there exists a scheduling policy such that for any realization of the system all jobs with criticality level at least the realized level finish before their deadlines. In [1] the decision problem if the system is MC-schedulable is claimed to be in NP if L the number of criticality levels is fixed.

Theorem A (Theorem 2 of [1])**.** *The problem of deciding* MC-SCHEDULABILITY *for L criticality levels is in* NP *when L is a constant.*

The proof of this theorem was based on Lemma 2 of [1] to which Kahil, Poplavko, Socci and Bensalem [2] provide a counter-example. In fact, the lemma was also implicitly used in the proof of Theorem 3 in [1], which states that the problem is in PSPACE for arbitrary L.

We restate Lemma 2 in [1] and give a counter-example similar to (but smaller than) the one in [2], and point out the error in the proof. Then we repair the proof of Theorem A. We also show that the validity of Theorem 3 in [1] is not affected by the error in Lemma 2.

Baruah is with Washington University in St. Louis, USA. Email: baruah@wustl.edu. Bonifaci is with IASI-CNR, Rome, Italy. D'Angelo is with GSSI, L'Aquila, Italy. Ekberg is with Uppsala University, Sweden. Li is with The University of North Carolina, Chapel Hill, NC, USA. Marchetti-Spaccamela is with Sapienza University of Rome, Italy. Megow is with University of Bremen, Germany. Stougie is with CWI & Vrije Universiteit, Amsterdam, The Netherlands.

Claim 1 (Lemma 2 of [1])**.** *If an instance is* MC-SCHEDULABLE*, then there exists an optimal scheduling policy that preempts each job j only at time points t at which either some other job is released, or j has executed for exactly $P_j(i)$ units of time for some $1 \leq i \leq L$.*

The following example, with three jobs and 2 criticality levels, which is smaller (and to our taste simpler) than the one in [2], witnesses falseness of this lemma.

Example 1. *Let's consider the following instance of three jobs:*

job j	r_j	d_j	χ_j	$P_j(1)$	$P_j(2)$
1	*0*	*6*	*2*	*2*	*3*
2	*0*	*4*	*1*	*2*	*2*
3	*2*	*4*	*2*	*1*	*2*

Observe that an optimal policy must have scheduled by time 2 both,

(i) at least one unit of job 1, as otherwise if job 3 runs for $P_3(2) = 2$ time units at criticality level 2, jobs 3 and 1 cannot finish both by their deadlines, and

(ii) at least one unit of job 2, as otherwise in criticality level 1, we cannot finish both jobs 2 and 3 by their deadline 4.

Thus, both jobs 1 and 2 must be scheduled for one time unit in the time interval $[0, 2]$ which means that one job must be preempted at some time $t \leq 1$ even though t is no release date and no job has finished $P_j(i)$. □

We remark that Lemma 2 is true for MC scheduling instances with equal release dates. We don't prove this here, since it is irrelevant for the proof of Theorem A below.

Proof of Theorem A: We will prove inductively that for every integer constant $L \geq 2$, deciding MC-SCHEDULABILITY of instances with L criticality levels is an NP problem. Membership to the class NP requires that for a YES-answer a polynomial-time verifiable *certificate* of polynomial length can be given. As a base case, we first consider the case $L = 2$ and prove that there exists a certificate for a YES-answer that requires length and verification time $O(n^L)$.

We claim that the following is a certificate for positive MC-SCHEDULABILITY. For each job j we specify $C_j(1)$, the time at which job j has been assigned $P_j(1)$ processing time. Together with the release times of the jobs, this partitions the time into at most $2n$ intervals. For each interval I_i, we specify for each job j the amount x_{ij} of processing time it receives in the interval. This certificate has length $O(n^2)$.

To check that a certificate represents an admissible schedule of jobs requires to verify that: *i)* for each I_i the total processing requirement for all jobs assigned to I_i is less than its length $|I_i|$, i.e. $\sum_j x_{ij} \leq |I_i|$; *ii)* let $I_i = [a_i, b_i)$, then for all i and j we have that $x_{ij} > 0$ implies $r_j \leq a_i$. Given a certificate representing an admissible schedule, to check feasibility in case the system remains in criticality level 1 it is sufficient to verify for each j that $C_j(1) \leq d_j$ and $\sum_{i|C_j(1)\geq b_i} x_{ij} = P_j(1)$. Thus level-1 schedulability can be verified in $O(n^2)$ time.

Next, we need to check for each job j with criticality level 2, that, if it does not signal completion at time $C_j(1)$, then the yet unfinished level-2 jobs can still be scheduled before their deadlines. For each such a job k its remaining processing time at level 2 is $P_k(2) - \sum_{i|C_j(1)\geq b_i} x_{ik}$, which we denote by p'_k. Thus, deleting all level-1 jobs and all level-2 jobs k that have $C_k(1) < C_j(1)$, and setting for each job k with $p'_k > 0$ its release date to $r'_k = \max\{r_k, C_j(1)\}$ creates a single machine preemptive scheduling problem with release times and deadlines, which, if schedulable, is known to be schedulable by the Earliest Deadline First (EDF) rule. The outcome of EDF can clearly be verified in $O(n)$ time. Hence, accounting all values of j, which are $O(n)$, yields a total of $O(n^2)$ verification time for level-2 schedulability. Notice that an instance that is not MC-schedulable cannot have such a certificate, since either the level 1 or the level 2 verification step would fail.

To be precise we should also show that certificates for YES-answers do not require exponentially small values of x_{ij} and $C_j(1)$. Assuming that all input numbers are integers, clearly each $C_j(1)$ can be taken integer, whence all intervals I_i have integer boundaries. Level-1 schedulability is certified by any feasible solution of the transportation problem constituted by supply points i with supply the length $|I_i|$ of I_i, and demand points j with demand $P_j(1)$. By total unimodularity, if feasible, an integer solution must exist, yielding integral x_{ij} values. As a consequence, the EDF instances that are checked for level-2 schedulability also have integral release times and deadlines.

Now suppose NP-membership holds for $L-1$ criticality levels and consider the case with L levels. As in the level-2 case we specify the level-1 behavior specifying $C_j(1)$ for each job j and x_{ij} for each interval i and j. By the same argument this requires input of length $O(n^2)$ and we can verify level-1 schedulability in $O(n^2)$ time.

Then for each job j with criticality level higher than 1 we need to verify MC-SCHEDULABILITY in case it does not signal completion at time $C_j(1)$. Again, as in the level-2 case we redefine release time and required remaining processing time at level-2 of all jobs of level ≥ 2 that have not yet completed (at level 1), and obtain a system with $L-1$ criticality levels. By induction schedulability can be certificated using input length and time $O(n^{L-1})$. Since there are $O(n)$ jobs of level ≥ 2 we obtain a certificate of length and verification time $O(n \cdot n^{L-1}) = O(n^L)$. ■

Example 2. *In the case of Example 1, a possible guess is that $C_1(1) = 5$, $C_2(1) = 4$, and $C_3(1) = 3$; the following is a corresponding certificate of* MC-SCHEDULABILITY*:*

interval	x_{i1}	x_{i2}	x_{i3}
$I_1 = [0, 2)$	*1*	*1*	*0*
$I_2 = [2, 3)$	*0*	*0*	*1*
$I_3 = [3, 4)$	*0*	*1*	*0*
$I_4 = [4, 5)$	*1*	*0*	*0*
$I_5 = [5, 6)$	*0*	*0*	*0*

□

As mentioned earlier, the correctness of Theorem 3 in [1] is not compromised by the error in Lemma 2. For the sake of completeness, we restate Theorem 3 in [1] and its proof.

Theorem B (Theorem 3 of [1])**.** *The problem of deciding* MC-SCHEDULABILITY *is in* PSPACE.

Proof: Consider the decision tree representation of an optimal online policy. Notice that we cannot store the whole tree in space that is polynomial in n when L is large. However, we can still check that such a tree exists by generating in depth first order all paths from the root to a leaf, while making sure that the common portion of consecutive paths is consistent. It is enough to store two paths at a time. Each path requires space proportional to its depth, which is $\mathcal{O}(n^2L)$ because Lemma 1 in [1] implies that there are $\mathcal{O}(nL)$ preemptions between any two release dates. Moreover, to keep track of the depth first search, a counter of size

$\mathcal{O}(n^2L)$ suffices because there are $\mathcal{O}(2^{n^2L})$ potential paths, the tree being binary. Finally, as in the proof of Theorem 2 in [1] we verify for each path that the decisions of the policy generate a valid schedule. This yields a nondeterministic algorithm for deciding MC-schedulability that uses polynomial space. The claim follows by the well-known fact that nondeterminism can be removed from the algorithm, at the cost of squaring the required space. ■

We conclude with a summary of the complexity landscape.

– MC-SCHEDULABILITY is NP-complete for any fixed number $L \geq 2$ of criticality levels
– MC-SCHEDULABILITY is NP-hard but not known to be in NP (for non-fixed L)
– MC-SCHEDULABILITY is in PSPACE.

References

[1] S. K. Baruah, V. Bonifaci, G. D'Angelo, H. Li, A. Marchetti-Spaccamela, N. Megow, and L. Stougie. Scheduling real-time mixed-criticality jobs. *IEEE Trans. Computers*, 61(8):1140–1152, 2012.

[2] R. Kahil, P. Poplavko, D. Socci, and S. Bensalem. Revisiting the computational complexity of mixed-critical scheduling. In *Proceedings of the Fifth International Workshop on Mixed Criticality Systems (WMC)*, pages 25–30, December 2017.