Scheduling with IMITATOR: Some Case Studies

Romain Soulat

1 Sketch of the Method illustrated on an example

1.1 Instantiated Problem

In [1] Abdeddaïm et Maler have shown how to encode the preemptive scheduling using Timed Automata with stopwatches (denoted here by TA). The Job-shop scheduling problem is a generic resource allocation problem in which common resources ("machines") are required at various time points (and for given duration) by different tasks. The goal is to find a way to allocate the resources such that all the tasks terminate as soon as possible (or "minimal makespan" in the scheduling jargon). We consider a fixed set M of resources. A *step* is a pair (m, d) where $m \in M$ and $d \in \mathcal{N}$, indicating the required utilization of resource m for time duration d. A *job specification* is a finite sequence

$$J = (m_1, d_1), (m_2, d_2), \cdots, (m_k, d_k)$$

of steps stating that in order to accomplish job J, one needs to use a machine m_1 for d_1 time, then use machine m_2 for d_2 time etc. For details, see [1]. For example, consider the job-shop problem with 2 jobs on 3 machines defined by $\{J_1, J_2\}$ with $J_1 = (m_1, d_1), (m_2, d_2), (m_3, d_3)$ and $J_2 = (m_2, d_2')$ with $d_1 = 3, d_2 = 2, d_3 = 4, d_2' = 5$. There are many possible schedules; two of them are illustrated in Figure. 1.1. The problem is to find a schedule such that the value

Figure 1: Schedules

of the last completed job is minimal. This minimal value is called makespan. There are plenty techniques designed to solve this makespan problem. In [1], the authors show that the system can be modeled as a product of TAs, each TA corresponding to a job. The makespan problem then reduces to find the shortest path in the product. As a recapitulation, we have:

- Model M: A product of TAs
- Input I: $\{(m_{i,j}, d_{i,j})_{j \in J(i)}\}_{i \in \mathcal{I}}$
- Output: μ = minimal makespan

For the example, the makespan is found to be equal to 9.

1.2 Robustness Analysis

We first transform the instantiated problem by parametrizing some of the duration values. In the example, we will consider that d_2 and d'_2 are parameters. We consider accordingly the parametric version $\mathcal{PA}$ of the original TA. Besides, we consider the makespan (viz, 9) as an additional input, and we construct a simple test automaton which is composed with $\mathcal{PA}$ in order to test wether or not the last job is completed within the makespan. Finally, we give to IMITATOR as inputs:

1. the automaton resulting from the product of $\mathcal{PA}$ with the test automaton
2. an input valuation $\pi_0 = \{d_2 = 2, d'_2 = 5\}$ for the parameters.

As an output, IMITATOR gives us a constraint K on d_2, d'_2 such that:

1. $(2, 5) \in K$
2. the makespan of the system obtained by replacing d_2, d'_2 by v_2, v'_2 is less than or equal to 9, for all $(v_2, v'_2) \in K$.

On this example K is equal to $7 > d'_2 \wedge 3 > d_2 \wedge d'_2 + d_2 \geq 7$.[1] This gives us a quantitative measure of robustness for the input valuation $\pi_0 = \{d_2 = 2, d'_2 = 5\}$.

2 Cyclic Tasks with a given scheduling policy

2.1 Context

In this framework of case studies, we want to prove the schedulability of cyclic tasks on a single machine, we consider the preemptive framework, where a currently running task can be interrupted by another one when it becomes activated and has a higher priority. Each task τ_i is defined by an offset $O_i \in \mathcal{N}$ (the time before the first release), a period $T_i \in \mathcal{N}$ (the amount of time between two releases), a deadline $D_i \in \mathcal{N}$ (the maximum timed allowed to perform the task) and a Worst Case Execution Time $C_i \in \mathcal{N}$ (the maximum of work required to execute the task).

Added to these automata, we have an a priori scheduling policy either Fixed Priority (FP) or Earliest Deadline First (EDF). The first policy gives to each task a priority and when two tasks are activated at the same time, the computational time is given to the highest priority. The EDF policy gives the computational time to the task that has its deadline coming the sooner.

In this context, given a scheduler, the system is said to be *schedulable* if each task τ_i is completed before D_i time units after the beginning of its period. Actually because of the periodicity of the problem, we only have to ensure that it is schedulable within $lcm_{i \in \mathcal{I}}(T_i)$.

[1] There three couples of integer values that satisfy K: $(1, 6)$, $(2, 5)$, $(2, 6)$.

2.2 Instantiated Problem

The problem is the following: given a list of tasks $\{\tau_i\}_{i\in\mathcal{I}}$

$$\tau_i = (O_i, T_i, D_i, C_i),$$

is the system schedulable?

- Model M: A TA per task and a TA for the scheduler
- Input I: $\{O_i, T_i, D_i, C_i\}_{i\in I}$
- Output: Yes/No

2.3 Robustness Analysis

The problem is parametrized typically by letting deadlines D_is instantiated and parametrizing some of the C_is, O_is and T_is.

- Model M': a set of PTA deduced from M by parametrizing the values of interest
- Input: I and an identified set of parameters
- Output: a constraint K such that for every $\pi \models K$, the π-instantiated model $M'[\pi]$ is schedulable.

3 Premptive scheduling on one machine with Fixed Priority and variable execution times [6]

3.1 Context

The problem addressed here is how to determine whether every job in n independent job chains, denoted $J_1, J_2, \cdots, J_n$ can complete in time when the jobs are scheduled on a processor according to a priority-driven algorithm. Roughly speaking, we let $J_{i,j}$ denote the jth job of the chain J_i. Each job J_{i_j} has a fixed priority $\Phi_{i,j}$ and is preemptable. The execution time of $J_{i,j}$ is in the range $[e^-_{i,j}, e^+_{i,j}]$ with $e^-_{i,j}, e^+_{i,j}$ in $\mathcal{N}$. The release time $r_{i,j}$ of job $J_{i,j}$ is set to an integer value. $J_{i,1}$ is ready for execution at its release time $r_{i,1}$; for each $j > 1$, $J_{i,j}$ cannot execute until its immediate predecessor $J_{i,j-1}$ completes. For details, see [6].

3.2 Instantiated Problem

We are given a list of chain of tasks $\{J_i\}_{i\in\mathcal{I}}$ where $J_i = J_{i,j\,j\in\mathcal{J}(i)}$ and

$$J_{i,j} = ((r_{i,j}, e^-_{i,j}, e^+_{i,j}, \Phi_{i,j}).$$

The problem is: what is the minimum time needed to do all the chain of tasks?

- Model: A product of TAs, with a TA per task and a TA for the FP scheduler
- Input: $I = (r_{i,j}, e^-_{i,j}, e^+_{i,j}, \Phi_{i,j})_{i\in\mathcal{I}}$
- Output: Δ = Worst case completion time of the last task (of the last chain).

3.3 Robustness Analysis

The problem is parametrized typically by letting the priorities $\Phi_{i,j}$ and the $[e^-_{i,j}, e^+_{i,j}]$ instantiated as well as the computed value Δ, and by parametrizing some of the r_is.

- Model M': a PTA deduced from M by parametrizing the values of interest and composed with a TA that tests wether or not the last task is completed within Δ (using an error state)
- Input: I together with Δ and an identified set of parameters
- Output: a constraint K such that for every $\pi \models K$, the last task of the π-instantiated model $M'[\pi]$ is completed within Δ.

4 Cyclic Tasks with an FP scheduling policy and an error state [4, 5]

4.1 Context

if S is a periodic task system, the activation automaton simply consists of a network of n independent timed automata, each representing the activation of periodic task τ_i, as shown in Fig. 4.1; each task τ_i has a period T_i, a fixed duration of time between two activation events, and it may have an offset O_i for its first activation time. Its activation automaton contains a clock, which is reset every time the period is reached and $Release_i$ is fired. When a job is activated, it executes for at most a time C_i, and has to terminate within the relative deadline D_i. The activation of jobs can be modeled by a parametric timed automaton, where activation events are associated with transition labels, see Fig. 4.1.

A Fixed Priority scheduler automaton is added to these tasks. When the time elapsed from the last activation of a task equals to its deadline time and

Figure 2: Automaton for a periodic task with an offset

the total demanded time exceeds the length of the interval, the task misses the deadline and the automaton reaches an error state. For details see n [4].

We say that the system is *schedulable* if each task τ_i is completed before its relative deadline D_i. Actually, because of the periodicity of the system, we only have to be sure that it is schedulable within $lcm_{i\in I}(T_i)$.

4.2 Instantiated Problem

We are given a list of tasks $\{\tau_i\}_{i\in\mathcal{I}}$ with

$$\tau_i = (O_i, T_i, D_i, C_i).$$

The problem is: is the system schedulable?

- Model M: A TA per task and a TA for the scheduler
- Input I: $\{O_i, T_i, D_i, C_i\}_{i\in I}$
- Output: Yes or No

4.3 Robustness Analysis

The problem is parametrized typically by letting the deadlines D_is instantiated and by parametrizing some of the C_is, O_is and T_is.

- Model M': a PTA deduced from M by parametrizing the values of interest
- Input: I and an identified set of parameters
- Output: a constraint K such that for every $\pi \models K$, the π-instantiated model $M'[\pi]$ is schedulable.

5 Premptive Job-Shop Scheduling with multiple machines [1]

5.1 Context

The Job-shop scheduling problem is a generic resource allocation problem in which common resources ("machines") are required at various time points (and for given duration) by different tasks. The goal is to find a way to allocate the resources such that all the tasks terminate as soon as possible (or "minimal makespan" in the scheduling jargon). We consider a fixed set M of resources. A *step* is a pair (m, d) where $m \in M$ and $d \in \mathcal{N}$, indicating the required utilization of resource m for time duration d. A *job specification* is a finite sequence

$$J = (m_1, d_1), (m_2, d_2), \cdots, (m_k, d_k)$$

of steps stating that in order to accomplish job J, one needs to use a machine m_1 for d_1 time, then use machine m_2 for d_2 time etc. For details, see [1].

5.2 Instantiated Problem

We are given a list of job specifications $\{J_i\}_{i\in\mathcal{I}}$ where

$$J_i = \{(m_{i,j}, d_{i,j})\}_{j\in J(i)}.$$

The problem is: What is the minimal makespan to complete all the chain of tasks?

- Model M: A product of TAs, with a TA per job
- Input I: $\{(m_{i,j}, d_{i,j})_{i\in I(j)}\}_{j\in J}$
- Output: μ = minimal makespan

5.3 Robustness Analysis

The problem is parametrized typically by parametrizing some of the d_is.

- Model M': a PTA deduced from M by parametrizing the values of interest and composed with a TA that tests wether or not the last task is completed within μ (using an ok and an error state)
- Input: I together with an identified set of parameters and the value of the computed makespan μ
- Output: a constraint K such that for every $\pi \models K$, there is a schedule which allows to complete the last task of the π-instantiated model $M'[\pi]$ within μ.

6 Schedulability Analysis with Behavioral Cartography

When one is only interested in finding all the parameters valuation such that the system is schedulable, without considering a particular schedule, it is interesting to consider the Behavioral Cartography Method included in IMITATOR. By iterating IM over all the integers points inside a finite but dense rectangle V_0 in the parametric space, one is able to decompose (most of) the parametric space included into V_0 into behavioral tiles[2] [2].

[2] Actually, one can take a finer discretization step than integers to ensure a better coverability of V_0

For a given tile, we ensure that for every valuation of the parameters in this tile, the behavior of $\mathcal{A}$ is the same; therefore, we only have to test the schedulability of a single point for each tile to ensure the schedulability on the whole tile.

Let us apply this method on a "Rate monotonic" example of [3, Section III]. There are three periodic tasks τ_1, τ_2 and τ_3 with periods of $T_1 = 3$, $T_2 = 8$ and $T_3 = 20$ and deadlines of $D_1 = 3$, $D_2 = 8$ and $D_3 = 20$. We are interested in finding the set of computation times of each task such that the system is schedulable. (The interested reader can find the full details in [3].)

We set V_0 as $C_1 \in [0, 3]$, $C_2 \in [0, 8]$ and $C_3 \in [0, 20]$, where C_i is the computational time of τ_i. The system is unschedulable if there exists a task τ_i such that $C_i > T_i$. Algorithm *BC* outputs the set of tiles and we check for one point of each tile whether the system is schedulable or not. The result for this example is given in Figures 3, 4, 5 for this V_0 with a discretization step of 0.2.

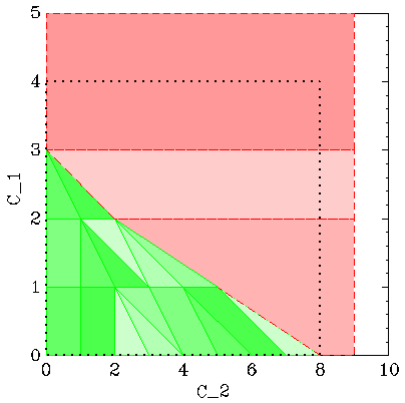

Figure 3: Cartography output by IMITATOR 2.5 (in green the system is schedulable; in red, unschedulable)

7 Results

The examples have been executed with the following commands:

- For [AM02], [SGL97], [CPR08], [HPPR10]:
  IMITATOR case_study.imi case_study.pi0 -merge
- For [LA02] 2×5, [LA02] 3×5:
  IMITATOR case_study.imi case_study.pi0 -merge -inclusion
- For the cartography of [BB04]:
  IMITATOR case_study.imi case_study.v0 -mode cover -merge -step 0.2

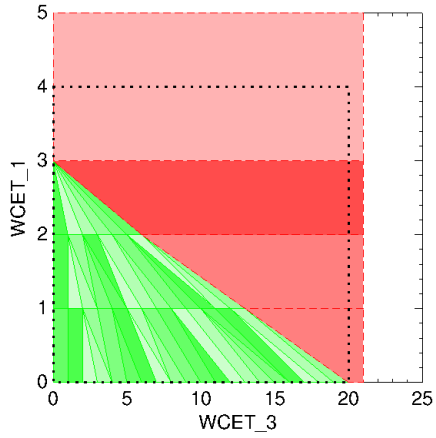

Figure 4: Cartography output by IMITATOR 2.5 (in green the system is schedulable; in red, unschedulable)

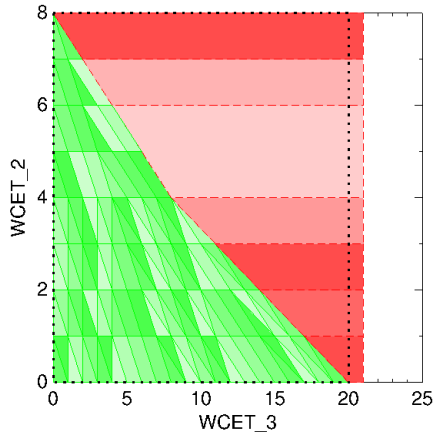

Figure 5: Cartography output by IMITATOR 2.5 (in green the system is schedulable; in red, unschedulable)

References

[1] Yasmina Abdeddaïm and Oded Maler. Preemptive job-shop scheduling using stopwatch automata. In *TACAS*, pages 113–126, 2002.

[2] É. André and L. Fribourg. Behavioral cartography of timed automata. In *RP*, volume 6227 of *LNCS*, pages 76–90. Springer, 2010.

[3] Enrico Bini and Giorgio C. Buttazzo. Schedulability analysis of periodic fixed priority systems. *IEEE Trans. Computers*, 53(11):1462–1473, 2004.

Case Study	$\lvert\mathcal{A}\rvert$	$\lvert\mathcal{X}\rvert$	$\lvert\mathcal{P}\rvert$	$\lvert s\rvert$	$\lvert T\rvert$	n	$\lvert\mathcal{K}\rvert$	t (s)
[AM02]	3	3	4	53	70	10	5	0.45
[LA02] 2×5	3	3	11	371	528	21	10	63.4
[LA02] 3×5	4	3	16	4903	9043	30	5	160.6
[SGL97]	8	15	18	215	264	15	17	85.3
[CPR08]	4	6	8	676	886	15	15	288.3
[HPPR10]	3	4	9	60	103	10	7	2.1
[BB04]	5	7	10	-	-	-	-	66

Figure 6: Results for the schedulability problem

[4] A. Cimatti, L. Palopoli, and Y. Ramadian. Symbolic computation of schedulability regions using parametric timed automata. In *RTSS*, pages 80–89, Washington, DC, USA, 2008. IEEE Computer Society.

[5] Thi Le, Luigi Palopoli, Roberto Passerone, Yusi Ramadian, and Alessandro Cimatti. Parametric analysis of distributed firm real-time systems: A case study. In *ETFA*, pages 1–8, 2010.

[6] Jun Sun, Mark K. Gardner, and Jane W. S. Liu. Bounding completion times of jobs with arbitrary release times, variable execution times, and resource sharing. *IEEE Trans. Softw. Eng.*, 23:603–615, 1997.