On the Existence of a Cyclic Schedule for Non-Preemptive Periodic Tasks with Release Offset

Mitra Nasri
Max Planck Institute for Software Systems, Germany.
mitra@mpi-sws.org

Emmanuel Grolleau
ENSMA Chasseneuil, France.
grolleau@ensma.fr

I. Introduction

Non-preemptive scheduling reduces both runtime and design time overhead since it avoids context switches, preserves cache affinity, and simplifies the use of shared resources. As a result, it increases the accuracy of worst-case execution time (WCET) estimation and improves system predictability. Many safety critical systems use table driven scheduling in which tasks are dispatched according to a schedule stored in a table. This table contains a cyclic schedule that needs to be repeated during the lifetime of the system. Since the table must be kept in memory, a smaller table is preferable particularly for embedded systems with limited memory. For example, the Atmel UC3A0512 microcontroller, which is used in mission critical space applications [1], has 64 KiB of internal SRAM, 512 KiB of internal flash memory, and is clocked at 12 MHz. Similarly, an Arm Cortex-M3 has 4 KiB RAM, 16 KiB flash, and clock speed 24 MHz. With such limited resources, non-preemptive execution becomes the natural way of executing real-time tasks in these systems [2].

In a non-preemptive scheduling table, the number of entries is analogous with the number of instances of the tasks (a.k.a jobs) in a cyclic schedule. It is known that in a synchronous periodic task set with constrained deadlines, the schedule becomes cyclic after H units of time where H is the least-common-multiple (LCM) of periods [3] for any feasible schedule. However, many systems use release offsets for their tasks in order to enforce particular properties such as avoiding the critical instant, or enforcing precedence constraints. In that case, a cyclic schedule may not be found in the first hyperperiod, which then increases the size of scheduling table. In our work, we are interested in the length of a cyclic schedule since many embedded systems have limited amount of memory to store the table.

Choquet-Geniet et al. [4] have shown that any preemptive periodic task set with independent tasks and constrained deadlines that is feasible in a work-conserving schedule, satisfies the following conditions (we call it after the authors' names *Choquet Grolleau Conditions* (CGC)):

- **(i)** it becomes cyclic after time $t^{start} \in [0, O^{\max} + H]$ where $O^{\max}$ is the largest release offset,
- **(ii)** the length of cycle is H,
- **(iii)** the cycle is the first window of length H in the schedule having a slack exactly equal to $(1-U)\cdot H$, where U is the task set utilization,
- **(iv)** there is no deadline miss from time 0 to the end of the cycle.
- **(v)** every job appearing in the cycle starts and ends in the cycle.

Goossens et al. [5] have extended [4] and derived an upper bound on the length of the cycle for any feasible schedule (work-conserving or not, uni or multiprocessor). However, this upper bound is a function of task's periods, deadlines, and release offsets and may become exponential with respect to H.

In this work, we study the existence of a cyclic schedule for non-preemptive tasks when they are scheduled by non-work-conserving scheduling algorithms. In the remainder of the paper we consider a set of non-preemptive periodic tasks denoted by $\tau = \{\tau_1, \tau_2, \ldots, \tau_n\}$, where each task τ_i is identified by its execution time C_i, its period T_i, its deadline $D_i \leq T_i$, and its release offset O_i. We denote the least-common-multiple (LCM) of the periods by H and utilization of the task set by $U = \sum_{i=1}^{n} C_i/T_i$. Moreover, it is assumed that the system has a dispatcher to dispatch the jobs one after another according to the start times stored in the table, hence, the schedulability of the task set will not be affected by runtime release jitters or execution time variations.

In Sec. II, by an example we show that there exists online non-work-conserving scheduling algorithms that create a cycle with length $2H$ (i.e., they violate condition (ii) of CGC). We also show that it is possible to modify our example schedule such that it satisfies condition (ii) of CGC although it will no longer be consistent with the scheduling algorithm. In Sec. III, we formally introduce the open problems.

II. Motivating Examples

Precautious-RM is one of the recently introduced online non-work-conserving scheduling algorithms for non-preemptive periodic tasks [6]. It priorities the tasks according to the rate monotonic (RM) policy, however, before scheduling a task such as τ_i at time t, Precautious-RM verifies the following condition: $t + C_i \leq r_1^{next} + D_1 - C_1$, where r_1^{next} is the expected release

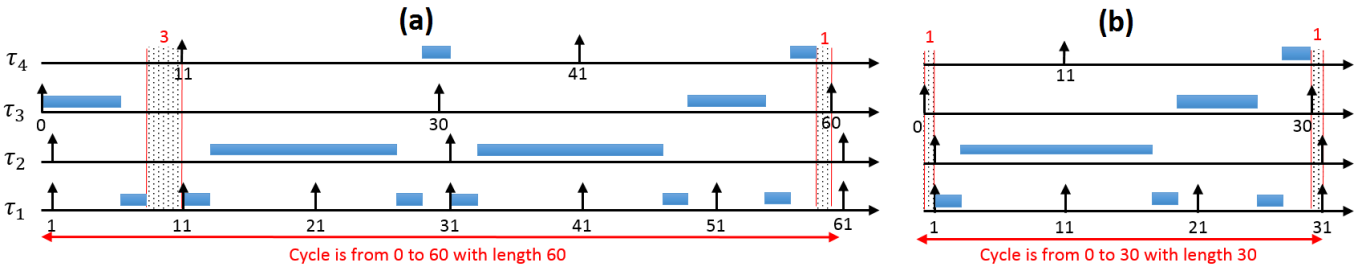

Fig. 1. A task set with 4 tasks scheduled by (a) Precautious-RM [6] and (b) an optimal schedule. In this task set $C_1 = 2$, $C_2 = 14$, $C_3 = 6$, $C_4 = 2$, $T_1 = 10$, $T_2 = T_3 = T_4 = 30$, $O_1 = O_2 = O_4 = 1$, and $O_3 = 0$. Here, H is 30 and $U = \frac{28}{30}$.

time of the next job of τ_1. If the condition holds, it schedules τ_i, otherwise it schedules an idle-interval from t to r_1^{next}. In other words, it schedules a low priority task only if it will not cause a deadline miss for the next job of τ_1. If Precautious-RM is used to create a non-preemptive schedule for the following task set, the resulting schedule will have a cycle with length $2H$ as it is shown in Fig. 1.

As it can be seen in Fig. 1-(a), Precautious-RM has inserted an idle-time from time 8 to 11 because if τ_2 was executed at 8, it would have caused a deadline miss for the second job of τ_1. Even though the length of cycle is $2H$ in Fig. 1-(a), the total amount of slack is 4 which is equal to $(1 - U) \cdot 2H$. In Fig. 1-(b), an optimal schedule has been presented that schedules the task set in a cycle with length H. This schedule satisfies all CGC.

III. Open Problems

The first open problem is about the existence of a task set for which none of the feasible schedules satisfies CGC. The existence of such task set implies that memory required for storing an offline table of a set of non-preemptive tasks with release offsets might become unbounded or unreasonably large. Moreover, Since the problem of scheduling a set of non-preemptive periodic tasks is known to be strongly NP-Hard (see Jeffay et al. [7]), not knowing the existence of a cycle with a limited length will only make the situation worse. It means that in order to find a schedule, a large number of jobs must be considered (if a cycle exists).

Problem 1. Find a feasible task set τ for which there does not exist a cyclic schedule or the length of cycle is larger than H.

If there exists a task set that matches with Problem 1, then the next question is how to find a cyclic schedule such that the number of cycles is minimized. Otherwise, if there is no such task set, then we will be interested to find the CGC-compatible schedule. Next we define these two follow-up problems.

Problem 2. Given a feasible non-preemptive task set τ, find a schedule S that has the smallest cycle length and satisfies condition (iv) of CGC.

Problem 3. Given a feasible non-preemptive task set τ, find a schedule S that satisfies CGC.

It is worth noting that we still do not know whether an answer to Problem 2 satisfies condition (i) of CGC or not. This answer will most probably satisfy condition (iii) because that is the notion of reaching to an equilibrium state.

Acknowledgment

This work is supported by a fellowship from Alexander von Humboldt Foundation. The authors would like to thank Schloss Dagstuhl for seminar number 17131 that helped this work to happen.

References

[1] J. F. Ruiz, "GNAT Pro for On-board Mission-Critical Space Applications," in *Ada-Europe International Conference on Reliable Software Technologies*, 2005, pp. 248–259.

[2] B. Nasri, Mitra andBrandenburg, "Offline Equivalence: A Non-Preemptive Scheduling Technique for Resource-Constrained Embedded Real-Time Systems," in *IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS)*, 2017.

[3] J. Y.-T. Leung and M. Merrill, "A note on preemptive scheduling of periodic, real-time tasks," *Information processing letters*, vol. 11, no. 3, pp. 115–118, 1980.

[4] A. Choquet-Geniet and E. Grolleau, "Minimal schedulability interval for real-time systems of periodic tasks with offsets," *Theoretical computer science*, vol. 310, no. 1-3, pp. 117–134, 2004.

[5] J. Goossens, E. Grolleau, and L. Cucu-Grosjean, "Periodicity of real-time schedules for dependent periodic tasks on identical multiprocessor platforms," *Real-Time Systems*, vol. 52, no. 6, pp. 808–832, 2016.

[6] M. Nasri, G. Nelissen, and G. Fohler, "A new approach for limited preemptive scheduling in systems with preemption overhead," in *Euromicro Conference on Real-Time Systems (ECRTS)*, 2016, pp. 23–35.

[7] K. Jeffay, D. F. Stanat, and C. U. Martel, "On non-preemptive scheduling of period and sporadic tasks," in *IEEE Real-Time Systems Symposium (RTSS)*, 1991, pp. 129–139.