# Complexity of partitioned scheduling for periodic tasks

Pontus Ekberg (Speaker) * Sanjoy Baruah †

## 1 Introduction

We consider the partitioned scheduling of a set of periodic tasks $\mathcal{T} = \{\tau_1, \ldots, \tau_n\}$ on $m$ identical processors. A periodic task $\tau_i = (o_i, c_i, d_i, p_i) \in \mathbb{N}^4$ releases a job at each time point $o_i + kp_i$ for $k \in \mathbb{N}$. Every job released by $\tau_i$ has an execution time requirement of $c_i$ time units and a deadline $d_i$ time units after its release.

We say that $\mathcal{T}$ is *synchronous* if $o_i = 0$ for all $\tau_i$ (and *asynchronous* otherwise). We say that $\mathcal{T}$ has *implicit deadlines* if $d_i = p_i$ for all $\tau_i$ and has *constrained deadlines* if $d_i \leqslant p_i$ for all $\tau_i$ (and has *arbitrary deadlines* otherwise).

We want to understand the complexity of the following family of decision problems.

**Definition 1** (PARTITIONED $\mathcal{A}$-SCHEDULABILITY)**.**
Instance: $\langle \mathcal{T}, m \rangle$*, where* $\mathcal{T}$ *is a set of periodic tasks and* $m$ *is the number of processors.*

Question: *Is there an* $m$*-partitioning* $\mathcal{T}_1, \ldots, \mathcal{T}_m$ *of* $\mathcal{T}$ *such that each partition* $\mathcal{T}_i$ *is schedulable by scheduling algorithm* $\mathcal{A}$ *on a single processor with no deadline misses?*

Partitioned scheduling is a very commonly used strategy for real-time systems. It reduces overhead costs by disallowing migrations between processors and importantly allows the reuse of mature results for single-processor scheduling in a multiprocessor setting.

It is not difficult to see that PARTITIONED $\mathcal{A}$-SCHEDULABILITY generalizes BIN PACKING in most settings and is NP-hard, but other previously known lower bounds on its complexity are only the ones for the corresponding single-processor $\mathcal{A}$-schedulability problems. We want to narrow down the complexity, which among other things lets us find a distinction between the partitioned schedulability problems can efficiently be formulated as ILP or SAT instances, and which cannot unless the polynomial hierarchy collapses. To find better bounds we first define a partitioned version of a helpful number-theoretic decision problem that has been

*pontus.ekberg@it.uu.se. Department of Information Technology, Uppsala University, P.O. Box 337, SE-751 05 Uppsala, Sweden.

†baruah@wustl.edu. Department of Computer Science & Engineering, MSC 1045-213-1010J, Washington University in Saint Louis, 1 Brookings Drive, St. Louis, MO 63130-4899.

used for several hardness results for periodic tasks in single processor systems, the Simultaneous Congruences Problem.

## 2 Partitioned simultaneous congruences

The Simultaneous Congruences Problem (SCP) concerns finding a set of congruence classes that all share some element. The SCP was shown to be NP-complete by Leung and Whitehead [3] (and was later shown to be so in the strong sense by Baruah et al. [1]).

**Definition 2** (SCP)**.**
Instance: $\langle A, k\rangle$*, where* $A = \{(a_1, b_1), \ldots, (a_n, b_n)\}$ *is a set of pairs of non-negative integers satisfying* $a_i < b_i$ *for each* $i$*, and* $k$ *is a positive integer.*
Question: *Is there an integer* $x$ *and an* $A' \subseteq A$ *such that* $|A'| = k$ *and*

$$x \equiv a_i \pmod{b_i}$$

*for each* $(a_i, b_i) \in A'$*?*

We define PARTITIONED SCP as a natural generalization to the SCP.

**Definition 3** (PARTITIONED SCP)**.**
Instance: $\langle A, k, m\rangle$*, where* $A = \{(a_1, b_1), \ldots, (a_n, b_n)\}$ *is a set of pairs of non-negative integers satisfying* $a_i < b_i$ *for each* $i$*, and* $k$ *and* $m$ *are positive integers.*
Question: *Is there an* $m$*-partitioning* $A_1, \ldots, A_m$ *of* $A$ *such that* $\langle A_i, k\rangle \notin$ SCP *for each partition* $A_i$*?*

Through a reduction from GENERALIZED GRAPH COLORING [4] we can pinpoint the complexity of PARTITIONED SCP. The reduction is closely inspired by Leung and Whitehead's reduction from CLIQUE to SCP [3] and works by assigning congruences classes to vertices such that a set of congruences classes have a joint element if and only if the corresponding set of vertices form a clique.

**Theorem 4.** PARTITIONED SCP *is* $\Sigma_2^{\mathsf{P}}$*-complete, even when* $m = 2$*.*

## 3 Asynchronous tasks

For asynchronous periodic tasks we find a new general lower bound by reducing from PARTITIONED SCP in a fairly straight-forward manner.

**Theorem 5.** *The partitioned* $\mathcal{A}$*-schedulability problem for asynchronous periodic tasks with constrained (or arbitrary) deadlines is* $\Sigma_2^{\mathsf{P}}$*-hard for any work-conserving scheduler* $\mathcal{A}$ *(preemptive or non-preemptive), even when restricted to* $m = 2$ *processors and task sets with* $\sum_{\tau_i \in \mathcal{T}} \frac{c_i}{p_i} < \varphi$ *for any constant* $\varphi > 0$*.*

When $\mathcal{A}$ is an optimal preemptive single-processor scheduling algorithm, such as Earliest Deadline First (EDF), this can be shown to be a tight bound.

**Theorem 6.** *The partitioned EDF-schedulability problem for asynchronous periodic tasks with constrained (or arbitrary) deadlines is* $\Sigma_2^{\mathsf{P}}$*-complete.*

In contrast, for preemptive Fixed Task Priority (FP) scheduling we still have a gap to the lower bound of Theorem 5.

**Theorem 7.** *The partitioned FP-schedulability problem for asynchronous periodic tasks is in* $\Sigma_3^{\mathsf{P}}$*. This holds both if we are given the priority ordering or are asked to find one.*

The partitioned EDF-schedulability problem with implicit deadlines is essentially just plain BIN PACKING and is NP-complete. However, FP-scheduling with implicit deadlines can be shown to be harder than EDF when we are given a predefined priority ordering.

**Theorem 8.** *The partitioned FP-schedulability problem for asynchronous periodic tasks is* $\Sigma_2^{\mathsf{P}}$*-hard if we are given a priority ordering, even when restricted to implicit-deadline tasks,* $m = 2$ *processors and task sets with* $\sum_{\tau_i \in \mathcal{T}} \frac{c_i}{p_i} < \varphi m$ *for any constant* $\varphi > 1/2$*.*

It is an open problem if this remains $\Sigma_2^{\mathsf{P}}$-hard if we are instead asked if the set of tasks is FP-schedulable with *any* priority ordering.

# 4 Synchronous tasks

There is relatively less to say about synchronous periodic tasks at this time, but we have the following observations.

**Theorem 9.** *With synchronous periodic tasks, partitioned EDF-schedulability with implicit deadlines and partitioned FP-schedulability with implicit or constrained deadlines are* NP*-complete.*

The above follows directly from observing that the corresponding single processor problems are in NP. However, EDF-schedulability with constrained or arbitrary deadlines is not contained in the first level of the polynomial hierarchy (unless it collapses), but is contained in the second.

**Theorem 10.** *Partitioned EDF-schedulability of synchronous periodic tasks with constrained or arbitrary deadlines is both* NP- *and* coNP*-hard, and is in* $\Sigma_2^{\mathsf{P}}$*.*

FP-schedulability with arbitrary deadlines has a larger gap still. (Here even the single processor case is not yet pinpointed!)

**Theorem 11.** *Partitioned FP-schedulability of synchronous periodic tasks with arbitrary deadlines is* NP*-hard, and is in* $\Sigma_3^{\mathsf{P}}$*.*

We do suspect that the partitioned EDF-schedulability problem with constrained or arbitrary deadlines (as in Theorem 10) is in fact $\Sigma_2^{\mathsf{P}}$-complete, just as with asynchronous tasks, but have yet been unable to show this. It seems significantly more difficult to construct a reduction from PARTITIONED SCP here due to the lack of the tasks' offset parameters $o_i$, which in many ways mirror the $a_i$ parameters in PARTITIONED SCP. Still, such a reduction has been done from SCP to the single processor synchronous case [2], so there is still hope that PARTITIONED SCP can help in resolving this important outstanding problem.

## References


[1] S. Baruah, R. Howell, and L. Rosier, *Algorithms and complexity concerning the preemptive scheduling of periodic, real-time tasks on one processor*, Real-Time Systems: The International Journal of Time-Critical Computing, 2 (1990), pp. 301–324.

[2] P. Ekberg and W. Yi, *Uniprocessor feasibility of sporadic tasks with constrained deadlines is strongly coNP-complete*, in The 27th Euromicro Conference on Real-Time Systems, 2015, pp. 281–286.

[3] J. Y.-T. Leung and J. Whitehead, *On the complexity of fixed-priority scheduling of periodic, real-time tasks*, Performance Evaluation, 2 (1982), pp. 237–250.

[4] V. Rutenburg, *Complexity of generalized graph coloring*, in Mathematical Foundations of Computer Science, Springer Berlin Heidelberg, 1986, pp. 573–581.