Uniprocessor Feasibility of Sporadic Tasks with Constrained Deadlines is Strongly coNP-complete

Pontus Ekberg and Wang Yi
Uppsala University, Sweden
Email: {pontus.ekberg | yi}@it.uu.se

***Abstract*—Deciding the feasibility of a sporadic task system on a preemptive uniprocessor is a central problem in real-time scheduling theory. The computational complexity of this problem has been a long-standing open question. We show that it is coNP-complete in the strong sense, even when deadlines are constrained. This is achieved by means of a pseudo-polynomial transformation from the strongly NP-hard Simultaneous Congruences Problem to the complement of the feasibility problem.**

I. Introduction

We let a sporadic task system be defined as a finite multiset T of *tasks*, where each task is a triple $(e, d, p) \in \mathbb{N}^3_+$, representing its worst-case execution time, relative deadline and minimum inter-arrival separation (or period), respectively.

A sporadic task *generates* potentially unbounded sequences of *jobs*. A job is an instance of the task's workload, characterized by a release time, an execution time and an absolute deadline. A job from task (e, d, p) has execution time not larger than e time units and absolute deadline exactly d time units after its release time. Release times of two consecutive jobs from (e, d, p) are separated by at least p time units. A sporadic task system T generates any interleaving of job sequences that can be generated by each of the tasks $(e, d, p) \in \mathsf{T}$.

For a sequence of jobs to be successfully scheduled, every job must be executed for a total duration equal to its execution time, between its release time and its absolute deadline. In the following we assume a preemptive uniprocessor, meaning that only one job can be executed at a time, but a job can be paused and resumed at a later time at no additional cost.

Definition I.1 (Feasibility)**.** *A task system* T *is* feasible *if and only if there is some scheduling algorithm that will successfully schedule all job sequences that can be generated by* T*.* □

A task system T is said to have *implicit deadlines* if $d = p$ for all $(e, d, p) \in \mathsf{T}$, and said to have *constrained deadlines* if $d \leqslant p$ for all $(e, d, p) \in \mathsf{T}$. The utilization $\mathrm{U}(\mathsf{T})$ of a task system T is defined as $\mathrm{U}(\mathsf{T}) \stackrel{\text{def}}{=} \sum_{(e,d,p)\in\mathsf{T}} e/p$.

Liu and Layland [1] showed that a task system T with implicit deadlines is feasible if and only if $\mathrm{U}(\mathsf{T}) \leqslant 1$, but this is not a sufficient condition with constrained deadlines. Dertouzos [2] showed that Earliest Deadline First (EDF) is an optimal scheduling algorithm on preemptive uniprocessors, which means that the terms "feasibility" and "EDF-schedulability" can be used interchangeably here.

Theorem I.2 ([2])**.** *A task system* T *is feasible if and only if it is EDF-schedulable.* □

A related workload model is that of (strictly) periodic tasks, where each task is a quadruple $(s, e, d, p) \in \mathbb{N} \times \mathbb{N}^3_+$. The difference to the sporadic task model is that the release times of two consecutive jobs from task (s, e, d, p) must be exactly p time units apart, and the release time of the first job is fixed at time point s. A periodic task system is *synchronous* if $s = 0$ for all tasks (s, e, d, p), and *asynchronous* otherwise. It is known [3] that feasibility testing of sporadic tasks is equally hard as that of synchronous periodic tasks, which means that the terms "sporadic" and "synchronous periodic" can be used interchangeably for the results in this paper as well.

Theorem I.3 ([3])**.** *A sporadic task system* T *is feasible if and only if the synchronous periodic task system* $\{(0, e, d, p) \mid (e, d, p) \in \mathsf{T}\}$ *is feasible.* □

A classic result is that the feasibility problem for asynchronous periodic task systems with constrained deadlines is strongly coNP-complete [4], [5]. Baruah et al. [5], [3] also developed a pseudo-polynomial time algorithm for the special case of synchronous periodic (or sporadic) task systems with utilization *a priori* bounded by some constant $c < 1$. However, the complexity of the general synchronous case remained open for many years. It was listed as one of five "open problems" in real-time scheduling by Baruah and Pruhs [6]. Shortly thereafter it was partially resolved by Eisenbrand and Rothvoß [7] who showed it to be weakly coNP-hard, but the question remained if it allowed a pseudo-polynomial time solution like the special case with bounded utilization. Eisenbrand and Rothvoß conjectured that it did, but we show that the feasibility problem for synchronous periodic task systems with constrained deadlines is strongly coNP-complete, and thus that it can have no pseudo-polynomial time solution unless $\mathsf{P} = \mathsf{NP}$. Figure 1 summarizes the current knowledge.

	General case	Utilization bounded by a constant $c < 1$
Asynchronous periodic tasks	coNP-complete in the strong sense. [5]	coNP-complete in the strong sense. [5]
Synchronous periodic tasks (or sporadic)	coNP-complete in the strong sense (from this work).	There exists a pseudo-polynomial algorithm. [5], [3]

Fig. 1. The current knowledge on the feasibility problem of periodic tasks with constrained deadlines on preemptive uniprocessors.

II. Preliminaries

A. The Simultaneous Congruences Problem

The *Simultaneous Congruences Problem* (SCP) is a number-theoretic decision problem that has been used to establish several complexity results in real-time scheduling theory.

Definition II.1 (The Simultaneous Congruences Problem)**.** *An instance is defined by a multiset* $A = \{(a_1, b_1), \ldots, (a_n, b_n)\}$ *and an integer* k*, such that* $2 \leqslant k \leqslant n$ *and* $(a_i, b_i) \in \mathbb{N} \times \mathbb{N}_+$ *for all* $i \in \{1, \ldots, n\}$.

The simultaneous congruences problem asks whether there exists a subset $A' \subseteq A$ *of at least* k *elements and an* $x \in \mathbb{N}$*, such that*

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

for all $(a_i, b_i) \in A'$.

In the remainder of this paper we assume, without loss of generality, that $a_i < b_i$ *for all* $(a_i, b_i) \in A$. □

SCP was first shown to be weakly NP-complete by Leung and Whitehead [8] via a reduction from CLIQUE, and then used by them to show various hardness results concerning fixed-priority scheduling. Leung and Merrill [4] reduced SCP to the complement of the asynchronous periodic feasibility problem on uniprocessors, thus showing that problem to be weakly coNP-hard. Baruah et al. [5] later showed that SCP is in fact strongly NP-complete via an alternative reduction from 3-SAT, which then implied the strong coNP-hardness of the asynchronous periodic feasibility problem.

Theorem II.2 ([5])**.** SCP *is strongly* NP*-complete.* □

B. The Theory of Demand Bound Functions

The hardness proof we present in the next section relies heavily on demand bound functions, and in particular the following, well-known theorem due to Baruah et al.

Theorem II.3 ([3], [5])**.** *A sporadic task system* T *is feasible on a preemptive uniprocessor if and only if* $\mathrm{U}(\mathsf{T}) \leqslant 1$ *and*

$$\forall \ell \geqslant 0, \quad \mathrm{dbf}(\mathsf{T}, \ell) \leqslant \ell, \tag{1}$$

where

$$\mathrm{dbf}(\mathsf{T}, \ell) \stackrel{\text{def}}{=} \sum_{(e,d,p) \in \mathsf{T}} \left(\left\lfloor \frac{\ell - d}{p} \right\rfloor + 1 \right) \cdot e \tag{2}$$

is the demand bound function *of* T *in interval lengths* ℓ. □

As a notational convenience, let $\mathrm{dbf}(\tau, \ell) \stackrel{\text{def}}{=} \mathrm{dbf}(\{\tau\}, \ell)$ for any task $\tau = (e, d, p)$.

One of the things that Baruah et al. [3] show using this theorem is that the sporadic feasibility problem is in coNP. A witness to the infeasibility of a task system is simply an $\ell \geqslant 0$ such that the formula in Eq. (1) is false (if there exists a witness, we are guaranteed that there is also one in $\mathbb{N}$ representable with polynomially many bits). A similar argument can be made for asynchronous periodic tasks.

Theorem II.4 ([3], [5])**.** *The feasibility problem for periodic tasks, both synchronous and asynchronous, is in* coNP. □

III. The Hardness of Sporadic Feasibility

Here we will show the strong coNP-hardness of the feasibility problem for sporadic tasks on preemptive uniprocessors. This is achieved by means of a pseudo-polynomial transformation (as defined by Garey and Johnson [9]) from SCP to the complement of the feasibility problem.

A. Overview of the Transformation

First we describe the intuition behind the transformation. In Figure 2 we have marked along a number line the $x \in \mathbb{N}$ such that $x \equiv a_i \pmod{b_i}$, for four example pairs (a_i, b_i).

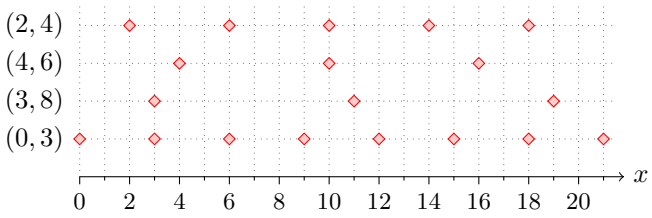

Fig. 2. Congruence classes a_i modulo b_i for the four different pairs $(a_i, b_i) \in A$, where $A = \{(2,4), (4,6), (3,8), (0,3)\}$. It is clear from the figure that there are several $x \in \mathbb{N}$ belonging to two congruence classes simultaneously, but it can be shown that there is no x belonging to three. Thus, $(A, 2)$ is a *yes*-instance and $(A, 3)$ is a *no*-instance of SCP.

We would like to somehow match the structure of these congruence classes with demand bound functions. For each pair (a_i, b_i) we want to create a demand bound function (in interval lengths ℓ) that is highly regular, but has "hard points" of slightly increased demand at those ℓ that in some given way are related to the congruence class of a_i modulo b_i. If we have two such functions, their hard points should align at exactly those ℓ related to both congruences classes. Figure 3 illustrates this basic idea. The goal is to take any instance (A, k) of SCP and create $|A|$ such functions that, when summed, result in a violation of Eq. (1) at some ℓ if and only if at least k of them align such hard points at ℓ.

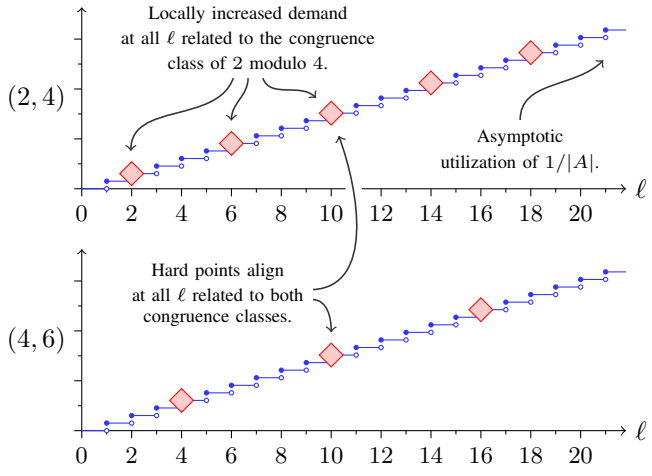

Fig. 3. Conceptual demand bound functions corresponding to the two pairs $(2, 4)$ and $(4, 6)$ from A. The marked areas are where we want slightly increased demand. Note that these functions do not match exactly those that we will get from the transformation.

B. Encoding into Task Systems

We now show how the high-level idea for a transformation that was presented in the last section is encoded with actual task systems. It does not appear possible to capture the structure of a congruence class in a way that achieves our goals using only a single sporadic task. We can, however, create a set of tasks that have the sought structure in its (joint) demand bound function. The details of this follow.

Definition III.1 (Transformation from SCP to (in-)feasibility)**.** *The transformation takes an arbitrary instance* (A, k) *of* SCP*, where* $A = \{(a_1, b_1), \ldots, (a_n, b_n)\}$*, and produces a sporadic task system* $\mathsf{T}_{(A,k)}$*.*

For each $(a_i, b_i) \in A$*, we create the following set of* b_i *constrained-deadline sporadic tasks:*

$$\mathsf{T}_{(a_i,b_i)} \stackrel{\text{def}}{=} \left\{ \tau^y_{(a_i,b_i)} \mid y \in \{1, \ldots, b_i\} \right\},$$

where

$$\tau^y_{(a_i,b_i)} \stackrel{\text{def}}{=} \begin{cases} (1, a_i n + k - 1, b_i n), & \text{if } y = a_i + 1, \\ (1, yn, b_i n), & \text{otherwise.} \end{cases} \tag{3}$$

The multiset

$$\mathsf{T}_{(A,k)} \stackrel{\text{def}}{=} \biguplus_{(a_i,b_i)\in A} \mathsf{T}_{(a_i,b_i)} \tag{4}$$

is the produced task system. □

Each task set $\mathsf{T}_{(a_i,b_i)}$ has a highly regular demand bound function, where the hard points are encoded in the slightly shorter deadline of the task $\tau^{a_i+1}_{(a_i,b_i)}$. Figure 4 shows the demand bound function of a generated set of tasks $\mathsf{T}_{(a_i,b_i)}$, corresponding to some $(a_i, b_i) \in A$. It can be noted that it has the same general structure as the conceptual functions shown in Figure 3.

Note that the number of tasks in $\mathsf{T}_{(A,k)}$, as well as the values of their parameters, are bounded by some two-variable polynomial in the size and the maximum numerical value found in the corresponding SCP instance (A, k). Also, the transformation can trivially be computed in time bounded by such a polynomial. To establish that the above is a valid pseudo-polynomial transformation, what remains to be shown is that $\mathsf{T}_{(A,k)}$ is a *no*-instance of the feasibility problem if and only if (A, k) is a *yes*-instance of SCP. We show this in the next section.

C. Correctness of the Transformation

Before showing that the transformation in Definition III.1 is correct, we need to prove two auxiliary lemmas about the characteristics of the demand bound functions of the generated task systems. The first of the lemmas is about the identity property of the demand bound functions at all points $\ell \in \{0, n, 2n, 3n, \ldots\}$.

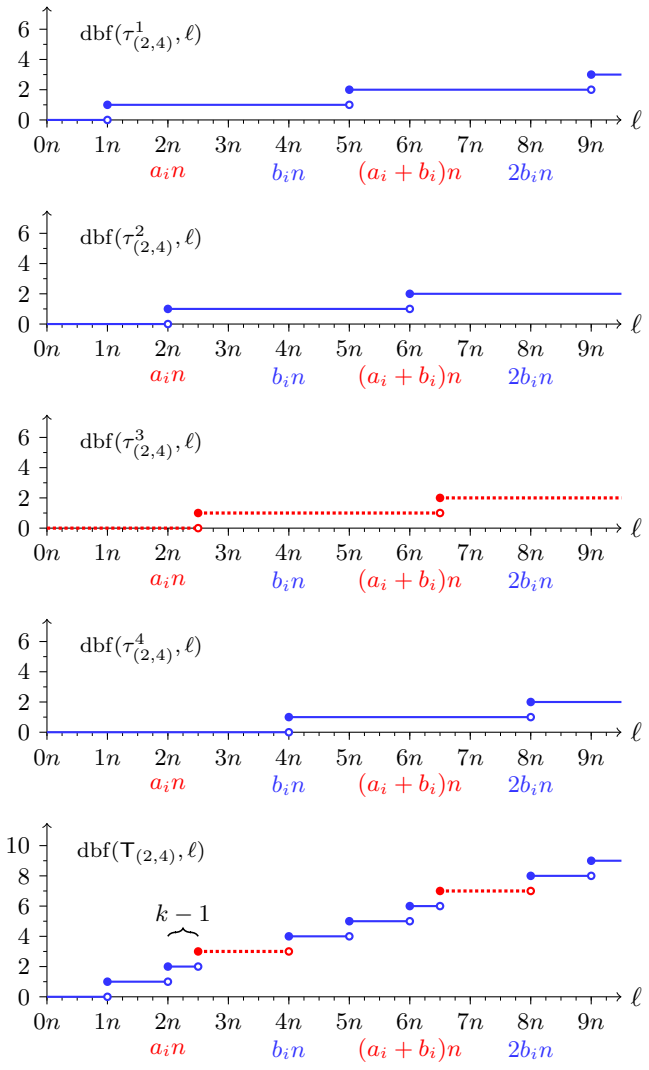

Fig. 4. The demand bound functions for the tasks in $\mathsf{T}_{(2,4)}$, generated from an SCP instance (A, k) where $|A| = n = 4$ and $k = 3$. The top four functions are the demand bound functions for the individual tasks. The dotted function is for the task $\tau^3_{(2,4)}$, which is defined by the special case in Eq. (3). At the bottom is the sum of their demand bound functions, $\mathrm{dbf}(\mathsf{T}_{(2,4)}, \ell)$. Note that $\mathrm{dbf}(\mathsf{T}_{(2,4)}, \ell)$ has fixed size steps of width n, except for those corresponding to steps of $\mathrm{dbf}(\tau^3_{(2,4)}, \ell)$, which occur only $k-1$ points away from the preceding steps. These shorter steps make up the "hard points" that we envisioned earlier.

Lemma III.2 (Identity of demand)**.** *Let* (A, k) *be any* SCP *instance, where* $A = \{(a_1, b_1), \ldots (a_n, b_n)\}$*, and let* $x \in \mathbb{N}$ *be any natural number. Then,*

$$\mathrm{dbf}(\mathsf{T}_{(A,k)}, xn) = xn.$$

Proof: Consider any pair $(a_i, b_i) \in A$. By construction, all tasks $(e, d, p) \in \mathsf{T}_{(a_i,b_i)}$ have $e = 1$ and $p = b_i n$. By putting these values in Eq. (2) we get

$$\mathrm{dbf}(\mathsf{T}_{(a_i,b_i)}, xn) = \sum_{(e,d,p)\in\mathsf{T}_{(a_i,b_i)}} \left(\left\lfloor \frac{xn - d}{b_i n} \right\rfloor + 1 \right).$$

Let ω be the remainder of x divided by b_i,

$$\omega \stackrel{\text{def}}{=} x - \lfloor x/b_i \rfloor b_i.$$

Now, take any of the tasks $(e,d,p) \in \mathsf{T}_{(a_i,b_i)}$, and observe that because $d \leqslant p = b_i n$, we have

$$\begin{aligned}\left\lfloor \frac{xn-d}{b_i n} \right\rfloor + 1 &= \left\lfloor \frac{x}{b_i} - \frac{d}{b_i n} \right\rfloor + 1 \\ &= \left\lfloor \left\lfloor \frac{x}{b_i} \right\rfloor + \frac{\omega}{b_i} - \frac{d}{b_i n} \right\rfloor + 1 \\ &= \left\lfloor \left\lfloor \frac{x}{b_i} \right\rfloor + \frac{\omega n - d}{b_i n} \right\rfloor + 1 \\ &= \begin{cases} \lfloor x/b_i \rfloor + 1, & \text{if } d \leqslant \omega n, \\ \lfloor x/b_i \rfloor, & \text{otherwise.} \end{cases}\end{aligned}$$

How many of the tasks $(e,d,p) \in \mathsf{T}_{(a_i,b_i)}$ have $d \leqslant \omega n$? From Eq. (3) it is clear that the only tasks $(e,d,p) \in \mathsf{T}_{(a_i,b_i)}$ with $d \leqslant \omega n$ are the tasks $\tau^y_{(a_i,b_i)}$ for which $y \in \{1,\ldots,\omega\}$. Hence, ω out of the b_i tasks in $\mathsf{T}_{(a_i,b_i)}$ have $d \leqslant \omega n$, and

$$\begin{aligned}\mathrm{dbf}(\mathsf{T}_{(a_i,b_i)}, xn) &= \lfloor x/b_i \rfloor (b_i - \omega) + (\lfloor x/b_i \rfloor + 1)\,\omega \\ &= \lfloor x/b_i \rfloor b_i + \omega \\ &= \lfloor x/b_i \rfloor b_i + x - \lfloor x/b_i \rfloor b_i \\ &= x.\end{aligned} \tag{5}$$

From Eq. (2), (4) and (5) we conclude that

$$\mathrm{dbf}(\mathsf{T}_{(A,k)}, xn) = \sum_{(a_i,b_i)\in A} \mathrm{dbf}(\mathsf{T}_{(a_i,b_i)}, xn) = xn,$$

which is the claim of the lemma. ∎

The second auxiliary lemma is about the increased demand of task sets $\mathsf{T}_{(a_i,b_i)}$ for all interval lengths ℓ that are related to the congruence class of a_i modulo b_i. In particular, for all $\ell = xn + k - 1$, where $x \equiv a_i \pmod{b_i}$.

Lemma III.3 (Increased demand at congruences)**.** *Let (A,k) be any* SCP *instance, where $A = \{(a_1,b_1),\ldots(a_n,b_n)\}$, and let $x \in \mathbb{N}$ be any natural number. Then,*

$$\mathrm{dbf}(\mathsf{T}_{(a_i,b_i)}, xn+k-1) = \begin{cases} x+1, & \textit{if } x \equiv a_i \pmod{b_i}, \\ x, & \textit{otherwise,} \end{cases}$$

for all $(a_i,b_i) \in A$.

Proof: Let $\ell^* \stackrel{\text{def}}{=} xn + k - 1$. Again, by construction we have $e = 1$ and $p = b_i n$ for all $(e,d,p) \in \mathsf{T}_{(a_i,b_i)}$, and therefore

$$\mathrm{dbf}(\mathsf{T}_{(a_i,b_i)}, \ell^*) = \sum_{(e,d,p)\in\mathsf{T}_{(a_i,b_i)}} \left(\left\lfloor \frac{\ell^* - d}{b_i n} \right\rfloor + 1 \right).$$

Let $\omega \stackrel{\text{def}}{=} x - \lfloor x/b_i \rfloor b_i$ be defined as before. Take any $(e,d,p) \in \mathsf{T}_{(a_i,b_i)}$ and note that $d - k + 1 \leqslant p = b_i n$. Hence,

$$\begin{aligned}\left\lfloor \frac{\ell^* - d}{b_i n} \right\rfloor &= \left\lfloor \frac{xn - (d-k+1)}{b_i n} \right\rfloor \\ &= \begin{cases} \lfloor x/b_i \rfloor, & \text{if } d-k+1 \leqslant \omega n, \\ \lfloor x/b_i \rfloor - 1, & \text{otherwise.} \end{cases}\end{aligned}$$

We rewrite this to obtain

$$\left\lfloor \frac{\ell^* - d}{b_i n} \right\rfloor + 1 = \begin{cases} \lfloor x/b_i \rfloor + 1, & \text{if } d \leqslant \omega n + k - 1, \\ \lfloor x/b_i \rfloor, & \text{otherwise.} \end{cases}$$

From Eq. (3) it is clear that if $\omega \neq a_i$, then $d \leqslant \omega n + k - 1$ holds for all the tasks $\tau^y_{(a_i,b_i)} \in \mathsf{T}_{(a_i,b_i)}$, such that $y \leqslant \omega$. However, if $\omega = a_i$, then $d \leqslant \omega n + k - 1$ additionally holds for $\tau^{a_i+1}_{(a_i,b_i)}$. Hence, if we let α denote the number of tasks $(e,d,p) \in \mathsf{T}_{(a_i,b_i)}$ for which $d \leqslant \omega n + k - 1$ holds, then

$$\alpha = \begin{cases} \omega + 1, & \text{if } \omega = a_i, \\ \omega, & \text{otherwise.} \end{cases}$$

By the definition of ω as the remainder of x divided by b_i, we have $\omega = a_i$ if and only if $x \equiv a_i \pmod{b_i}$. Thus, we can rewrite the above as

$$\alpha = \begin{cases} x - \lfloor x/b_i \rfloor b_i + 1, & \text{if } x \equiv a_i \pmod{b_i}, \\ x - \lfloor x/b_i \rfloor b_i, & \text{otherwise.} \end{cases}$$

By applying similar steps as in Eq. (5), we conclude that

$$\begin{aligned}\mathrm{dbf}(\mathsf{T}_{(a_i,b_i)}, \ell^*) &= \lfloor x/b_i \rfloor (b_i - \alpha) + (\lfloor x/b_i \rfloor + 1)\,\alpha \\ &= \lfloor x/b_i \rfloor b_i + \alpha \\ &= \begin{cases} x+1, & \text{if } x \equiv a_i \pmod{b_i}, \\ x, & \text{otherwise,} \end{cases}\end{aligned}$$

from which the lemma follows. ∎

We can now prove the correctness of the transformation.

Lemma III.4 (Validity of transformation)**.** *For any* SCP *instance (A,k), the corresponding task system $\mathsf{T}_{(A,k)}$ is infeasible if and only if (A,k) is a* yes*-instance.*

Proof: Let $A = \{(a_1,b_1),\ldots,(a_n,b_n)\}$. First we note that $\mathrm{U}(\mathsf{T}_{(a_i,b_i)}) = 1/n$ for any $(a_i,b_i) \in A$, and consequently that $\mathrm{U}(\mathsf{T}_{(A,k)}) = 1$. By Theorem II.3 it follows that the feasibility of $\mathsf{T}_{(A,k)}$ is exactly decided by the truth value of the formula in Eq. (1).

We prove the two directions of the lemma separately, beginning with the if-case. Figure 5 serves as an illustration.

- *(A,k) is a* yes*-instance $\Rightarrow$ $\mathsf{T}_{(A,k)}$ is infeasible:*
  By assumption, there is a subset $A' \subseteq A$ of size at least k and an $x \in \mathbb{N}$, such that $x \equiv a_i \pmod{b_i}$ for all $(a_i,b_i) \in A'$. Without loss of generality, let that A' be the largest such subset.

Let $\ell^* \stackrel{\text{def}}{=} xn + k - 1$. By Lemma III.3 and the above assumptions, we have

$$\mathrm{dbf}(\mathsf{T}_{(a_i,b_i)}, \ell^*) = \begin{cases} x+1, & \text{if } (a_i, b_i) \in A', \\ x, & \text{otherwise.} \end{cases}$$

Because $|A'| \geqslant k$, it follows that

$$\begin{aligned} \mathrm{dbf}(\mathsf{T}_{(A,k)}, \ell^*) &= \sum_{(a_i,b_i)\in A} \mathrm{dbf}(\mathsf{T}_{(a_i,b_i)}, \ell^*) \\ &\geqslant xn + k > \ell^*, \end{aligned}$$

and $\mathsf{T}_{(A,k)}$ is infeasible by Theorem II.3.

- (A, k) *is a* no*-instance* $\Rightarrow \mathsf{T}_{(A,k)}$ *is feasible:*
  From Eq. (2) it is clear that $\mathrm{dbf}(\mathsf{T}_{(A,k)}, \ell)$ is a right-continuous, non-decreasing step function in ℓ. Let Δ be the set of points at which this function changes value, including point 0:

  $$\Delta \stackrel{\text{def}}{=} \left\{\ell \mid \mathrm{dbf}(\mathsf{T}_{(A,k)}, \ell) \text{ is discontinuous at } \ell\right\} \cup \{0\}$$

  It is easily seen that if there exists some $\ell \geqslant 0$ such that $\mathrm{dbf}(\mathsf{T}_{(A,k)}, \ell) > \ell$, then there must exist some $\ell' \in \Delta$ such that $\mathrm{dbf}(\mathsf{T}_{(A,k)}, \ell') > \ell'$.
  Hence, by showing that $\mathrm{dbf}(\mathsf{T}_{(A,k)}, \ell) \leqslant \ell$ for all $\ell \in \Delta$, we can conclude that there exists no $\ell \geqslant 0$ such that $\mathrm{dbf}(\mathsf{T}_{(A,k)}, \ell) > \ell$. In order to do so we need to find a more concrete characterization of Δ.
  Note that from Eq. (3) we know that for all tasks $(e, d, p) \in \mathsf{T}_{(A,k)}$ we have

  $$\begin{aligned} p &= bn, \text{ for some } b \in \mathbb{N}, \\ d &\in \{yn, yn + k - 1\}, \text{ for some } y \in \mathbb{N}. \end{aligned}$$

  By the definition of demand bound functions given in Eq. (2), it follows that the points at which $\mathrm{dbf}(\mathsf{T}_{(A,k)}, \ell)$ is discontinuous are of the form xn or $xn + k - 1$ for some $x \in \mathbb{N}$, and therefore

  $$\Delta \subseteq \{xn \mid x \in \mathbb{N}\} \cup \{xn + k - 1 \mid x \in \mathbb{N}\}.$$

  From Lemma III.2 we directly have

  $$\mathrm{dbf}(\mathsf{T}_{(A,k)}, \ell) = \ell, \text{ for all } \ell \in \{xn \mid x \in \mathbb{N}\}.$$

  Consider instead any $\ell^* = xn + k - 1$, where $x \in \mathbb{N}$. By Lemma III.3, we know that for any $(a_i, b_i) \in A$,

  $$\mathrm{dbf}(\mathsf{T}_{(a_i,b_i)}, \ell^*) = \begin{cases} x+1, & \text{if } x \equiv a_i \pmod{b_i}, \\ x, & \text{otherwise,} \end{cases}$$

  Now, by assumption, (A, k) is a *no*-instance of SCP. It follows that there are at most $k - 1$ pairs $(a_i, b_i) \in A$ such that $x \equiv a_i \pmod{b_i}$. Hence,

  $$\begin{aligned} \mathrm{dbf}(\mathsf{T}_{(A,k)}, \ell^*) &= \sum_{(a_i,b_i)\in A} \mathrm{dbf}(\mathsf{T}_{(a_i,b_i)}, \ell^*) \\ &\leqslant xn + k - 1 = \ell^*, \end{aligned}$$

  for all $\ell^* \in \{xn + k - 1 \mid x \in \mathbb{N}\}$.

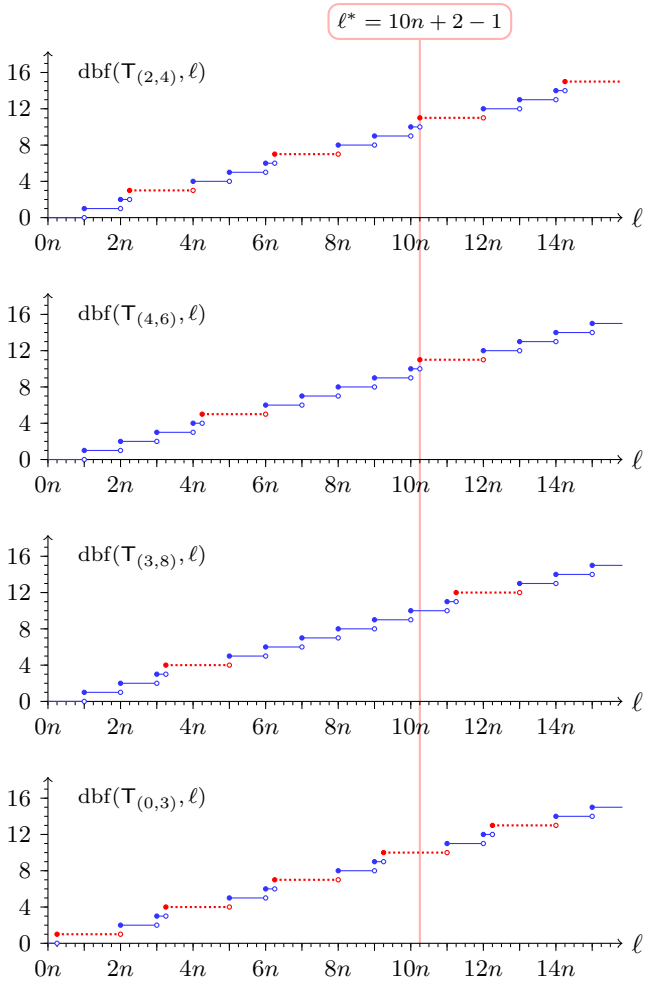

Fig. 5. Demand bound functions for task sets generated from an SCP instance (A, k), where $A = \{(2, 4), (4, 6), (3, 8), (0, 3)\}$ and $k = 2$. As we have seen in Figure 2, $(A, 2)$ is a *yes*-instance of SCP and therefore $\mathsf{T}_{(A,2)}$ should be infeasible. Indeed, for any x belonging to two of A's congruence classes (such as $x = 10$) we have two "early" steps in the corresponding demand bound functions aligning at $\ell^* = xn + 2 - 1$, and $\mathrm{dbf}(\mathsf{T}_{(A,2)}, \ell^*) = \ell^* + 1$. If instead $k = 3$, then the early steps would move one unit to the right in the figures, and two such steps aligning at some $\ell^* = xn + 3 - 1$ is no longer enough to witness infeasibility, because $\mathrm{dbf}(\mathsf{T}_{(A,3)}, \ell^*) = \ell^*$.

In conclusion, we have shown that $\mathrm{dbf}(\mathsf{T}_{(A,k)}, \ell) \leqslant \ell$ for all $\ell \in \Delta$, and consequently for all $\ell \geqslant 0$. The feasibility of $\mathsf{T}_{(A,k)}$ is ensured by Theorem II.3. ■

Our main theorem follows.

Theorem III.5 (Intractability)**.** *Deciding whether a sporadic task system with constrained deadlines is feasible on a preemptive uniprocessor is* coNP*-complete in the strong sense.*

Proof: There is a pseudo-polynomial transformation from SCP to the complement of the feasibility problem. By Theorems II.2 and II.4 we know that SCP is strongly NP-hard and that the feasibility problem is in coNP. ■

IV. Conclusions

We have showed that there can be no pseudo-polynomial time algorithm for deciding the feasibility of constrained-deadline sporadic task systems on a preemptive uniprocessor, unless $\mathsf{P} = \mathsf{NP}$.

This highlights the inherent practical importance of the special case where the utilization of the task system is a priori bounded by some constant $c < 1$, for which Baruah et al. [3] have described a pseudo-polynomial time feasibility test. This test (or variations thereof) is widely used in the literature and has proven to be quite tractable, at least for off-line analysis, even for large c such as 0.9 or 0.95. An important outstanding question is whether this special case also has a polynomial time solution.

References

[1] C. L. Liu and J. W. Layland, "Scheduling algorithms for multiprogramming in a hard-real-time environment," *Journal of the ACM*, vol. 20, no. 1, pp. 46–61, 1973.

[2] M. L. Dertouzos, "Control robotics: The procedural control of physical processes," in *Proceedings of the IFIP congress*, vol. 74, 1974, pp. 807–813.

[3] S. Baruah, A. Mok, and L. Rosier, "Preemptively scheduling hard-real-time sporadic tasks on one processor," in *Proceedings of the Real-Time Systems Symposium (RTSS)*, 1990, pp. 182–190.

[4] 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.

[5] S. K. Baruah, L. E. Rosier, and R. R. Howell, "Algorithms and complexity concerning the preemptive scheduling of periodic, real-time tasks on one processor," *Real-Time Systems*, vol. 2, no. 4, pp. 301–324, 1990.

[6] S. Baruah and K. Pruhs, "Open problems in real-time scheduling," *Journal of Scheduling*, vol. 13, no. 6, pp. 577–582, 2010.

[7] F. Eisenbrand and T. Rothvoß, "EDF-schedulability of synchronous periodic task systems is coNP-hard," in *Proceedings of the ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 2010, pp. 1029–1034.

[8] J. Y.-T. Leung and J. Whitehead, "On the complexity of fixed-priority scheduling of periodic, real-time tasks," *Performance Evaluation*, vol. 2, no. 4, pp. 237–250, 1982.

[9] M. R. Garey and D. S. Johnson, "Strong NP-completeness results: Motivation, examples, and implications," *Journal of the ACM*, vol. 25, no. 3, pp. 499–508, 1978.

$\boxtimes$