Fuzzy Simultaneous Congruences

Max A. Deppert
Kiel University, Germany

Klaus Jansen
Kiel University, Germany

Kim-Manuel Klein
Kiel University, Germany

Abstract

We introduce a very natural generalization of the well-known problem of simultaneous congruences. Instead of searching for a positive integer s that is specified by n *fixed remainders* modulo integer divisors $a_1, \ldots, a_n$ we consider *remainder intervals* $R_1, \ldots, R_n$ such that s is feasible if and only if s is congruent to r_i modulo a_i for *some* remainder r_i in interval R_i for all i.

This problem is a special case of a 2-stage integer program with only two variables per constraint which is is closely related to directed Diophantine approximation as well as the mixing set problem. We give a hardness result showing that the problem is NP-hard in general.

By investigating the case of *harmonic divisors*, i.e. a_{i+1}/a_i is an integer for all $i < n$, which was heavily studied for the mixing set problem as well, we also answer a recent algorithmic question from the field of real-time systems. We present an algorithm to decide the feasibility of an instance in time $\mathcal{O}(n^2)$ and we show that if it exists even the *smallest* feasible solution can be computed in strongly polynomial time $\mathcal{O}(n^3)$.

2012 ACM Subject Classification Theory of computation → Discrete optimization; Theory of computation → Integer programming

Keywords and phrases Simultaneous congruences, Integer programming, Mixing Set, Real-time scheduling, Diophantine approximation

Digital Object Identifier 10.4230/LIPIcs.MFCS.2021.39

Related Version *Full Version*: `https://arxiv.org/abs/2002.07746`

Funding *Max A. Deppert*: Supported by German Research Foundation (DFG) project JA 612/20-1.
Klaus Jansen: Supported by German Research Foundation (DFG) project JA 612/20-1.

1 Introduction

In the recent past there was a great interest in the so-called *n-fold* integer programs [10, 17, 19] and *2-stage* integer programs [18, 20]. The matrix $\mathcal{A}$ of a 2-stage integer program is constructed by block matrices $A^{(1)}, \ldots, A^{(n)} \in \mathbb{Z}^{r \times k}$ and $B^{(1)}, \ldots, B^{(n)} \in \mathbb{Z}^{r \times t}$ as follows:

$$\mathcal{A} = \begin{pmatrix} A^{(1)} & B^{(1)} & 0 & \cdots & 0 \\ A^{(2)} & 0 & B^{(2)} & \ddots & \vdots \\ \vdots & \vdots & \ddots & \ddots & 0 \\ A^{(n)} & 0 & \cdots & 0 & B^{(n)} \end{pmatrix}$$

For an objective vector $c \in \mathbb{Z}_{\geq 0}^{k+nt}$, a right-hand side $b \in \mathbb{Z}^{nr}$, and bounds $\ell, u \in \mathbb{Z}_{\geq 0}^{k+nt}$ the 2-stage integer program is formulated as

$$\max \{ c^T x \mid \mathcal{A}x = b,\ \ell \leq x \leq u,\ x \in \mathbb{Z}^{k+nt} \}.$$

A special case of a 2-stage integer program is given by the problem Mixing Set [6, 7, 15] (with only two variables in each constraint) where especially $r = k = t = 1$ and $A^{(1)} = \cdots = A^{(n)}$. Remark that 2-variable integer programming problems were extensively studied by various authors, e.g. [3, 22] or [12] (with two variables in total). MIXING SET plays an important role for example in integer programming approaches for production planning [26]. Given vectors $a, b \in \mathbb{Q}^n$ one aims to compute

$$\min \{ f(s,x) \mid s + a_i x_i \geq b_i \forall i = 1, \ldots, n, (s,x) \in \mathbb{Z}_{\geq 0} \times \mathbb{Z}^n \} \tag{1}$$

for some objective function f. Conforti et al. [8] pose the question whether the problem can be solved in polynomial time for linear functions f. Unless P = NP this was ruled out by Eisenbrand and Rothvoß [13] who proved that optimizing any linear function over MIXING SET is NP-hard. However, the problem can be solved in polynomial time if $a_i = 1$ [15, 24] or if the capacities a_i fulfil a *harmonic* property [30], i.e. a_{i+1}/a_i is integer for all $i < n$. The case of harmonic capacities was intensively studied - see [8, 9] for simpler approaches.

More recently, real-time systems with harmonic tasks (the periods are integer multiples of each other) have received increased attention [5] and also harmonic periods have been considered before [2, 11, 27, 29]. Now a recent manuscript in the field of real-time systems by Nguyen et al. [25] gives rise to the study of a new problem. Nguyen et al. present an algorithm for the worst-case response time analysis of harmonic tasks with constrained release jitter running in polynomial time. The release jitter of a task is the maximum difference between the arrival times and the release times over all jobs of the task. Their algorithm uses heuristic components to solve an integer program that can be stated as a bounded version of MIXING SET with additional upper bounds B_i as follows.

BOUNDED MIXING SET (BMS)
Given capacities $a_1, \ldots, a_n \in \mathbb{Z}$ and bounds $b, B \in \mathbb{Z}^n$ find $(s,x) \in \mathbb{Z}_{\geq 0} \times \mathbb{Z}^n$ such that

$$b_i \leq s + a_i x_i \leq B_i \quad \forall i = 1, \ldots, n.$$

In particular they depend on minimizing the value of s which can be achieved in linear time in case of MIXING SET. While BMS may look artificial at first sight it is not; in fact, leading to a very natural generalization it can be restated in the well-known form of *simultaneous congruences*.

FUZZY SIMULTANEOUS CONGRUENCES (FSC)
Given divisors $a_1, \ldots, a_n \in \mathbb{Z} \setminus \{0\}$ and remainder intervals $R_1, \ldots, R_n \subseteq \mathbb{Z}$ and an interval $S \subseteq \mathbb{Z}_{\geq 0}$ find a number $s \in S$ such that

$$\exists\, r_i \in R_i : s \equiv r_i \pmod{a_i} \quad \forall i = 1, \ldots, n.$$

Obviously, this also generalizes over the well-known problem of the Chinese Remainder Theorem (CRT). Here we give its generalized form (cf. [21]).

▶ **Theorem 1** (Generalized Chinese Remainder Theorem). *Given divisors $a_1, \ldots, a_n \in \mathbb{Z}_{\geq 1}$ and remainders $r_1, \ldots, r_n \in \mathbb{Z}_{\geq 0}$ the system of n simultaneous congruences $s \equiv r_i \pmod{a_i}$ admits a solution $s \in \mathbb{Z}$ if and only if $r_i \equiv r_j \pmod{\gcd(a_i, a_j)}$ for all $i \neq j$.*

Furthermore, Leung and Whitehead [23] showed that k-Simultaneous Congruences (k-SC) is NP-complete in the weak sense. Given divisors $a_1, \ldots, a_n \in \mathbb{Z}_{\geq 1}$ and remainders $r_1, \ldots, r_n \in \mathbb{Z}_{\geq 0}$ the task is to find a number $s \in \mathbb{Z}_{\geq 0}$ and a subset $I \subseteq \{1, \ldots, n\}$ with $|I| = k$ s.t. $s \equiv r_i \pmod{a_i}$ for all $i \in I$. Later it was shown by Baruah et al. [4] that k-SC also is NP-complete in the strong sense.

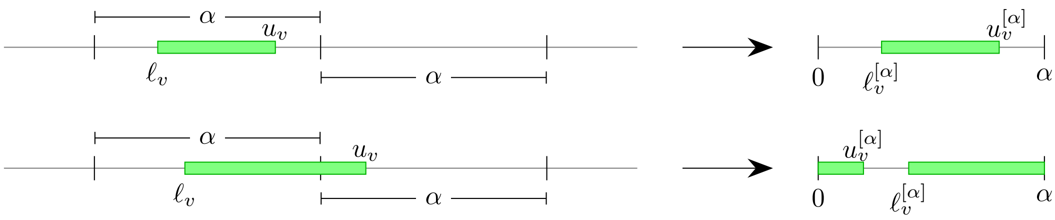

Figure 1 The two possibilities for the modular projection of an interval.

Both problems BMS and FSC are interchangeable formulations of the same problem (see Section 2). Therefore, we will use them as synonyms and we especially assume formally that $R_i = [b_i, B_i]$. Interestingly and to the best of our knowledge, FSC/BMS was not considered before. However, the investigation of simultaneous congruences has always been of transdisciplinary interest connecting a variety of fields and applications, e.g. [1, 14, 16].

Our Contribution

(a) We show that BMS is NP-hard for general capacities a_i. For the reduction from Directed Diophantine Approximation we refer to the appendix. Compared to Mixing Set this is a stronger hardness result as BMS by itself only asks for an *arbitrary* feasible solution. Remark that every feasible instance of Mixing Set may be solved by $s = \|b\|_\infty, x = \mathbf{0}$.

(b) In the case of harmonic capacities (i.e. a_{i+1}/a_i is an integer for all $i < n$), which was heavily studied for Mixing Set as mentioned before, we give an algorithm exploiting a merge idea based on modular arithmetic on intervals to decide the feasibility problem of FSC in time $\mathcal{O}(n^2)$. See Section 3.1 for the details.

(c) Furthermore, for a feasible instance of FSC with harmonic capacities we present a polynomial algorithm as well as a strongly polynomial algorithm to compute the smallest feasible solution to FSC in time $\mathcal{O}(\min\{n^2 \log(a_n), n^3\}) \leq \mathcal{O}(n^3)$. See Section 3.2 for the details.

(d) Our algorithm gives a strongly polynomial replacement for the heuristic component (which may fail to compute a solution) in the algorithm of Nguyen et al. [25]. However, we present an algorithm to solve the problem in linear time. See Section 4 for the details.

2 Notation and General Properties

For the sake of readability we write $X^{[\alpha]} = (X \bmod \alpha)$ for numbers X as well as $X^{[\alpha]} = \{\, z \bmod \alpha \mid z \in X \,\}$ for sets X (of numbers) to denote the modular projection of some number or interval, respectively. Extending the usual notation we also write $X \equiv Y \pmod{\alpha}$ if $X^{[\alpha]} = Y^{[\alpha]}$ for sets X, Y. Notice that on the one hand $(X \cup Y)^{[\alpha]} = X^{[\alpha]} \cup Y^{[\alpha]}$ but on the other hand be aware that $(X \cap Y)^{[\alpha]} \neq X^{[\alpha]} \cap Y^{[\alpha]}$ in general (cf. Lemma 9). Figure 1 depicts the structure of $v^{[\alpha]}$ if $v = [\ell_v, u_v]$ is an interval in $\mathbb{Z}$.

The empty set is denoted by $\varnothing$. Also we use the well-tried notation $t+X = \{\, t+z \mid z \in X \,\}$ to express the *translation* of a set of numbers X by some number t. For a set of sets $\mathcal{S}$ we write $\bigcup \mathcal{S}$ to denote the union $\bigcup_{S\in\mathcal{S}} S$. Furthermore, we identify constraints by their indices. So for $i \leq n$ we say that "$b_i \leq s + a_i x_i \leq B_i$" *is* constraint i.

Identity of BMS and FSC

It is important to notice that BMS allows zero capacities while FSC cannot allow zero divisors since (mod 0) is undefined. However, consider a constraint i of BMS with $a_i \neq 0$. Let $b_i \le s + a_i x_i \le B_i$ be satisfied and set $r_i = s + a_i x_i$. Then $r_i^{[a_i]} = s^{[a_i]}$ and $r_i \in [b_i, B_i] = R_i$. Vice-versa let $r_i \in R_i$ s.t. $r_i \equiv s \pmod{a_i}$. Then there is an $x_i \in \mathbb{Z}$ s.t. $s + a_i x_i = r_i \in R_i = [b_i, B_i]$.

A constraint i that holds $a_i = 0$ simply demands that $s \in R_i$. Hence, if $a_i = a_j = 0$ for two constraints $i \neq j$ they can be replaced by one new constraint k defined by $R_k = R_i \cap R_j$. Therefore, one may assume that there is at most one constraint i with a zero capacity a_i. However, as all our results can be lifted back to the general case with low effort we will assume in terms of BMS that all capacities are non-zero and for FSC we make the equivalent assumption that $S = \mathbb{Z}_{\ge 0}$.

With our notation we may easily express the feasibility of a value s for a single constraint i as follows.

▶ **Observation 2.** *A value s satisfies constraint i if and only if $s^{[a_i]} \in R_i^{[a_i]}$.*

Proof. It holds that $\exists r_i \in R_i : r_i \equiv s \pmod{a_i}$ iff $\exists r_i \in R_i : r_i^{[a_i]} = s^{[a_i]}$ iff $s^{[a_i]} \in R_i^{[a_i]}$. ◀

By simply swapping the signs of the x_i we may assume that $a_i \ge 0$ for all i. We may also assume that the intervals are small in the sense that $B_i - b_i + 1 < a_i$ holds for all i. Assume that $B_i - b_i + 1 \ge a_i$ for an i and let $s \ge 0$ be an arbitrary integer. Then $b_i \le B_i - a_i + 1$ and constraint i may always be solved by setting $x_i = \lceil (b_i - s)/a_i \rceil$ which satisfies

$$b_i \le s + a_i \underbrace{\lceil \tfrac{b_i - s}{a_i} \rceil}_{x_i} \le s + a_i \lceil \tfrac{B_i - a_i + 1 - s}{a_i} \rceil = s + a_i \lfloor \tfrac{B_i - s}{a_i} \rfloor \le B_i.$$

Hence, constraint i is redundant and may be omitted. As a direct consequence there can be at most one feasible value for each x_i for a given guess s. In fact, we can decide the feasibility of a guess s in time $\mathcal{O}(n)$ as for all constraints i and values x_i it holds $b_i \le s + a_i x_i \le B_i$ if and only if $\lceil (b_i - s)/a_i \rceil = x_i = \lfloor (B_i - s)/a_i \rfloor$. So a guess s is feasible if and only if $\lceil (b_i - s)/a_i \rceil = \lfloor (B_i - s)/a_i \rfloor$ holds for all constraints i. Another consequence is that BMS is a generalization of Mixing Set as one can always add trivial upper bounds. By $s_{\min}$ we denote the smallest feasible solution s that satisfies all constraints.

▶ **Observation 3.** *For feasible instances it holds that* $\mathrm{s}_{\min} < \mathrm{lcm}(a_1, \ldots, a_n)$.

Proof. Let $\varphi = \mathrm{lcm}(a_1, \ldots, a_n)$. Remark that φ / a_i is integral for all i. Assume that (s, x) is a solution with $s = s_{\min} \ge \varphi$. Let $t = s - \varphi$ and $y_i = x_i + \varphi / a_i$ f.a. i. Then $0 \le t < s_{\min}$ and $t + a_i y_i = s + a_i x_i$ f.a. i. So (t, y) is a solution that contradicts the optimality. ◀

3 Harmonic Divisors

Here we consider harmonic divisors in the sense that a_{i+1}/a_i is an integer for all $i < n$.

As we investigate some kind of a generalization of the setting of the Chinese Remainder Theorem, it is natural to ask for a CRT for *harmonic* (instead of the usually coprime) divisors and of course the (generalized) CRT answers this question; in this case we have $\gcd(a_i, a_n) = a_i$ and so Theorem 1 reveals that if the system of n simultaneous congruences $s \equiv r_i \pmod{a_i}$ admits a solution then $r_i \equiv r_n \pmod{a_n}$ which says that if there is any solution then the set of all solutions is $a_n \mathbb{Z} + r_n^{[a_n]}$. However, it turns out that the investigation of FSC is a lot more complicated.

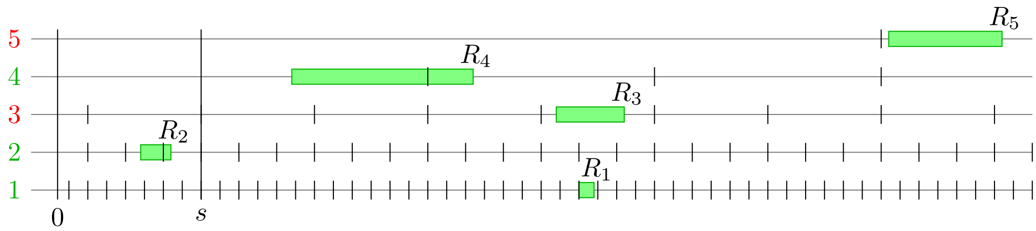

Figure 2 $36a_1 = 18a_2 = 6a_3 = 3a_4 = a_5$. The guess s is not feasible for constr. 3 and 5.

In this section we present an algorithm to decide the feasibility of an instance of FSC. Also we show how optimal solutions can be computed in (strongly) polynomial time. Both of these results are based on the fine-grained interconnection between modular arithmetic on sets and the harmonic property. For some intuition Figure 2 gives a perspective on s as an anchor for 1-dimensional lattices with basis a_i which have to "hit" the intervals R_i. For example, in the figure it holds that $s + a_2 \cdot (-1) = s - a_2 \in R_2$, so the 1-dimensional lattice $(s + a_2 z)_{z\in\mathbb{Z}}$ hits interval R_2. Therefore, the choice of s satisfies constraint 2.

3.1 Deciding feasibility

The idea for our first algorithm will be to decide the feasibility problem by iteratively computing modular projections from constraint $i = n$ down to $i = 1$. In the following we will say that an interval $w \subseteq \mathbb{Z}$ *represents* a set $M \subseteq \mathbb{Z}$ (modulo α) if $w^{[\alpha]} = M^{[\alpha]}$. Also a set of intervals $\mathcal{R}$ represents a set $M \subseteq \mathbb{Z}$ (modulo α) if $M^{[\alpha]} = \bigcup_{w\in\mathcal{R}} w^{[\alpha]}$. Given an integer $\alpha \geq 1$ and two intervals v, w we need to study the structure of the intersection $v^{[\alpha]} \cap w^{[\alpha]} \subseteq [0, \alpha)$. To express it let $v = [\ell_v, u_v], w = [\ell_w, u_w]$ and we define the basic intervals

$$\varphi_\alpha(v, w) = [\ell_v^{[\alpha]}, u_w^{[\alpha]}] \quad \text{and} \quad \psi_\alpha(v, w) = [\max\{\ell_v^{[\alpha]}, \ell_w^{[\alpha]}\}, \alpha + \min\{u_v^{[\alpha]}, u_w^{[\alpha]}\}]$$

for all intervals v, w. The former may be thought as the cases where $v^{[\alpha]}$ and $w^{[\alpha]}$ are two overlapping intervals while the intuition for the latter are situations where $v^{[\alpha]}$ and $w^{[\alpha]}$ both consist of two intervals which are in pairs overlapping. Remark that $\psi_\alpha(w, v) = \psi_\alpha(v, w)$ is always true.

▶ **Lemma 4.** *Given an integer $\alpha \geq 1$ and two intervals $v, w \subseteq \mathbb{Z}$ it holds that*

$$\begin{aligned} v^{[\alpha]} \cap w^{[\alpha]} \in \{ \ & \varnothing,\ v^{[\alpha]},\ w^{[\alpha]},\ \psi_\alpha(v, w)^{[\alpha]},\ \varphi_\alpha(v, w),\ \varphi_\alpha(w, v), \\ & \varphi_\alpha(v, w) \,\dot\cup\, \varphi_\alpha(w, v),\ \varphi_\alpha(v, w) \,\dot\cup\, \psi_\alpha(v, w)^{[\alpha]},\ \varphi_\alpha(w, v) \,\dot\cup\, \psi_\alpha(v, w)^{[\alpha]} \ \}. \end{aligned}$$

The important intuition is that such a "modulo α intersection" can always be represented by at most two intervals. Remark that the sets in the second row are the only ones which are represented by $2 > 1$ intervals. Due to space reasons for the case distinction to prove Lemma 4 we refer to Appendix B and especially to Figure 6.

While Lemma 4 gives structure to intersections of two modular projections of intervals, the next lemma reveals how many intervals will be required to represent a *one-to-many* intersection. We will use this bound in every step of our algorithm. We want to add that both of these lemmas and even Lemma 6 do not depend on the harmonic property by themselves. However, they turn out to be especially useful in this setting.

▶ **Lemma 5.** *Let $\alpha \geq 1$, let v be an interval and let Q be a set of $k \geq 1$ intervals. Then there is a set R of at most $k+1$ intervals s.t. $v^{[\alpha]} \cap (\bigcup Q)^{[\alpha]} = (\bigcup R)^{[\alpha]}$.*

Proof. We simply obtain that

$$v^{[\alpha]} \cap \Big(\bigcup Q\Big)^{[\alpha]} = \bigcup_{w\in Q} (v^{[\alpha]} \cap w^{[\alpha]}) = \bigcup_{\substack{w\in Q\\ w^{[\alpha]}\subseteq v^{[\alpha]}}} w^{[\alpha]} \cup \bigcup_{w\in D} (v^{[\alpha]} \cap w^{[\alpha]})$$

where $D = \{\, w \in Q \mid w^{[\alpha]} \nsubseteq v^{[\alpha]}, w^{[\alpha]} \cap v^{[\alpha]} \neq \varnothing \,\}$ denotes the subset of intervals that cause the interesting intersections with $v^{[\alpha]}$ (cf. Lemma 4). Obviously, all other intersections can be represented by at most one interval each. So we study the intersections with D. In fact, everything gets simple if there are $w_1, w_2 \in D$ such that $v^{[\alpha]} \cap w_1^{[\alpha]} = \varphi_\alpha(v, w_1) \,\dot\cup\, \psi_\alpha(v, w_1)^{[\alpha]}$ and $v^{[\alpha]} \cap w_2^{[\alpha]} = \varphi_\alpha(w_2, v) \,\dot\cup\, \psi_\alpha(v, w_2)^{[\alpha]}$. By simply adapting the inequalities of the first case distinction in the proof of Lemma 4 we find

$$\begin{aligned}
&(v^{[\alpha]} \cap w_1^{[\alpha]}) \cup (v^{[\alpha]} \cap w_2^{[\alpha]})\\
&\quad = ([0, u_v^{[\alpha]}] \,\dot\cup\, [\ell_v^{[\alpha]}, u_{w_1}^{[\alpha]}] \,\dot\cup\, [\ell_{w_1}^{[\alpha]}, \alpha)) \cup ([0, u_{w_2}^{[\alpha]}] \,\dot\cup\, [\ell_{w_2}^{[\alpha]}, u_v^{[\alpha]}] \,\dot\cup\, [\ell_v^{[\alpha]}, \alpha))\\
&\quad = [0, u_v^{[\alpha]}] \,\dot\cup\, [\ell_v^{[\alpha]}, \alpha) = v^{[\alpha]}
\end{aligned}$$

which implies that $v^{[\alpha]} \cap (\bigcup Q)^{[\alpha]} = v^{[\alpha]}$ can be represented by only one interval, namely v. Therefore, in order to get an upper bound we assume that these two types of intersections do not come together. In more detail, we may assume by symmetry that $D = D_1 \,\dot\cup\, D_2$ where

$$\begin{aligned}
D_1 &= \{\, w \in D \mid v^{[\alpha]} \cap w^{[\alpha]} = \varphi_\alpha(v, w) \,\dot\cup\, \varphi_\alpha(w, v) \,\} \text{ and}\\
D_2 &= \{\, w \in D \mid v^{[\alpha]} \cap w^{[\alpha]} = \varphi_\alpha(v, w) \,\dot\cup\, \psi_\alpha(v, w)^{[\alpha]} \,\}\,.
\end{aligned}$$

It turns out that

$$\begin{aligned}
\bigcup_{w\in D_1} (v^{[\alpha]} \cap w^{[\alpha]}) &= \bigcup_{w\in D_1} ([\ell_v^{[\alpha]}, u_w^{[\alpha]}] \,\dot\cup\, [\ell_w^{[\alpha]}, u_v^{[\alpha]}])\\
&= [\ell_v^{[\alpha]}, \max_{w\in D_1} u_w^{[\alpha]}] \cup [\min_{w\in D_1} \ell_w^{[\alpha]}, u_v^{[\alpha]}] \quad \text{and}\\
\bigcup_{w\in D_2} (v^{[\alpha]} \cap w^{[\alpha]}) &= \bigcup_{w\in D_2} ([\ell_v^{[\alpha]}, u_w^{[\alpha]}] \,\dot\cup\, [\ell_w^{[\alpha]}, \alpha + u_v^{[\alpha]}]^{[\alpha]})\\
&= [\ell_v^{[\alpha]}, \max_{w\in D_2} u_w^{[\alpha]}] \cup [\min_{w\in D_2} \ell_w^{[\alpha]}, \alpha + u_v^{[\alpha]}]^{[\alpha]}
\end{aligned}$$

which finally joins up to

$$\bigcup_{w\in D} (v^{[\alpha]} \cap w^{[\alpha]}) = [\ell_v^{[\alpha]}, \max_{w\in D} u_w^{[\alpha]}] \cup [\min_{w\in D} \ell_w^{[\alpha]}, \alpha + u_v^{[\alpha]}]^{[\alpha]}.$$

Hence, all intersections with intervals in D may be represented by at most two intervals in total while each other intersection can be represented by at most one interval. Thus, if $|D| = 0$ then the whole intersection can be represented by at most k intervals. If $|D| \geq 1$ then there are at most $2 + |Q| - |D| \leq 2 + k - 1 = k + 1$ intervals required. ◀

Let S_i denote the set of all solutions $s \in \mathbb{Z}_{\geq 0}$ that are feasible for each of the constraints $i, i+1, \dots, n$. We set $S_{n+1} = \mathbb{Z}_{\geq 0}$ to denote the feasible solutions to an empty set of constraints. The correctness of Algorithm 1 is implied by the following fundamental lemma. See Figure 3 for an example of a step inside the algorithm.

▶ **Lemma 6.** *It holds true that* $S_i^{[a_i]} = R_i^{[a_i]} \cap S_{i+1}^{[a_i]}$ *for all* $i = 1, \dots, n$.

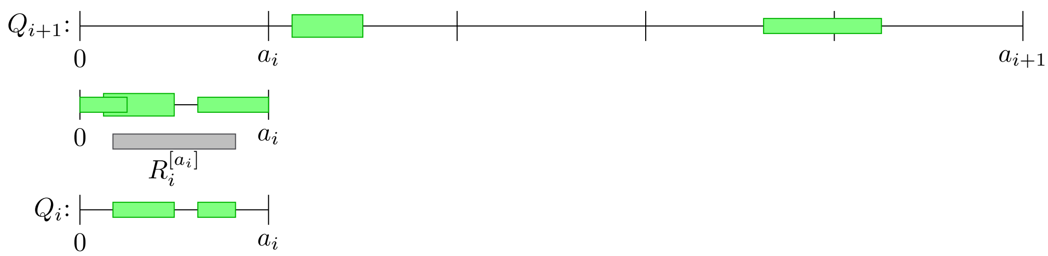

Figure 3 A step from $i+1$ to i; modular projection to $[0, a_i)$ and intersection with $R_i^{[a_i]}$.

Algorithm 1 Feasibility test for FSC.

```
procedure FEASIBLE(I = (a_1, ..., a_n, R_1, ..., R_n))
    Q_n ← {R_n}
    for i = n − 1, ..., 1 do
        Compute set Q_i s. t. (⋃Q_i)^[a_i] = R_i^[a_i] ∩ (⋃Q_{i+1})^[a_i] and |Q_i| ≤ O(n − i)
    if ⋃Q_1 = ∅ then
        return "infeasible"
    else
        return "feasible"
```

Proof. Let $r \in S_i^{[a_i]}$. So there is a solution $s \in S_i$ such that $r = s^{[a_i]} \in R_i^{[a_i]}$. It holds that $S_i \subseteq S_{i+1}$ which implies $s \in S_{i+1}$ and thus $r = s^{[a_i]} \in S_{i+1}^{[a_i]}$.

Vice-versa let $r \in R_i^{[a_i]} \cap S_{i+1}^{[a_i]}$. So there is a solution $s \in S_{i+1}$ with $s^{[a_i]} = r$. From $r \in R_i^{[a_i]}$ we get $s^{[a_i]} \in R_i^{[a_i]}$. Hence, $s \in S_i$ and $r = s^{[a_i]} \in S_i^{[a_i]}$. ◀

▶ **Theorem 7.** *Algorithm 1 decides the feasibility of an instance in time $\mathcal{O}(n^2)$.*

Proof. We show that $\bigcup Q_i \equiv S_i \pmod{a_i}$ for all $i = n, \ldots, 1$. This will prove the algorithm correct since then $\bigcup Q_1 \equiv S_1 \pmod{a_1}$ and that means $\bigcup Q_1$ is empty if and only if S_1 is empty. Obviously it holds that $\bigcup Q_n \equiv S_n \pmod{a_n}$ since $\bigcup Q_n = R_n$. Now suppose that $\bigcup Q_{i+1} \equiv S_{i+1} \pmod{a_{i+1}}$ for some $i \geq 1$. We have that $(\bigcup Q_i)^{[a_i]} = R_i^{[a_i]} \cap (\bigcup Q_{i+1})^{[a_i]}$ where the harmonic property implies $(\bigcup Q_{i+1})^{[a_i]} = ((\bigcup Q_{i+1})^{[a_{i+1}]})^{[a_i]} = (S_{i+1}^{[a_{i+1}]})^{[a_i]} = S_{i+1}^{[a_i]}$. Together with Lemma 6 this yields $(\bigcup Q_i)^{[a_i]} = R_i^{[a_i]} \cap S_{i+1}^{[a_i]} = S_i^{[a_i]}$ and that proves the algorithm correct. Using Lemmas 4–6 each set Q_i can be computed in time $\mathcal{O}(n)$ and this yields a total running time of $\mathcal{O}(n^2)$. ◀

3.2 Optimal solutions

Unfortunately, Algorithm 1 neither calculates a solution nor it directly implies one. Here we present an algorithm to compute the smallest feasible solution $s_{\min}$ to FSC. However, by searching in the opposite direction the same technique also applies to the computation of the largest feasible solution $s_{\max} < a_n$. We start with a simple binary search approach.

▶ **Corollary 8.** *For feasible instances $s_{\min}$ can be computed in time $\mathcal{O}(n^2 \log(a_n))$.*

This can be achieved by introducing an additional constraint measuring the value of s as follows. Let β be a positive integer. We extend the problem instance by a new constraint with number $n+1$ defined by $a_{n+1} = 2 \cdot a_n$, $b_{n+1} = 0$, and $B_{n+1} = \beta$. Remark that this

β-instance admits the same set of solutions as the original instance as long as β is large enough, e.g. $\beta = a_n$ (cf. Observation 3). Consider a feasible solution to the β-instance where $\beta \leq a_n$. It holds that

$$2a_n x_{n+1} = a_{n+1}x_{n+1} \leq s + a_{n+1}x_{n+1} \leq B_{n+1} = \beta \leq a_n$$

which implies $x_{n+1} \leq \lfloor \frac{1}{2} \rfloor = 0$. However, if $x_{n+1} < 0$ then $s \geq a_{n+1} \cdot |x_{n+1}|$ and therefore the solution $s' = s + a_{n+1}x_{n+1}$ with $x'_{n+1} = 0$ and $x'_i = x_i - (a_{n+1}/a_i)x_{n+1}$ for all $i = 1, \ldots, n$ is better than s and $x'_{n+1} = 0$.

Thus we may assume generally that $x_{n+1} = 0$ which allows us to measure the value of s using the upper bound β. We use β to do a binary search in the interval $[0, a_n]$ using Algorithm 1 to check the β-instance for feasibility. The smallest possible value for β then states the optimum value and that proves Corollary 8. However, with additional ideas we are able to achieve strongly polynomial time. We want to give some helpful intuition first.

Clearly, after revealing the intervals in Q_1 with Algorithm 1 a straightforward idea is to try tracing them back to a small solution for s, but routing through the modulus operations appears to become a non-polynomial bottleneck.

However, the following idea is a first step to end up with a constraint aggregation approach. Given the projections $A^{[ab]}$ and $B^{[a]}$ of two sets $A, B \subseteq \mathbb{Z}$ one can compute the intersection $A^{[a]} \cap B^{[a]}$ in at least two ways; primitively we compute $(A^{[ab]})^{[a]} = A^{[a]}$ and then intersect it with $B^{[a]}$, but also we can intersect $A^{[ab]}$ with b translated copies $B^{[a]}, a + B^{[a]}, \ldots, (b-1)a + B^{[a]}$ of $B^{[a]}$ before computing the $[a]$-projection. In fact, the following lemma seems to be a characteristic property of modular arithmetic on sets.

▶ **Lemma 9.** *For all numbers $a, b \in \mathbb{Z}_{\geq 1}$ and sets $A, B \subseteq \mathbb{Z}$ it holds*

$$A^{[a]} \cap B^{[a]} = \left(A^{[ab]} \cap \bigcup_{i=0}^{b-1} (ia + B^{[a]}) \right)^{[a]}.$$

Proof. Let x be a number. Then it holds

$$\begin{aligned} x \in \left(A^{[ab]} \cap \bigcup_{i=0}^{b-1} (ia + B^{[a]}) \right)^{[a]} &\Leftrightarrow \exists y \in A^{[ab]} : y \in \bigcup_{i=0}^{b-1} (ia + B^{[a]}) \wedge x = y^{[a]} \\ &\Leftrightarrow \exists y \in A^{[ab]} : y^{[a]} \in B^{[a]} \wedge x = y^{[a]} \\ &\Leftrightarrow x \in A^{[a]} \cap B^{[a]} \end{aligned}$$

where the last equivalence follows from $(A^{[ab]})^{[a]} = A^{[a]}$. ◀

Since the right side can be written as the modular projection of a *union of intersections* we can find a sensible strengthening; in fact, for arbitrary sets $X, M_0, \ldots, M_{m-1}$ it holds that

$$\bigcup_{i=0}^{m-1} (X \cap M_i) = \bigcup_{i=0}^{m-1} (X \cap (M_i \setminus \bigcup_{j=0}^{i-1} (X \cap M_j))).$$

While the left-hand side may not, the right-hand side is always a *disjoint* union. Taking into account the modular projections this leads to the following corollary.

▶ **Corollary 10.** *For all numbers $a, b \in \mathbb{Z}_{\geq 1}$ and sets $A, B \subseteq \mathbb{Z}$ it holds $A^{[a]} \cap B^{[a]} = (\bigcup_{i=0}^{b-1} D_i)^{[a]}$ where $D_i = A^{[ab]} \cap Y_i$ and $Y_i = ia + (B^{[a]} \setminus \bigcup_{j=0}^{i-1} D_j^{[a]})$ for all $i = 0, \ldots, b-1$.*

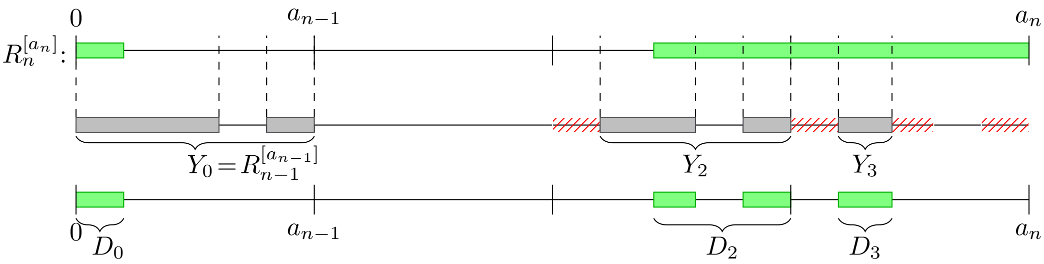

Figure 4 An example of four required intervals to represent $R_{n-1}^{[a_{n-1}]} \cap R_n^{[a_{n-1}]}$ in Lemma 13.

We will use Corollary 10 to aggregate constraints in order to reduce the problem size. The following observation gives a first bound for the smallest feasible solution $s_{\min}$.

▶ **Observation 11.** *For feasible instances it holds that $s_{\min} \in R_n^{[a_n]}$.*

This is true since in the harmonic case $s_{\min} < \operatorname{lcm}(a_1, \ldots, a_n) = a_n$ due to Observation 3 which then implies that $s_{\min} = s_{\min}^{[a_n]} \in R_n^{[a_n]}$ using Observation 2. Motivated by Observation 11 the idea is to search for $s_{\min}$ in the modular projection $R_n^{[a_n]}$ by aggregating the penultimate constraint $n-1$ into the last constraint n. In fact, the number of required intervals to represent both constraints can be bounded by a constant. A fine-grained construction then enforces the algorithm to efficiently iterate the feasibility test on aggregated instances to find the optimum value.

▶ **Theorem 12.** *For feasible instances $s_{\min}$ can be computed in time $\mathcal{O}(n^3)$.*

Remark that the set of feasible solutions for the last two constraints is $S_{n-1} = R_{n-1}^{[a_{n-1}]} \cap (R_n^{[a_n]})^{[a_{n-1}]} = R_{n-1}^{[a_{n-1}]} \cap R_n^{[a_{n-1}]}$. Therefore, the next lemma states the crucial argument of the algorithm.

▶ **Lemma 13.** *The intersection $R_{n-1}^{[a_{n-1}]} \cap R_n^{[a_{n-1}]}$ can always be represented by the disjoint union $U \subseteq R_n^{[a_n]}$ of only constant many intervals in $R_n^{[a_n]}$ such that*
(a) *$U^{[a_{n-1}]} = R_{n-1}^{[a_{n-1}]} \cap R_n^{[a_{n-1}]}$ and*
(b) *$u \equiv r \pmod{a_{n-1}}$ implies $u \le r$ for all $u \in U$, $r \in R_n^{[a_n]}$.*

Here the former property states that indeed the intervals in U are a proper representation for the last two constraints. The important property is the latter; in fact, it ensures that U is the best possible representation in the sense that U consists of the *smallest* intervals possible (see Figure 4).

Proof of Lemma 13. (a). By defining $D_i = Y_i \cap R_n^{[a_n]}$ and

$$Y_i = ia_{n-1} + \Big(R_{n-1}^{[a_{n-1}]} \setminus \bigcup_{j=0}^{i-1} D_j^{[a_{n-1}]}\Big)$$

for all $i \in \{0, \ldots, a_n/a_{n-1} - 1\}$ Corollary 10 proves the claim (cf. Figure 4). (b) follows by construction.

It remains to show that $\bigcup_i D_i$ is the union of only constant many disjoint intervals. Apparently, the intervals are disjoint by construction.

We claim that there are at most three non-empty sets D_i. Assume there are at least four non-empty translates D_i, namely D_i, D_j, D_k, D_ℓ. Then, since R_n is an interval it holds for at least two $p, q \in \{i, j, k, \ell\}$ that the *full interval translates* $F_p = [pa_{n-1}, (p+1)a_{n-1})$ and $F_q = [qa_{n-1}, (q+1)a_{n-1})$ are subsets of $R_n^{[a_n]}$. For p (and also for q) we get

$$D_p^{[a_{n-1}]} = (\underbrace{Y_p}_{\subseteq F_p} \cap R_n^{[a_n]})^{[a_{n-1}]} = Y_p^{[a_{n-1}]} = R_{n-1}^{[a_{n-1}]} \setminus \bigcup_{j=0}^{p-1} D_j^{[a_{n-1}]}$$

which implies with $\bigcup_{j=0}^{p-1} D_j^{[a_{n-1}]} \subseteq R_{n-1}^{[a_{n-1}]}$ that

$$\bigcup_{j=0}^{p} D_j^{[a_{n-1}]} = D_p^{[a_{n-1}]} \cup \bigcup_{j=0}^{p-1} D_j^{[a_{n-1}]} = R_{n-1}^{[a_{n-1}]}.$$

Then it follows $\bigcup_{j=0}^{p} D_j^{[a_{n-1}]} = R_{n-1}^{[a_{n-1}]} = \bigcup_{j=0}^{q} D_j^{[a_{n-1}]}$. W.l.o.g. let $p < q$. Then $D_q = Y_q \cap R_n^{[a_n]}$ is empty since

$$Y_q = qa_{n-1} + \left(R_{n-1}^{[a_{n-1}]} \setminus \bigcup_{j=0}^{q-1} D_j^{[a_{n-1}]} \right) \subseteq qa_{n-1} + \left(R_{n-1}^{[a_{n-1}]} \setminus R_{n-1}^{[a_{n-1}]} \right)$$

is empty and we have a contradiction.

Using the same case distinctions as in the proof of Lemma 4 one can show that each set D_i consist of at most two intervals. Therefore, all the non-empty sets D_i consist of at most $3 \cdot 2 = 6$ intervals in total. In fact, one can improve this bound to a total number of at most 4 intervals (see Figure 4) by a more sophisticated case distinction. ◀

This admits an algorithm using an aggregation argument as follows. For constraints n and $n-1$ we use Lemma 13 to compute disjoint intervals $E_1, \ldots, E_k \subseteq R_n^{[a_n]}$ (representing the constraints n and $n-1$) where $k \leq C$ for a small constant C. If $k \geq 1$ then use Algorithm 1 to check the feasibility of the instances $\mathcal{I}_1, \ldots, \mathcal{I}_k$ defined by

$$(\mathcal{I}_j) \qquad \min \{ \, s \mid s^{[a_i]} \in R_i^{[a_i]} \forall i = 1, \ldots, n-2, \; s^{[a_n]} \in E_j^{[a_n]}, s \in \mathbb{Z}_{\geq 0} \, \} \, .$$

If none of the instances $\mathcal{I}_1, \ldots, \mathcal{I}_k$ admits a solution then the original instance can not be feasible. Assume that there is at least one feasible instance. Now, since $E_1, \ldots, E_k$ are disjoint exactly one of them contains the optimum value for s. W.l.o.g. assume that $E_1 < \cdots < E_k$. Then there is a smallest index j such that $\mathcal{I}_j$ is feasible and we solve $\mathcal{I}_j$ recursively to find the optimum value. Together this yields an algorithm running in time $n \cdot C \cdot \mathcal{O}(n^2) = \mathcal{O}(n^3)$.

4 Uniprocessor Real-Time Scheduling

In real-time systems an important question is to ask for the *worst-case response time* of a task system. While the complexity is pseudo-polynomial in general [28], Nguyen et al. proposed a new algorithm [25] to compute it in polynomial time for preemptive sporadic tasks $\tau_1, \ldots, \tau_n$ with harmonic periods $T_i \geq 0$ and job processing times $C_i \geq 0$ running on a uniprocessor platform. The worst-case response time is the first point in time where $t = C_n + \sum_{i=1}^{n-1} C_i \cdot \lceil t/T_i \rceil$. Be aware that they assume the harmonic property in the opposite direction, i.e. $T_i/T_{i+1} \in \mathbb{Z}$. Their algorithm even allows the task execution to be delayed by some release jitter J_i. However, their algorithm depends on a heuristic component which may fail to compute correct solutions [25, Section 5.5, 6]. In fact, the fundamental computation problem can be expressed as a BMS instance which immediately implies a robust approach in time $\mathcal{O}(n^3)$ with our algorithm. Nevertheless, it can be solved even more efficiently in time $\mathcal{O}(n)$ which we describe here. The overall result will be the following theorem.

▶ **Theorem 14.** *The worst-case response time of a harmonic task real-time system with constrained release jitter can be computed in polynomial time.*

We adapt the notation of Nguyen et al. and extend it to our needs. The jobs of task τ_i have the processing time C_i and we define $c_i = \sum_{t=i+1}^{n-1} C_t$ to accumulate the last of them. The utilization of task τ_i is denoted by $U_i = C_i/T_i$ and it holds that $\sum_{t=1}^{n-1} U_t < 1$. In [25, Section 5.4.1] Nguyen et al. describe that also $x_1 = 1$ may be assumed. The system to solve (eq. (55), (56)) is described in [25, Section 5.5]:

$$\begin{aligned}\min \{\, x_n \mid\ & J_i + T_i x_i \leq J_n + T_n x_n, \\ & J_n + T_n x_n - c_i \leq J_i + T_i x_i \ \ \forall i \leq n-1 \,\}\end{aligned} \tag{2}$$

which can be formulated as the following BMS instance:

$$\min \left\{ x_n \,\middle|\, \left\lceil \frac{J_i - J_n}{T_n} \right\rceil \leq x_n - \frac{T_i}{T_n} x_i \leq \left\lfloor \frac{J_i - J_n + c_i}{T_n} \right\rfloor \quad \forall i \leq n-1 \right\} \tag{3}$$

▶ **Lemma 15.** *If $i < j \leq n$ and $(c_i + c_j)/T_j < 1$ then in terms of variable x_i there is at most one feasible value for variable x_j.*

Proof. If $j < n$ then by combining the constraints for i and j in (2) we find

$$\begin{aligned} T_i x_i + J_i - J_n \ &\leq\ T_j x_j + J_j - J_n + c_j \quad \text{ and} \\ T_j x_j + J_j - J_n \ &\leq\ T_i x_i + J_i - J_n + c_i \end{aligned}$$

which with the harmonic property and the integrality of x_j yields

$$\frac{T_i}{T_j} x_i + \left\lceil \frac{J_i - J_j - c_j}{T_j} \right\rceil \leq x_j \leq \frac{T_i}{T_j} x_i + \left\lfloor \frac{J_i - J_j + c_i}{T_j} \right\rfloor. \tag{4}$$

However, if $j = n$ then $c_j = \sum_{t=n+1}^{n-1} C_t = 0$ and thus (4) follows from (2) too (cf. (3)). Now by simply dropping the roundings we obtain in both cases that

$$\frac{T_i}{T_j} x_i + \left\lfloor \frac{J_i - J_j + c_i}{T_j} \right\rfloor - \left(\frac{T_i}{T_j} x_i + \left\lceil \frac{J_i - J_j - c_j}{T_j} \right\rceil \right) \leq \frac{c_i + c_j}{T_j} < 1$$

which proves the claim. ◀

According to (4) we define interval bounds $\ell_j^{(i)}(z)$ and $u_j^{(i)}(z)$ to denote the feasible values for variable x_j in terms of variable x_i where z states a value for variable x_i, i.e.

$$\ell_j^{(i)}(z) = \frac{T_i}{T_j} z + \left\lceil \frac{J_i - J_j - c_j}{T_j} \right\rceil \qquad \text{and} \qquad u_j^{(i)}(z) = \frac{T_i}{T_j} z + \left\lfloor \frac{J_i - J_j + c_i}{T_j} \right\rfloor.$$

Thus, (4) is equivalent to $x_j \in [\ell_j^{(i)}(x_i), u_j^{(i)}(x_i)]$ and if $(c_i + c_j)/T_j < 1$ then it either holds that $\ell_j^{(i)}(x_i) = x_j = u_j^{(i)}(x_i)$ or there is no solution at all.

Fortunately, there is always a sequence of variables such that the value of every next variable can be determined by the value of the current variable. The following lemma is crucial.

▶ **Lemma 16.** *If $i < n$ and $k = \max\{\, t \leq n \mid T_{i+1} = T_t \,\}$ then there is at most one feasible value for variable x_k.*

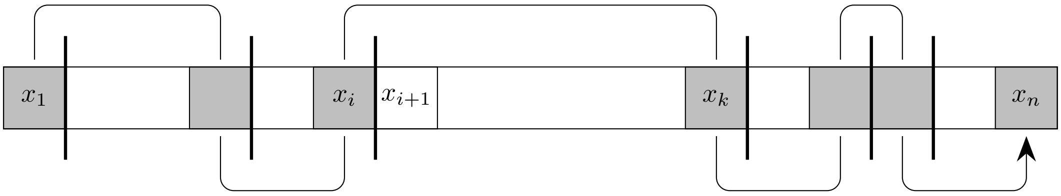

Figure 5 The variable revealing flow with vertical lines between blocks of equal periods.

Proof. If $k < n-1$ then it holds by the harmonic property and the maximality of k that $T_k \geq 2T_{k+1} \geq 2T_{k+2} \geq \cdots \geq 2T_{n-1}$ and thus $T_t/T_k \leq 1/2$ for all $t = k+1, \ldots, n-1$. Hence,

$$\frac{c_i + c_k}{T_k} = \sum_{t=i+1}^{n-1} U_t \frac{T_t}{T_k} + \sum_{t=k+1}^{n-1} U_t \frac{T_t}{T_k} = \sum_{t=i+1}^{k} U_t \underbrace{\frac{T_t}{T_k}}_{=1} + 2 \sum_{t=k+1}^{n-1} U_t \underbrace{\frac{T_t}{T_k}}_{\leq 1/2} \leq \sum_{t=i+1}^{n-1} U_t < 1.$$

If otherwise $k \geq n-1$ then $c_k = 0$ and hence

$$\frac{c_i + c_k}{T_k} = \frac{c_i}{T_k} = \sum_{t=i+1}^{n-1} U_t \underbrace{\frac{T_t}{T_k}}_{=1} = \sum_{t=i+1}^{n-1} U_t < 1.$$

By Lemma 15 this proves the claim. ◀

This gives rise to the following algorithm. By iterating Lemma 16 and starting with $x_1 = 1$ we can reveal the last variable of each block of indices of equal periods (cf. Figure 5). Finally, this reveals the variable x_n and we only need to assure that the value of x_n admits feasible values for variables which are not revealed so far. Apparently we may restate the constraints of (2) as

$$\left\lceil \frac{J_n - J_j - c_j + T_n x_n}{T_j} \right\rceil \leq x_j \leq \left\lfloor \frac{J_n - J_j + T_n x_n}{T_j} \right\rfloor \qquad \forall j = 1, \ldots, n-1.$$

Therefore, we can simply compare these bounds to assure the existence of a feasible value for each variable x_j. See Algorithm 2 for a formal description.

Algorithm 2 Variable revealing flow.

procedure REVEAL
  $x_1 \leftarrow 1$
  $k \leftarrow 1$
  while $k < n$ **do**
    $i \leftarrow k$
    $k \leftarrow \max\{ t \leq n \mid T_{i+1} = T_t \}$
    if $\ell_k^{(i)}(x_i) \neq u_k^{(i)}(x_i)$ **then**
      return -1
    else
      $x_k \leftarrow \ell_k^{(i)}(x_i)$ ▷ Lemma 16
  for $j = 1, \ldots, n-1$ **do**
    if $\left\lceil \frac{J_n - J_j - c_j + T_n x_n}{T_j} \right\rceil > \left\lfloor \frac{J_n - J_j + T_n x_n}{T_j} \right\rfloor$ **then** ▷ no feasible solution for x_j
      return -1
  return x_n

▶ **Observation 17.** *By a more sophisticated investigation the number of index blocks of equal periods can be bounded by a constant and thus, the* **while** *loop reveals x_n in constant time. Therefore, the final feasibility test appears to be the only computational bottleneck.*

References

1 Manindra Agrawal and Somenath Biswas. Primality and identity testing via chinese remaindering. In *Proc. FOCS 1999*, pages 202–209, 1999.

2 Saoussen Anssi, Stefan Kuntz, Sébastien Gérard, and François Terrier. On the gap between schedulability tests and an automotive task model. *J. Syst. Archit.*, 59(6):341–350, 2013.

3 Reuven Bar-Yehuda and Dror Rawitz. Efficient algorithms for integer programs with two variables per constraint. *Algorithmica*, 29(4):595–609, 2001.

4 Sanjoy K. Baruah, Louis E. Rosier, and Rodney R. Howell. Algorithms and complexity concerning the preemptive scheduling of periodic, real-time tasks on one processor. *Real-Time Systems*, 2(4):301–324, 1990.

5 Vincenzo Bonifaci, Alberto Marchetti-Spaccamela, Nicole Megow, and Andreas Wiese. Polynomial-time exact schedulability tests for harmonic real-time tasks. In *Proc. RTSS 2013*, pages 236–245, 2013.

6 Michele Conforti, Gerard Cornuéjols, and Giacomo Zambelli. *Integer Programming.* Springer, 2014.

7 Michele Conforti, Marco Di Summa, and Laurence A. Wolsey. The mixing set with flows. *SIAM J. Discrete Math.*, 21(2):396–407, 2007.

8 Michele Conforti, Marco Di Summa, and Laurence A. Wolsey. The mixing set with divisible capacities. In *Proc. IPCO 2008*, pages 435–449, 2008.

9 Michele Conforti and Giacomo Zambelli. The mixing set with divisible capacities: A simple approach. *Oper. Res. Lett.*, 37(6):379–383, 2009.

10 Jana Cslovjecsek, Friedrich Eisenbrand, Christoph Hunkenschröder, Lars Rohwedder, and Robert Weismantel. Block-structured integer and linear programming in strongly polynomial and near linear time, 2020 (Manuscript). `arXiv:2002.07745`.

11 Friedrich Eisenbrand, Karthikeyan Kesavan, Raju S. Mattikalli, Martin Niemeier, Arnold W. Nordsieck, Martin Skutella, José Verschae, and Andreas Wiese. Solving an avionics real-time scheduling problem by advanced ip-methods. In Mark de Berg and Ulrich Meyer, editors, *Proc. ESA 2010*, volume 6346 of *Lecture Notes in Computer Science*, pages 11–22, 2010.

12 Friedrich Eisenbrand and Günter Rote. Fast 2-variable integer programming. In *Proc. IPCO 2001*, pages 78–89, 2001.

13 Friedrich Eisenbrand and Thomas Rothvoß. New hardness results for diophantine approximation. In *Proc. APPROX 2009*, pages 98–110, 2009.

14 Oded Goldreich, Dana Ron, and Madhu Sudan. Chinese remaindering with errors. In *Proc. STOC 1999*, pages 225–234, 1999.

15 Oktay Günlük and Yves Pochet. Mixing mixed-integer inequalities. *Math. Program.*, 90(3):429–457, 2001.

16 Venkatesan Guruswami, Amit Sahai, and Madhu Sudan. "soft-decision" decoding of chinese remainder codes. In *Proc. FOCS 2000*, pages 159–168, 2000.

17 Raymond Hemmecke, Shmuel Onn, and Lyubov Romanchuk. n-fold integer programming in cubic time. *Math. Program.*, 137(1-2):325–341, 2013.

18 Raymond Hemmecke and Rüdiger Schultz. Decomposition of test sets in stochastic integer programming. *Math. Program.*, 94(2-3):323–341, 2003.

19 Klaus Jansen, Alexandra Lassota, and Lars Rohwedder. Near-linear time algorithm for n-fold ilps via color coding. In *Proc. ICALP 2019*, pages 75:1–75:13, 2019.

20 Kim-Manuel Klein. About the complexity of two-stage stochastic ips. In Daniel Bienstock and Giacomo Zambelli, editors, *Proc. IPCO 2020*, volume 12125 of *Lecture Notes in Computer Science*, pages 252–265, 2020.

21 Donald E. Knuth. *The Art of Computer Programming, Volume II: Seminumerical Algorithms, 2nd Edition.* Prentice Hall, 1981.

22 J. C. Lagarias. The computational complexity of simultaneous diophantine approximation problems. *SIAM J. Comput.*, 14(1):196–209, 1985.

23 Joseph Y.-T. Leung and Jennifer Whitehead. On the complexity of fixed-priority scheduling of periodic, real-time tasks. *Perform. Evaluation*, 2(4):237–250, 1982.

24 Andrew J. Miller and Laurence A. Wolsey. Tight formulations for some simple mixed integer programs and convex objective integer programs. *Math. Program.*, 98(1-3):73–88, 2003.

25 Thi Huyen Chau Nguyen, Werner Grass, and Klaus Jansen. Exact polynomial time algorithm for the response time analysis of harmonic tasks with constrained release jitter, 2019 (Manuscript). `arXiv:1912.01161`.

26 Yves Pochet and Laurence A. Wolsey. *Production Planning by Mixed Integer Programming (Springer Series in Operations Research and Financial Engineering).* Springer, Berlin, Heidelberg, 2006.

27 Chi-Sheng Shih, Sathish Gopalakrishnan, Phanindra Ganti, Marco Caccamo, and Lui Sha. Template-based real-time dwell scheduling with energy constraint. In *Proc. RTAS 2003*, page 19, 2003.

28 Mikael Sjödin and Hans Hansson. Improved response-time analysis calculations. In *Proc. RTSS 1998*, pages 399–408, 1998.

29 Yang Xu, Anton Cervin, and Karl-Erik Årzén. Lqg-based scheduling and control co-design using harmonic task periods. Technical report, Department of Automatic Control, Lund Institute of Technology, Lund University, August 2016. URL: `https://lup.lub.lu.se/search/ws/files/10751577/bare_conf.pdf`.

30 Ming Zhao and Ismael R. de Farias Jr. The mixing-mir set with divisible capacities. *Math. Program.*, 115(1):73–103, 2008.

A Hardness of BMS

We reduce from Directed Diophantine Approximation with *rounding down*. For any vector $v \in \mathbb{R}^n$ let $\lfloor v \rfloor$ denote the vector where each component is rounded down, i.e. $(\lfloor v \rfloor)_i = \lfloor v_i \rfloor$ for all $i \leq n$.

> Directed Diophantine Approximation with rounding down (DDA$^{\downarrow}$)
> Given: $\alpha_1, \dots, \alpha_n \in \mathbb{Q}_+$, $N \in \mathbb{Z}_{\geq 1}$, $\varepsilon \in \mathbb{Q}$, $0 < \varepsilon < 1$
> Decide whether there is a $Q \in \{1, \dots, N\}$ such that $\|Q\alpha - \lfloor Q\alpha \rfloor\|_\infty \leq \varepsilon$.

Eisenbrand and Rothvoß proved that DDA$^{\downarrow}$ is NP-hard [13]. In fact, every instance of DDA$^{\downarrow}$ can be expressed as a BMS instance, which yields the following theorem.

▶ **Theorem 18.** BMS *is* NP*-hard (even if $b_i = 0$ for all i with $a_i \neq 0$).*

Proof. Write $\alpha_i = \beta_i/\gamma_i$ for integers $\beta_i \geq 0, \gamma_i \geq 1$ and set $\lambda = \prod_j \beta_j$. Then $\lambda/\alpha_i = (\lambda/\beta_i)\gamma_i \geq 0$ is integer. Let $\mathcal{M}$ denote the following instance of BMS:

$$
\begin{aligned}
0 &\leq Q' - (\lambda/\alpha_i) \cdot y_i \leq \lfloor (\lambda/\alpha_i) \cdot \varepsilon \rfloor && \forall i = 1, \dots, n && (5)\\
\lambda &\leq Q' - 0 \cdot y_{n+1} \leq \lambda \cdot N && && (6)\\
0 &\leq Q' - \lambda \cdot y_{n+2} \leq 0 && && (7)\\
& Q', y_i \in \mathbb{Z} && \forall i = 1, \dots, n+2
\end{aligned}
$$

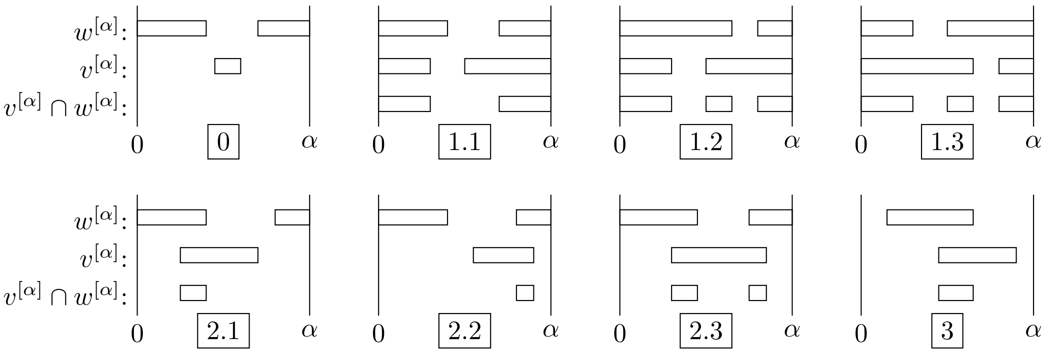

Figure 6 Examples for the cases of the case distinction in the proof of Lemma 4.

So let $Q \in \{1, \ldots, N\}$ with $\|Q\alpha - \lfloor Q\alpha \rfloor\|_\infty \leq \varepsilon$ be given. We obtain readily that $Q' = \lambda Q$ and $y = (\lfloor Q\alpha_1 \rfloor, \ldots, \lfloor Q\alpha_n \rfloor, 0, Q)$ defines a solution of $\mathcal{M}$ since

$$0 \leq Q\alpha_i - \lfloor Q\alpha_i \rfloor \leq \varepsilon \quad \text{if and only if} \quad 0 \leq \underbrace{\lambda Q - (\lambda/\alpha_i) \cdot \lfloor Q\alpha_i \rfloor}_{\in \mathbb{Z}} \leq (\lambda/\alpha_i) \cdot \varepsilon.$$

Vice-versa let (Q', y) be a solution to $\mathcal{M}$. We see that (5) implies that

$$0 \leq Q' - (\lambda/\alpha_i) \cdot y_i \leq \lfloor (\lambda/\alpha_i) \cdot \varepsilon \rfloor \leq (\lambda/\alpha_i) \cdot \varepsilon$$

and by (7) we get $Q' = \lambda \cdot y_{n+2}$ which then implies $0 \leq y_{n+2}\alpha_i - y_i \leq \varepsilon < 1$ for all $i \leq n$. Now, since y_i is integer, there can be only one value for y_i, i.e. $y_i = \lfloor y_{n+2}\alpha_i \rfloor$. By $Q' = \lambda \cdot y_{n+2}$ and (6) we get $y_{n+2} \in \{1, \ldots, N\}$ and by setting $Q = y_{n+2}$ this yields $\|Q\alpha - \lfloor Q\alpha \rfloor\|_\infty \leq \varepsilon$ and that proves the claim. ◀

B Omitted proofs

Proof of Lemma 4. We do a case distinction (see Figure 6) as follows. We only look at the non-trivial case, i.e. $v^{[\alpha]} \cap w^{[\alpha]} \notin \{\varnothing, v^{[\alpha]}, w^{[\alpha]}\}$, which especially implies $|v| < \alpha$ and $|w| < \alpha$.

We start with the case that neither $v^{[\alpha]}$ nor $w^{[\alpha]}$ is an interval, i.e. $u_v^{[\alpha]} < \ell_v^{[\alpha]}$ and $u_w^{[\alpha]} < \ell_w^{[\alpha]}$. Then it cannot be that $u_w^{[\alpha]} \geq \ell_v^{[\alpha]}$ and $u_v^{[\alpha]} \geq \ell_w^{[\alpha]}$ since that implies $\ell_v^{[\alpha]} \leq u_w^{[\alpha]} < \ell_w^{[\alpha]} \leq u_v^{[\alpha]}$. Hence, there are three cases as follows.

Case 1.1. $u_w^{[\alpha]} < \ell_v^{[\alpha]}$ and $u_v^{[\alpha]} < \ell_w^{[\alpha]}$. Then the intersection equals

$$\begin{aligned}[0, \min\{u_v^{[\alpha]}, u_w^{[\alpha]}\}] \,\dot\cup\, [\max\{\ell_v^{[\alpha]}, \ell_w^{[\alpha]}\}, \alpha) &= [\max\{\ell_v^{[\alpha]}, \ell_w^{[\alpha]}\}, \alpha + \min\{u_v^{[\alpha]}, u_w^{[\alpha]}\}]^{[\alpha]} \\ &= \psi_\alpha(v, w)^{[\alpha]}.\end{aligned}$$

Case 1.2. $u_w^{[\alpha]} \geq \ell_v^{[\alpha]}$ and $u_v^{[\alpha]} < \ell_w^{[\alpha]}$. Then the intersection equals

$$[0, u_v^{[\alpha]}] \,\dot\cup\, [\ell_v^{[\alpha]}, u_w^{[\alpha]}] \,\dot\cup\, [\ell_w^{[\alpha]}, \alpha) = [\ell_v^{[\alpha]}, u_w^{[\alpha]}] \,\dot\cup\, [\ell_w^{[\alpha]}, \alpha + u_v^{[\alpha]}]^{[\alpha]} = \varphi_\alpha(v, w) \,\dot\cup\, \psi_\alpha(v, w)^{[\alpha]}.$$

Case 1.3. $u_w^{[\alpha]} < \ell_v^{[\alpha]}$ and $u_v^{[\alpha]} \geq \ell_w^{[\alpha]}$. By symmetry we get $v^{[\alpha]} \cap w^{[\alpha]} = \varphi_\alpha(w, v) \dot\cup \psi_\alpha(v, w)^{[\alpha]}$.

Now, w.l.o.g. assume that $v^{[\alpha]}$ is an interval, i.e. $\ell_v^{[\alpha]} \leq u_v^{[\alpha]}$, while $w^{[\alpha]}$ consists of two intervals, i.e. $u_w^{[\alpha]} < \ell_w^{[\alpha]}$. Then there are three cases as follows.

CASE 2.1. $\ell_v^{[\alpha]} \leq u_w^{[\alpha]} < u_v^{[\alpha]} < \ell_w^{[\alpha]}$. Then the intersection equals $[\ell_v^{[\alpha]}, u_w^{[\alpha]}] = \varphi_\alpha(v, w)$.
CASE 2.2. $u_w^{[\alpha]} < \ell_v^{[\alpha]} < \ell_w^{[\alpha]} \leq u_v^{[\alpha]}$. Then the intersection equals $[\ell_w^{[\alpha]}, u_v^{[\alpha]}] = \varphi_\alpha(w, v)$.
CASE 2.3. $\ell_v^{[\alpha]} \leq u_w^{[\alpha]} < \ell_w^{[\alpha]} \leq u_v^{[\alpha]}$. Then the intersection is

$$[\ell_v^{[\alpha]}, u_w^{[\alpha]}] \,\dot\cup\, [\ell_w^{[\alpha]}, u_v^{[\alpha]}] = \varphi_\alpha(v, w) \,\dot\cup\, \varphi_\alpha(w, v).$$

Clearly, if both $v^{[\alpha]}$ and $w^{[\alpha]}$ are intervals (CASE 3) (which are not disjoint) then their intersection is either $\varphi_\alpha(v, w)$ or $\varphi_\alpha(w, v)$. ◀