Elastic Scheduling for Harmonic Task Systems

Marion Sudvarg, Ao Li, Daisy Wang,
Sanjoy Baruah, Jeremy Buhler, Chris Gill, Ning Zhang
Department of Computer Science and Engineering
Washington University in St. Louis
(msudvarg, ao, w.yanwang, baruah, jbuhler, cdgill, zhang.ning)@wustl.edu

Pontus Ekberg
Department of Information Technology
Uppsala University
pontus.ekberg@it.uu.se

***Abstract*—Elastic scheduling is a framework to reduce task utilizations (often by increasing periods) in response to system overload. This paper extends elastic scheduling to uniprocessor scheduling of implicit-deadline task sets for which periods must remain *harmonic*. We argue that for tasks with periods constrained to continuous intervals, the problem of selecting harmonic periods from those intervals is unlikely to have a polynomial time solution. However, we outline an approach that is pseudo-polynomial in the range of acceptable periods. We then show that the problem of *elastic scheduling* is $\mathcal{NP}$-hard with harmonic constraints. Nonetheless, if a total order is imposed on task periods (a natural restriction in many applications with execution pipelines that synchronize input data sources), the problem can be reduced offline to a lookup table, enabling polynomial-time online adaptation if available CPU bandwidth changes. We implement the proposed algorithm in two real-world applications: the Fast Integrated Mobility Spectrometer (FIMS) and ORB-SLAM3. We demonstrate that elastic scheduling allows FIMS to adjust its execution to avoid missing deadlines on a SWaP-constrained computational platform, and that it improves ORB-SLAM3's localization results by as much as $10.4\times$ when available CPU bandwidth changes dynamically during runtime.**

***Index Terms*—real-time systems, elastic scheduling, harmonic tasks, period assignment, harmonic periods, period ranges**

I. Introduction

Elastic real-time scheduling models provide a framework for dynamic adaptation of task utilizations in response to system overload. The model originally proposed in [1], [2] by Buttazzo et al. was formulated for uniprocessor scheduling of implicit-deadline tasks. Each task is assigned a continuous interval from which its period may be selected, as well as an elasticity parameter that "specifies the flexibility of the task to vary its utilization" [1]. If schedulable, each task executes at its minimum allowed period; otherwise, each task's utilization is "compressed" proportionally to its elastic constant by increasing its period until the total utilization no longer exceeds the schedulable bound of the system (or until the task's period would exceed its assigned interval).

This paper is the first to extend the elastic scheduling model to task systems for which periods are additionally constrained to be *harmonic*. Many control systems, such as those found in robotics applications [3] and real-time hybrid structural simulation [4], demand harmonic rates among their constituent tasks. In applications that capture and process frames from multiple sensing devices to be synchronized in backend processing tasks, such as simultaneous localization and mapping (SLAM) [5] and real-time mobile spectrometry [6], harmonic task periods guarantee consistent temporal alignment [7]. Furthermore, task sets with harmonic periods have hyperperiods equal to the largest period [8], [9], which reduces scheduling table sizes in time triggered systems [10] and constrains the test set in processor demand analysis [11].

Selecting harmonic periods from within acceptable intervals is nontrivial. Nasri et al. [12] formalized the problem, and proposed an approach to solve it in time linear in the number of tasks for restricted cases. In general, though, the algorithm's running time "can exponentially grow." In [13], Nasri and Fohler identified another restriction of the problem that can be solved in polynomial time, but they provide "no guarantee for reasonable computational complexity" in general.

In this work, we reason about the problem of assigning task periods from continuous intervals such that (i) the objective function defined for elastic scheduling by Chantem et al. in [14], [15] is minimized, (ii) periods remain harmonic, and (iii) schedulability is guaranteed. We call this the **harmonic elastic problem**. This paper makes the following contributions:

Complexity Results: In §IV, we show that for n tasks and a parameter k bounding the ratio of the task periods, finding a harmonic period assignment (if one exists within the allowed intervals) is unlikely to be solvable in time polynomial in n and $\log k$, but that there exists a *pseudo-polynomial* algorithm to do so in general. We also prove that the harmonic elastic problem on a uniprocessor is $\mathcal{NP}$-hard even for fixed k.

The Ordered Harmonic Elastic Problem: In §V, we propose a two-part algorithm to find an optimal solution to the harmonic elastic problem with an *a priori* order imposed over task periods. This is a natural restriction in many systems, such as multi-time-stepping decomposition of a real-time hybrid simulation (RTHS) [16] for natural hazards engineering, and applications such as ORB-SLAM [17] where frontend data collection tasks capture and process frames from sensing devices which must be aggregated downstream. Our algorithm searches offline for all projected harmonic intervals (PHIs) that can be constructed within the allowed period intervals, constructing a lookup table that identifies the *optimal* PHI for each possible utilization bound.[1] Despite being exponential in $\log k$, we demonstrate that in practice this algorithm is feasible for reasonable values of n and k and the lookup table remains small. During online execution, if available utilization

[1] Intuitively, for a given sequence of harmonic multipliers, the corresponding PHI describes the largest continuous range such that if the first task in period order is assigned a period from that range, all periods assigned as corresponding integer multiples remain within the allowed intervals for each task. We define PHI more formally in Definition 4 of §V.

2642-7346/24/$31.00 ©2024 IEEE
DOI 10.1109/RTAS61025.2024.00034

changes (e.g., due to interference from background processes, or the arrival of aperiodic workloads), binary search allows polynomial-time reassignment of task periods.

Real-World Applications: We apply this approach to two real-world applications: the Fast Integrated Mobility Spectrometer (FIMS) [6] and the ORB-SLAM3 [18] simultaneous localization and mapping system. FIMS, described in §VI-B, performs real-time atmospheric aerosol monitoring, aggregating and synchronizing harmonic inputs in a backend matrix inversion task that translates detected particle coordinates into a particle size distribution. Towards future deployment onboard a small UAV, we run FIMS *on a single core* of the Raspberry Pi 4, using our harmonic elastic scheduling model to prevent overload. We find that, by adjusting task periods, we can guarantee temporal alignment and avoid deadline misses, even when FIMS is limited to consuming only a fraction of that core's bandwidth. In ORB-SLAM3, dataflow and synchronization requirements impose a harmonic total ordering over the task periods. When computational resources are insufficient to guarantee completion of all tasks, frames may be dropped or desynchronized, giving increasingly erroneous results [19]. Our harmonic elastic scheduling model allows it to adjust task periods in response to changes in available CPU bandwidth when executing concurrently with other tasks on a *single core*, achieving a $10.4\times$ reduction in relative translational error (RTE) using this approach compared to its baseline execution.

II. Background and System Model

Elastic Scheduling: The elastic recurrent real-time workload model [1], [2] provides a framework for managing overload without suspending tasks or dropping jobs by reducing ("compressing") individual tasks' utilizations until the total utilization no longer exceeds the schedulable bound. Each task's utilization is likened to a spring whose elasticity reflects the task's ability to adapt its utilization to a lower quality of service. If the total length of the springs, placed end to end, exceeds some desired length (the utilization bound), a compressive force is applied to the system. Each spring (and corresponding utilization) is compressed proportionally to its elasticity until the total utilization no longer exceeds the bound, or until the individual task reaches its minimum serviceable utilization, by extending individual task periods.

More formally, the elastic model for implicit-deadline tasks on a uniprocessor [1], [2] characterizes each task $\tau_i{=}(C_i, T_i^{\min}, T_i^{\max}, T_i, E_i)$ by five non-negative parameters: C_i (the task's worst-case execution time); $T_i^{\min}$ (the task's minimum period, i.e., its nominal value when executing at the desired rate in an uncompressed state); $T_i^{\max}$ (its maximum period, beyond which correct behavior is not maintained); T_i (the task's assigned period, $T_i^{\min} \leq T_i \leq T_i^{\max}$); and E_i (a constant representing "the flexibility of the task to vary its utilization" [1]). From these parameters, corresponding values $U_i{=}C_i/T_i$, $U_i^{\min}{=}C_i/T_i^{\max}$, and $U_i^{\max}{=}C_i/T_i^{\min}$ can be derived. A task system $\Gamma = \{\tau_1, \ldots, \tau_n\}$ has a desired utilization U_D representing the utilization bound given by the scheduling algorithm in use. If the system is overloaded, the elastic model assigns a period T_i to each task τ_i by reducing its utilization from $U_i^{\max}$ in proportion to its elasticity E_i until either (i) the total utilization no longer exceeds U_D, or (ii) the task's period reaches $T_i^{\max}$. In [20], Sudvarg et al. presented an algorithm to achieve this in quasilinear time over the number of tasks, or in linear time for admission of a new task.

The elastic model has been extended to multiprocessor scheduling of sequential, implicit-deadline tasks [21] and to EDF [22] and fixed priority [23] scheduling of constrained-deadline tasks. However, some scheduling models impose constraints where proportional compression no longer makes sense – e.g., under federated scheduling [24] where every parallel task executes on dedicated cores, or (as in this paper) when periods must remain harmonic. To allow further generalization, Chantem et al. demonstrated in [14], [15] that utilizations satisfying the elastic scheduling model could be found by solving a constrained optimization problem:

$$\min_{U_i} \quad \sum_{i=1}^{n} \frac{1}{E_i}(U_i^{\max} - U_i)^2 \tag{1a}$$

$$\text{s.t.} \quad \sum_{i} U_i \leq U_D \quad \text{and} \tag{1b}$$

$$\forall_i, \quad U_i^{\min} \leq U_i \leq U_i^{\max} \tag{1c}$$

(We refer to (1a) as the "elastic objective.") By modifying the schedulability constraint (1b), Orr et al. developed elastic frameworks for federated scheduling of parallel tasks having periods [25] and workloads [26] that may be adjusted continuously, or selected from discrete values [4]. In particular, discrete utilizations may be constructed so as to guarantee harmonic period assignments [4]. This approach, however, does not allow task utilizations to be compressed over continuous ranges within the constraints of harmonicity; nor does it address the question of *how* to select candidate harmonic period values within a range that is acceptable for each task.

Harmonic Periods: Prior studies have considered using harmonic periods to improve schedulability. It is well-known (see, e.g., [27]) that rate monotonic (RM) preemptive scheduling on a uniprocessor permits a utilization bound of 1 if tasks are assigned harmonic periods. Han and Tyan presented a schedulability test that takes advantage of this by mapping task periods to smaller artificial harmonic values [28], borrowing from an earlier algorithm presented by Han and Lin [29]. Shih et al. used this same algorithm to schedule radar dwells [30]. Fan and Quan demonstrated a strategy to partition fixed-priority tasks on a multiprocessor according to their harmonic relationships [31]. Min-Allah et al. proposed a utilization bound test for task sets with deadlines larger than periods that constructs artificial harmonic deadlines [32]. In these studies, however, tasks still execute at their originally-assigned periods.

Period *assignment* from sets of acceptable values has also been studied in prior work. Kuo and Mok studied systems where each task is assigned a *discrete* set of periods at which it may execute, and for which execution times scale with periods (utilizations remain constant) [27]. They considered assigning

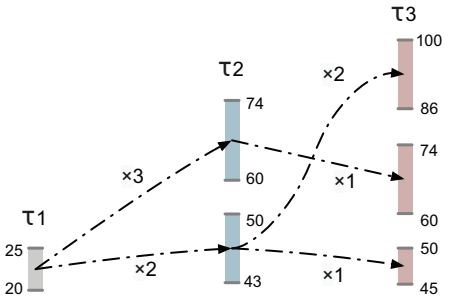


Fig. 1: Forward search for harmonic periods via projected harmonic zones. Tasks τ_1, τ_2, and τ_3 must take periods in the intervals I_1=$[20, 25]$, I_2=$[43, 74]$, and I_3=$[45, 100]$. Projected harmonic zones from I_1 to I_2 are re-projected. Since these projections overlap I_3, harmonic periods can be assigned.

periods to tasks so that the number of subsets of tasks with harmonic periods does not exceed some value h selected to guarantee RM schedulability. For $h = 1$, this guarantees that all periods are harmonic. Assigning periods to tasks from continuous ranges so as to minimize the hyperperiod has also been studied [8], [9]. While this approach will find an assignment (if one exists) of periods to each task such that the *largest* period is an integer multiple of the others, this does not guarantee that *all* periods are harmonic.

However, many applications demand harmonic task rates for functional correctness, not just to increase the schedulable utilization bound. These include robotic control systems [3], [7], real-time hybrid simulation [4], [16], [33], SLAM systems [5], [17], [18] and mobile spectrometry [6]. To satisfy such requirements, we consider the problem of assigning harmonic periods to tasks from *continuous* intervals. We build off work by Nasri et al. [12], who previously studied this problem and first proposed a solution. In their model a continuous interval $I_i = [T_i^{\min}, T_i^{\max}]$ is specified for each task τ_i from which its period must be selected. Their approach orders intervals according to $T_i^{\min}$, then searches for a sequence of "projected harmonic zones" from the first interval to the last.

Definition 1 (Projected Harmonic Zone)**.** *(This is a restatement of [12, Definition 1].) The projected harmonic zone* $\chi^a_{I_1 \to I_2} : [T_\chi^{\min}, T_\chi^{\max}]$ *from interval* I_1 *to* I_2*,* $T_1^{\min} \leq T_2^{\min}$*, with multiplier* $a \in \mathbb{N}^+$ *is a continuous range of numbers in* I_1 *that starts from* $T_\chi^{\min} = \max\{T_2^{\min}, a \cdot T_1^{\min}\}$ *and ends at* $T_\chi^{\max} = \min\{T_2^{\max}, a \cdot T_1^{\max}\}$.

Fig. 1 illustrates the forward search technique proposed in [12]. Also in [12], Nasri et al. show how to identify if all harmonic zones projected onto a given interval fully overlap; in the case where this is true for all intervals, the problem is solved in linear time in the number of tasks. However, they state that in general "the number of [projected harmonic zones] can exponentially grow." In [13], Nasri and Fohler propose a backward search starting from the last interval. If all backward projections are disjoint, the problem can be solved in time $\mathcal{O}(n^2 \log n)$ for n tasks, but they provide "no guarantee for reasonable computational complexity" in general. However, in §IV-B, we present a general pseudo-polynomial algorithm to assign harmonic periods from within the specified intervals.

In [34], Mohaqeqi, Nasri, et al. extend harmonic period assignment to various optimization problems. For example, they demonstrate that assigning harmonic periods from continuous intervals to minimize a weighted sum over the periods while respecting schedulability is $\mathcal{NP}$-hard. We show a similar result in §IV-C for the elastic objective. In [35], Pavić and Džapo also present algorithms and complexity results for the objectives from [34], but with discrete constraints on periods.

III. Problem Statements

In this work, we consider the following three problems:

The Harmonic Period Problem: Given a set Γ of n tasks, where each task τ_i is characterized by a continuous interval $I_i = [T_i^{\min}, T_i^{\max}]$, assign each task a period T_i such that for all $i, 1 \leq i \leq n$, these conditions are satisfied: (i) $T_i \in I_i$, and (ii) for any i, j, either $T_i/T_j \in \mathbb{N}^+$ or $T_j/T_i \in \mathbb{N}^+$.

This is the problem considered in [12] and [13], and does not consider schedulability under the resulting set of assignments. In §IV-A we argue that the problem is unlikely to have a polynomial time solution in the number of tasks, but in §IV-B, we present a pseudo-polynomial algorithm.

The Harmonic Elastic Problem: Given a set Γ of n implicit-deadline elastic tasks on a uniprocessor, where each task τ_i is characterized by five non-negative parameters $(C_i, T_i^{\min}, T_i^{\max}, T_i, E_i)$ as described in §II, find an assignment of periods T_i to each task τ_i satisfying the constrained optimization problem in Eqn. 1, with the additional constraint:

$$\forall_{i,j}, \quad T_i/T_j \in \mathbb{N}^+ \text{ or } T_j/T_i \in \mathbb{N}^+ \tag{2}$$

In §IV-C, we prove that this is $\mathcal{NP}$-hard in general.

The Ordered Harmonic Elastic Problem: This problem is equivalent to the harmonic elastic problem, but with the additional restriction that task periods must be selected to respect some total ordering assigned a priori. In other words, $\forall i, j$ such that $1 \leq i < j \leq n$, it is required that $T_i \leq T_j$. This is a natural restriction in several applications, and in §V we show that it allows for polynomial-time online period adjustment.

IV. Complexity Results

A. Complexity of the Harmonic Period Problem

The *decision version* of the harmonic period problem asks: Given a set Γ of n tasks where each task τ_i is characterized by a continuous interval $I_i = [T_i^{\min}, T_i^{\max}]$, *is it possible to* assign each task a period T_i such that for all $i, 1 \leq i \leq n$, the following conditions are satisfied: (i) $T_i \in I_i$, and (ii) for any i, j, either $T_i/T_j \in \mathbb{N}^+$ or $T_j/T_i \in \mathbb{N}^+$? We point out that this decision problem is no harder than the problem of actually finding the periods; hence, any lower bounds on the computational complexity of this decision problem also holds for the problem of finding the periods.

We now show, by reduction from integer factorization (defined below), that it is unlikely that this decision problem can be solved in time polynomial in n and $\log k$, where k bounds the ratio of the task periods.

Definition 2 (Integer Factorization)**.**

INSTANCE*: Positive integers N and ℓ, $1 < \ell < N$.*
QUESTION*: Does N have an integer factor in $[2, \ell]$?*

Integer factorization is widely believed to not be solvable by polynomial-time algorithms (assuming $\mathcal{P} \neq \mathcal{NP}$) [36]. The following theorem builds upon this belief to show that we are unlikely to be able to solve the decision version of the harmonic period problem in polynomial time:

Theorem 1. *If the decision version of the harmonic period problem can be solved by a polynomial-time algorithm, then integer factorization can as well.*

Proof. We can reduce integer factorization to the decision version of the harmonic period problem as follows. For any instance (N, ℓ) of integer factorization, we construct a set of three tasks $\Gamma = \{\tau_1, \tau_2, \tau_3\}$ with period intervals as follows:

$$I_1 = [1, 1] \qquad I_2 = [2, \ell] \qquad I_3 = [N, N]$$

Note that τ_1's period T_1 must equal 1 and τ_3's period T_3 must equal N. The harmonicity constraint mandates that (i) τ_2's period T_2 be divisible by $T_1 \equiv 1$, and hence an integer within the interval $[2, \ell]$; and (ii) this period T_2 must be a divisor of τ_3's period $T_3 \equiv N$. Taken together, these facts imply that the task system Γ constructed above is a YES instance for the decision version of the harmonic period problem if and only if N has an integer factor in $[2, \ell]$; i.e., if (N, ℓ) is a YES instance of integer factorization. □

The hardness result above means that we are unlikely to be able to obtain a polynomial-time algorithm for solving the Harmonic Period Problem; this motivates our efforts to obtain a *pseudo*-polynomial algorithm that solves it.

B. An Algorithm for the Harmonic Period Problem

We now derive a pseudo-polynomial algorithm that solves the harmonic period problem. Intuitively, it is similar to Nasri et al.'s forward projection approach [12] (illustrated in §II, Fig. 1), but we observe that if adjacent intervals I_i, I_{i+1} overlap (i.e., if $T_{i+1}^{\min} < T_i^{\max}$), then the search from projected harmonic zones into this overlapping region in I_i only requires a re-projection into I_{i+1} using the multiplier a=1 (see Fig. 2).

The procedure is outlined in Alg. 1. It first sorts tasks in ascending order of $T_i^{\min}$; in [12, Corollary 1], Nasri et al. showed that projecting harmonic zones in this order will find a set of harmonic periods, if one exists. It then performs a breadth-first search for projected harmonic zones over just those intervals that do not enclose any subsequent intervals (see Fig. 3). The following lemma proves that enclosing intervals can be removed from the search space.

Lemma 1. *If adjacent tasks τ_i and τ_{i+1} have intervals I_i and I_{i+1} such that $T_i^{\min} \leq T_{i+1}^{\min}$ and $T_i^{\max} \geq T_{i+1}^{\max}$, we say that the interval I_i **encloses** I_{i+1}. Then if any solution to the harmonic period problem exists, there must exist a solution that satisfies $T_i = T_{i+1}$.*

Algorithm 1: FIND-HARMONIC-PERIODS(Γ)

Input: A set Γ of n tasks with period intervals $I_i = [T_i^{\min}, T_i^{\max}]$
Output: A set of harmonic period assignments $\{T_i\}$

▷ Initialize task set
Sort Γ in ascending order of $T_i^{\min}$
$S \leftarrow \{\}$ ▷ Sequence of period intervals
forall $1 \leq i < n$ **do**
  if $T_i^{\max} < T_{i+1}^{\max}$ **then** Insert I_i into $\mathbf{S}$
Insert I_n into $\mathbf{S}$

▷ Find projected harmonic zones
$j \leftarrow$ first element in $\mathbf{S}$
SOURCES $\leftarrow \{(I_j, \{0\})\}$
(INTERVAL, $\{a_i\}$) $\leftarrow$ PROJECT($\mathbf{S}$, SOURCES, 2)
if FAILURE **then return** FAILURE

▷ Assign periods from harmonic zones
Assign to T_n any value in INTERVAL
forall $i : n{-}1 \rightarrow 1$ **do**
  if $I_i \in S$ **then** $T_i = T_{i+1}/a_j$
  else $T_i = T_{i+1}$
return $\{T_i\}$

Algorithm 2: PROJECT($\mathbf{S}$, SOURCES, i)

Inputs: A set $\mathbf{S}$ of continuous period intervals $I_i = [T_i^{\min}, T_i^{\max}]$
A set SOURCES of projected harmonic zones $[T^{\min}, T^{\max}]$ and corresponding sequence of multipliers $\mathbf{M}$
An index i of the period interval in $\mathbf{S}$ into which to project
Outputs: A projected harmonic zone into the last interval in $\mathbf{S}$
The sequence of integer multipliers that produces the corresponding set of projected harmonic zones

▷ Reached the end
if $i \geq |\mathbf{S}|$ **then**
  if SOURCES *is not empty* **then** return any element in SOURCES
  else return FAILURE

$j \leftarrow i^{\text{th}}$ interval in $\mathbf{S}$
TARGETS $\leftarrow \{\}$
$o \leftarrow T_j^{\min}$; $k \leftarrow -1$; ▷ Track lower-bound and index of largest non-overlapping split region
forall $([T_s^{\min}, T_s^{\max}], M) \in$ SOURCES **do**
  ▷ Source zone overlaps target
  if $T_j^{\min} < T_s^{\max}$ **then**
    Insert 1 into $\mathbf{M}$
    Insert $([T_j^{\min}, T_s^{\max}], \mathbf{M})$ into TARGETS
    if $T_s^{\min} < o$ **then** $o \leftarrow T_s^{\min}$; $k \leftarrow s$
  ▷ No overlap
  else
    $a^{\min} \leftarrow \lceil T_j^{\min}/T_s^{\max} \rceil$; $a^{\max} \leftarrow \lfloor T_j^{\max}/T_s^{\min} \rfloor$
    forall $a^{\min} \leq a \leq a^{\max}$ **do**
      $\mathbf{M}' \leftarrow \mathbf{M}$
      Insert a into $\mathbf{M}'$
      $T^{\min} \leftarrow \max\{T_j^{\min}, a \cdot T_s^{\min}\}$
      $T^{\max} \leftarrow \min\{T_j^{\max}, a \cdot T_s^{\max}\}$
      Insert $([T^{\min}, T^{\max}], \mathbf{M}')$ into TARGETS
▷ There was overlap
if $k > -1$ **then**
  $T^{\min} \leftarrow o$; $T^{\max} \leftarrow T_j^{\min}$; $a^{\max} \leftarrow \lfloor T_j^{\max}/T^{\min} \rfloor$;
  forall $2 \leq a \leq a^{\max}$ **do**
    $\mathbf{M}' \leftarrow k^{\text{th}}$ multiplier sequence in $\mathbf{S}$
    Insert a into $\mathbf{M}'$ $T^{\min} \leftarrow \max\{T_j^{\min}, a \cdot T_s^{\min}\}$
    $T^{\max} \leftarrow \min\{T_j^{\max}, a \cdot T_s^{\max}\}$
    Insert $([T^{\min}, T^{\max}], \mathbf{M}')$ into TARGETS
return PROJECT($\mathbf{S}$, SOURCES, $i + 1$)

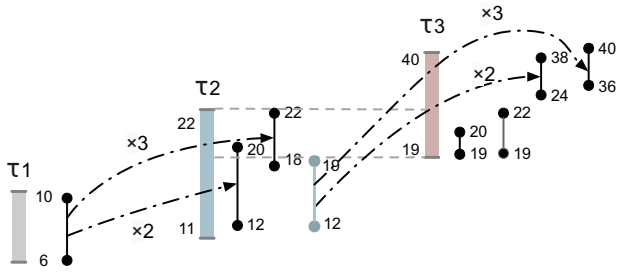


Fig. 2: Tasks τ_1–τ_3 have intervals I_1=[6, 10], I_2=[11, 22], I_3=[19, 40]. Projected harmonic zones $\chi^2_{I_1\to I_2}$: [12, 20] and $\chi^3_{I_1\to I_2}$: [18, 22] both overlap I_3. The overlapping portions are re-projected using only the multiplier $a = 1$, forming $\chi^1_{I_2\to I_3}$: [19, 20] and $\chi^1_{I_2\to I_3}$: [19, 22]. The non-overlapping portions [12, 19] and [18, 19] are merged and re-projected into I_3 starting from multiplier $a = 2$, forming $\chi^2_{I_2\to I_3}$: [24, 38] and $\chi^3_{I_2\to I_3}$: [36, 40].

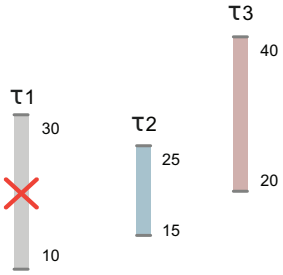


Fig. 3: Task τ_1 has period interval I_1=[10, 30] and τ_2 has I_2=[15, 25]. As I_1 encloses I_2, we remove τ_1 from the search space. Its period T_1 can take the value assigned to T_2. Task τ_3 has I_3=[20, 40], which overlaps but is not enclosed by I_2.

Proof. Assume that for a task system Γ there exists a harmonic assignment of periods T_i to each task τ_i such that $T_i \in I_i$, i.e., for all i, j, either $T_i/T_j \in \mathbb{N}^+$ or $T_j/T_i \in \mathbb{N}^+$. Then it follows from [12, Corollary 1] that for any two tasks τ_i, τ_j for which $T_i^{\min} \le T_j^{\min}$ there is some such assignment for which $T_j/T_i \in \mathbb{N}^+$, implying that $T_i \le T_j$. Now assume that $T_i^{\max} \ge T_j^{\max}$. If we reassign T_i to the value T_j, it still follows that for all k, either $T_i/T_k \in \mathbb{N}^+$ or $T_k/T_i \in \mathbb{N}^+$. Additionally, since $T_i^{\min} \le T_j^{\min} \le T_j \le T_j^{\max} \le T_i^{\max}$, it still holds that $T_i \in I_i$. So the harmonic period problem is still satisfied. □

The algorithm proceeds via recursive calls to PROJECT, outlined in Alg. 2. This takes a list (SOURCES) of projected harmonic zones in the current interval to be re-projected to the next interval, each paired with the sequence of multipliers that produced it. It starts with the complete first interval and terminates when the final interval is reached. To enable breadth-first search, it keeps a list (TARGETS) of projected harmonic zones into the next interval. For all projected harmonic zones in SOURCES, the algorithm first checks if the zone overlaps the next interval. If it does not, all re-projected harmonic zones (and their corresponding multiples) from the source zone into the next interval are added to TARGETS. However, if it does, the zone is split (if necessary) into overlapping and non-overlapping regions (see Fig. 2). The overlapping region (with multiplier 1) is added to TARGETS. The following lemma argues that this is sufficient:

Lemma 2. *If adjacent tasks τ_i and τ_{i+1} have intervals such that $T_i^{\min} \le T_{i+1}^{\min}$, $T_i^{\max} > T_{i+1}^{\min}$, and $T_i^{\max} < T_{i+1}^{\max}$, and if a solution to the harmonic period problem exists where T_i is assigned some value t such that $T_{i+1}^{\min} \le t$, then a solution must exist where both $T_i = t$ and $T_{i+1} = t$.*

Proof. Assume that for a task system Γ there exists a harmonic assignment of periods T_i to each task τ_i such that $T_i \in I_i$, i.e., for all i, j, either $T_i/T_j \in \mathbb{N}^+$ or $T_j/T_i \in \mathbb{N}^+$. Then it follows from [12, Corollary 1] that for any two tasks τ_i, τ_j for which $T_i^{\min} \le T_j^{\min}$ there is some such assignment for which $T_j/T_i \in \mathbb{N}^+$, implying that $T_i \le T_j$. Now assume that $T_i^{\max} > T_j^{\min}$ and $T_i^{\max} < T_j^{\max}$. If we reassign T_j to the value T_i, it still follows that for all k, either $T_j/T_k \in \mathbb{N}^+$ or $T_k/T_j \in \mathbb{N}^+$. Additionally, since $T_j^{\min} \le T_j \le T_i^{\max} \le T_j^{\max}$, it still holds that $T_{i+1} \in I_{i+1}$. So the harmonic period problem is still satisfied. □

This says that the portion of a projected harmonic zone χ in I_i containing periods that would be contained within the next interval I_{i+1} does not have to be projected into I_{i+1} using multipliers greater than 1 if $T_{i+1}^{\max} > T_\chi^{\max}$. Because we have eliminated from the search those intervals where $T_{i+1}^{\max} < T_i^{\max}$, this condition holds.

After all SOURCES have been re-projected into the next interval as described above, it remains to deal with the largest among the non-overlapping regions that have been split off from overlapping regions, if there are any (all such split regions are subsets of the largest). Projections from this region are added to TARGETS; these start from multiplier 2, since the overlapping region has already been projected with a multiplier of 1. The procedure is then called recursively for the next interval, with TARGETS passed as the new list of SOURCES.

If PROJECT succeeds in finding harmonic zones projected into the last interval, it returns one with its sequence of multiples to the calling procedure. Any value in the projected harmonic zone may be assigned as the period T_n of the last task τ_n, then other task periods are obtained by dividing by their corresponding multipliers. Any task τ_i with an interval completely enclosing the next (and was therefore not included in the search) is assigned a period T_i=T_{i+1} per Lemma 1.

Execution Time: For n tasks and $k = T_n^{\max}/T_1^{\min}$, this algorithm executes in time $\boldsymbol{\mathcal{O}(n \log n + nk^2)}$, according to the following steps. ① Sort intervals and eliminate from the search those which fully enclose subsequent intervals: $\mathcal{O}(n \log n)$. ② Recursively find projected harmonic zones: $\mathcal{O}(nk^2)$. ③ Assign periods from harmonic zones: $\mathcal{O}(n)$.

Steps ① and ③ are obvious, but the recursive procedure of step ② requires further analysis. We begin by showing that if intervals do *not* overlap, then the number of projected harmonic zones into the last interval does not exceed k^2.

Theorem 2. *Assume that a set of n intervals $\{I_i\}$ are ordered in such a way that $\forall i, j$, if $1 \le i < j \le n$, then $T_i^{\min} \le T_j^{\min}$. Assume further that intervals do not overlap, i.e., $T_i^{\max} \le T_j^{\min}$. Then there can be at most k^2 projected harmonic zones into the last interval I_n.*

Proof. For any i, $2{<}i{\leq}n$, the number of projected harmonic zones into I_i from a harmonic zone in I_{i-1} cannot exceed $\lfloor T_i^{\max}/T_{i-1}^{\min} \rfloor$. This follows from [12, Lemma 1].

Then the number of projected zones into I_n cannot exceed

$$\left\lfloor \frac{T_2^{\max}}{T_1^{\min}} \right\rfloor \left\lfloor \frac{T_3^{\max}}{T_2^{\min}} \right\rfloor \cdots \left\lfloor \frac{T_n^{\max}}{T_{n-1}^{\min}} \right\rfloor \leq \frac{T_2^{\max}}{T_1^{\min}} \cdot \frac{T_3^{\max}}{T_2^{\min}} \cdot \ldots \cdot \frac{T_n^{\max}}{T_{n-1}^{\min}}$$

Then since for all $1 \leq i < n$, $T_i^{\max} \leq T_{i+1}^{\min}$, we have:

$$\frac{T_2^{\max}}{T_1^{\min}} \cdot \frac{T_3^{\max}}{T_2^{\min}} \cdot \ldots \cdot \frac{T_n^{\max}}{T_{n-1}^{\min}} \leq \frac{T_2^{\max}}{T_1^{\min}} \cdot \frac{T_3^{\max}}{T_1^{\max}} \cdot \frac{T_4^{\max}}{T_2^{\max}} \cdot \ldots \cdot \frac{T_n^{\max}}{T_{n-2}^{\max}}$$

$$= \frac{T_{n-1}^{\max} \cdot T_n^{\max}}{T_1^{\min} \cdot T_1^{\max}} \leq \left(\frac{T_n^{\max}}{T_1^{\min}} \right)^2 = k^2 \qquad \square$$

We next show that if intervals *do* overlap, the number of projected zones searched by the algorithm does not increase.

Theorem 3. *Assume that a set of n intervals $\{I_i\}$ are ordered in such a way that $\forall i, j$, $1{\leq}i{<}j{\leq}n$, $T_i^{\min}{\leq}T_j^{\min}$ and $T_i^{\max} < T_j^{\max}$, but that some intervals overlap, i.e., for some i, $T_i^{\max} > T_{i+1}^{\min}$. Nonetheless, there are still at most k^2 harmonic zones projected into the last interval I_n.*

Proof. Consider adjacent intervals I_i, I_{i+1}. We define w_i, the number of projected harmonic zones into I_i. Then $w_i = x_i + y_i$, where x_i is the number of projected harmonic zones at least partially overlapping I_{i+1} and y_i is the number not overlapping. If $x_i = 0$ then the number of projections from I_i to I_{i+1} is the same as if the intervals themselves do not overlap. Otherwise, if $x_i > 0$, then the number of projected harmonic zones into I_{i+1} is no more than:

$$x_i + (y_i + 1) \cdot \left(\left\lfloor \frac{T_{i+1}^{\max}}{T_i^{\min}} \right\rfloor - 1 \right) \tag{3}$$

The x_i term is due to the overlapping regions which are projected only once with a multiplier of 1. The largest non-overlapping region split off from the projected harmonic zones that partially overlap I_{i+1}, as well as the y_i projected harmonic zones that do not overlap, contribute to the term $(y_i + 1)$. These must be re-projected into I_{i+1} (see Fig. 2). However, because those projected harmonic zones χ that do not overlap I_{i+1} have $T_\chi^{\max} < I_{i+1}^{\min}$, the projected harmonic zones from these regions into I_{i+1} have multiples no less than 2. Similarly (per line 31 of Alg. 2) projections from the largest split region may begin from 2. Thus, it follows that the number of projected harmonic zones from these regions cannot exceed $\left(\lfloor T_{i+1}^{\max}/T_i^{\min} \rfloor - 1\right)$. Then we define $z_i = \lfloor T_{i+1}^{\max}/T_i^{\min} \rfloor$, so the number of projected harmonic zones does not exceed:

$$x_i + (y_i + 1) \cdot (z_i - 1) = x_i + y_i z_i + z_i - y_i - 1$$

$$\leq x_i + z_i - 1 - x_i z_i + x_i z_i + y_i z_i = -(x_i - 1)(z_i - 1) + x_i z_i + y_i z_i$$

Then since $T_i^{\min} < T_{i+1}^{\max}$, it follows that $z_i \geq 1$. And since $x_i \geq 1$, it follows that $(x_i - 1)(z_i - 1) \geq 0$, so:

$$-(x_i - 1)(z_i - 1) + x_i z_i + y_i z_i \leq x_i z_i + y_i z_i = (x_i + y_i) z_i$$

This is precisely $w_i \cdot \lfloor T_{i+1}^{\max}/T_i^{\min} \rfloor$, which upper bounds the number of projected harmonic zones into I_{i+1} if it does not overlap I_i. Thus, overlapping regions do not increase the number of projected harmonic zones. $\square$

Then, since there can be no more than k^2 projected harmonic zones into any single interval, it follows that there can be no more than $(n-1)k^2$ projected harmonic zones across n intervals. So the described algorithm runs in time $\mathcal{O}(nk^2)$.

C. Complexity of the Harmonic Elastic Problem

Even if a set of assignments is found that satisfies the harmonic period problem, finding the assignment that satisfies the harmonic *elastic* problem is still at least weakly $\mathcal{NP}$-hard, even for a fixed value of k bounding the ratio of the task periods. We prove this by reduction from integer partitioning.

Definition 3 (Integer Partitioning)**.** *Let $A{=}\{s_1, \ldots, s_n\}$ be a set of positive integers. The problem is to determine whether A can be partitioned into two sets A_1 and A_2 such that the sum of integers in A_1 equals that of A_2. That is, if:*

$$S_1 = \sum_{s_i \in A_1} s_i \tag{4}$$

and similarly for S_2, then the problem is to decide whether there exist A_1 and A_2 such that $A_1 \cup A_2 = A$, $A_1 \cap A_2 = \emptyset$, and $S_1 = S_2$. We define $S{=}\sum_i s_i$, i.e., $S{=}S_1{+}S_2$.

Integer partitioning is weakly $\mathcal{NP}$-hard, and we demonstrate that any instance can be solved by a polynomial-time reduction to the harmonic elastic problem. Consider an arbitrary instance of the problem over n positive integers. We construct a corresponding instance of the harmonic elastic problem consisting of a set Γ of $n+1$ elastic tasks scheduled with a utilization bound $U_D = 1$. For each task τ_i, $i \leq n$, we assign as its worst-case execution time:

$$C_i = \frac{4 s_i}{3S} \tag{5}$$

We allow task periods to be assigned in the interval $[1, 2]$. Elasticity constants are assigned to each task τ_i, $i \leq n$ as $E_i = C_i/4$. The remaining task, τ_{n+1} is assigned $C_{n+1} = 0$ and $I_{n+1} = [1, 1]$, making it inelastic. Thus, the parameter k introduced in §IV-B that bounds the ratio of task periods is fixed to 2. Now, we let $\{T_1, T_2, \ldots, T_{n+1}\}$ denote a possible period assignment, and define a value O corresponding to the value taken by objective (1a) for this assignment. We also define a value O^* denoting the minimum (i.e., optimal) value of O for which constraints (1b)–(2) are met.

Lemma 3. *For the above problem instance, $O^* \geq 2/3$.*

Proof. Let T_i denote the period of task τ_i assigned by a solution to the harmonic elastic problem. According to the specified task parameters, the period of $T_{n+1}{=}1$, so all other tasks must take periods of either $T_i{=}1$ or $T_i{=}2$ to remain harmonic. We then define the sets $\mathbb{T}_1 = \{\tau_i | T_i = 1\}$ and $\mathbb{T}_2 = \{\tau_i | T_i = 2\}$. Similarly, $\mathbb{C}_1 = \sum_{\tau_i \in \mathbb{T}_1} C_i$, $\mathbb{C}_2 = \sum_{\tau_i \in \mathbb{T}_2} C_i$, and $\mathbb{C} = \sum_i C_i$, from which it follows that $\mathbb{C} = \mathbb{C}_1 + \mathbb{C}_2$ and

$$\mathbb{C} = \sum_{i=1}^{n} \frac{4 s_i}{3S} = \frac{4}{3S} \sum_{i=1}^{n} s_i = \frac{4}{3S} \cdot S = \frac{4}{3} \tag{6}$$

Then total utilization is:

$$U = \sum_{i=1}^{n} \frac{C_i}{T_i} = \mathbb{C}_1 + \frac{\mathbb{C}_2}{2} \tag{7}$$

Since for all $\tau_i \in \mathbb{T}_1$, $U_i = U_i^{\max}$, we can express O as:

$$O = \sum_{\tau_i \in \mathbb{T}_2} \frac{4}{C_i}(C_i - \frac{C_i}{2})^2 = \sum_{\tau_i \in \mathbb{T}_2} C_i = \mathbb{C}_2 \tag{8}$$

Then from (7) it follows that:

$$2\mathbb{C} - 2U = 2\mathbb{C}_1 + 2\mathbb{C}_2 - 2\mathbb{C}_1 - \mathbb{C}_2 = \mathbb{C}_2 = O \tag{9}$$

Equivalently, $O = 2\mathbb{C} - 2U$. Then from (6) it follows that:

$$O = 2(4/3) - 2U = 8/3 - 2U \tag{10}$$

Since U is constrained as in (1b) to $U \leq U_D$, we have $O \geq 8/3 - 2 = 2/3$, and so it follows that $O^* \geq 2/3$. □

Lemma 4. *A given instance of the partitioning problem is positive if and only if $O^* = 2/3$ in the constructed instance of the harmonic elastic problem.*

Proof. We first show that if the given partitioning problem is a positive instance, then the optimal period assignment satisfies O^*=$2/3$. Let A_1 and A_2 denote the partitions of A for which S_1=S_2. We can assign task periods such that T_i=1 if $s_i \in A_1$ and $T_i = 2$ if $s_i \in A_2$. Then from (5), $\mathbb{C}_1$=$\mathbb{C}_2$=$\mathbb{C}/2$. Since $\mathbb{C} = 4/3$ from (6), it follows that $\mathbb{C}_2$=$2/3$ and so from (8), O=$2/3$. From the previous lemma, O^*≥$2/3$, and so O^*=$2/3$.

Next, we show that if $O^* = 2/3$, then the given partitioning problem is a positive instance. If $O^* = 2/3$, then (8) implies that $\mathbb{C}_2 = 2/3$. Also, since from (6), $\mathbb{C} = 4/3$, this implies that $\mathbb{C}_1 = 2/3$. Since $\mathbb{C}_1 = \mathbb{C}_2$, this implies that:

$$\sum_{\tau_i \in \mathbb{T}_1} \frac{4s_i}{3S} = \sum_{\tau_i \in \mathbb{T}_2} \frac{4s_i}{3S} \tag{11}$$

or equivalently, $\sum_{\tau_i \in \mathbb{T}_1} s_i = \sum_{\tau_i \in \mathbb{T}_2} s_2$, which implies that $S_1 = S_2$, so the problem is positive. □

V. The Ordered Harmonic Elastic Problem

We restrict our attention now to the ordered harmonic elastic problem, as defined in §III. Suppose we have a set Γ of n tasks, indexed so that for any τ_i, τ_j, if $i < j$ then $T_i \leq T_j$. It follows, then, that an assignment of harmonic periods can be expressed according to T_1 (the shortest period assignment) and a set of multiples $\{a_i, 1{\leq}i{\leq}n\}$ where $T_i = a_i \cdot T_1$.

Definition 4 (Projected Harmonic Interval (PHI))**.** *For a set Γ of n elastic tasks, a projected harmonic interval is an ordered collection of values $P_j = (T_j^{\min}, T_j^{\max}, a_{1,j}, \ldots, a_{n,j})$ with $a_{1,j}{\equiv}1$. It represents the largest continuous interval $[T_j^{\min}, T_j^{\max}] \subset [T_1^{\min}, T_1^{\max}]$ from which a period T_1 can be selected so that a period $T_i = a_{i,j} \cdot T_1$ assigned to task τ_i is within the interval I_i that characterizes the task. Furthermore, for all b, c, $1 \leq b \leq c \leq n$ it holds that $a_{c,j}/a_{b,j} \in \mathbb{N}^+$.*

Enumeration-Based Solution Approach: The set of all PHIs $\mathbf{P} = \{P_j\}$ for a task set represents the complete space of joint harmonic period assignments that can be selected from the continuous intervals characterizing each task. Thus, a naïve approach to optimally solving the ordered harmonic elastic problem for a given utilization bound is to enumerate the complete set of PHIs, then determine which PHI minimizes the elastic objective (Eqn. 1a) for that utilization. To do so, we define $U_{P_j}^{\min}$, the minimum utilization bound that can accommodate PHI P_j. This can be calculated as:

$$U_{P_j}^{\min} = \sum_i \frac{C_i}{a_{i,j} \cdot T_j^{\max}} = \frac{1}{T_j^{\max}} \sum_i \frac{C_i}{a_{i,j}} \tag{12}$$

For simplicity of notation, we define the terms $y_{i,j} = C_i/a_{i,j}$ and $Y_j = \sum_i y_{i,j}$. Then $U_{P_j}^{\min} = Y_j/T_j^{\max}$. Similarly, the maximum utilization that can be achieved by PHI P_j is $U_{P_j}^{\max} = Y_j/T_j^{\min}$. It follows from Eqn. 1a that for $U_D \geq U_{P_j}^{\min}$, the elastic objective $O_{P_j}(U_D)$ for PHI P_j is:

$$O_{P_j}(U_D) = \begin{cases} \sum_{i=1}^{n} \frac{1}{E_i} \left(U_i^{\max} - \frac{y_{i,j} U_D}{Y_j} \right)^2 & \text{if } U_D \leq U_{P_j}^{\max} \\ \sum_{i=1}^{n} \frac{1}{E_i} \left(U_i^{\max} - \frac{y_{i,j} U_{P_j}^{\max}}{Y_j} \right)^2 & \text{if } U_D > U_{P_j}^{\max} \end{cases} \tag{13}$$

Our naïve approach computes $O_{P_j}(U_D)$ for every PHI P_j. The PHI that minimizes the objective is selected, and task τ_1 is assigned the period $T_{1,j}^{U_D}$:

$$T_{1,j}^{U_D} = \min\left\{T_j^{\min}, Y_j/U_D\right\} \tag{14}$$

Other task periods are then assigned as $T_i = a_{i,j} \cdot T_{1,j}^{U_D}$.

Bounding Enumeration: This enumeration-based approach may be inefficient as the number of PHIs grows rapidly. However, this growth is bounded, a result which we will use to provide a polynomial-time algorithm for online adjustment.

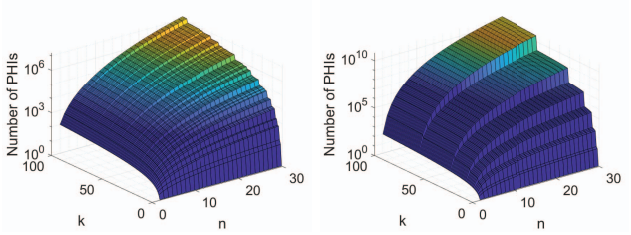
Theorem 4. *For n tasks and k=$T_n^{\max}/T_1^{\min}$ as in §IV, the number of PHIs $|\mathbf{P}|$ is bounded by $k(n-1)^{\lfloor \log k \rfloor}$.*

Proof. We know that there is a sequence $\{a_2, \ldots, a_n\}$, $a_i \in \mathbb{N}^+$ such that $T_1^{\min} \cdot a_2 \cdot \ldots \cdot a_n \leq T_n^{\max}$, which implies that $a_2 \cdot \ldots \cdot a_n \leq T_n^{\max}/T_1^{\min} = k$. We define $h_{n,k}$ as the number of unique such sequences satisfying $a_2 \cdot \ldots \cdot a_n \leq k$. Similarly, we define $h_{n,k}^*$ as the number of unique such sequences satisfying $a_2 \cdot \ldots \cdot a_n = k$. It follows that

$$h_{n,k} = \sum_{\ell=1}^{k} h_{n,\ell}^*$$

We say an integer ℓ has a unique factorization into p_ℓ primes. Then a sequence satisfying $a_2 \cdot \ldots \cdot a_n = \ell$ must be formed in such a way that each value a_i, $2{<}i{\leq}n$ is the product of 1 and zero or more of the prime factors of ℓ, selected without replacement. The number of unique assignments of p_ℓ primes to $n{-}1$ factors is exactly $(n-1)^{p_\ell}$ for non-repeated prime factors, fewer otherwise. And the value p_ℓ is upper bounded by $\lfloor \log \ell \rfloor$. So $h_{n,\ell}^* \leq (n-1)^{\lfloor \log \ell \rfloor}$, and so

$$h_{n,k} = \sum_{\ell=1}^{k} h_{n,\ell}^* \leq \sum_{\ell=1}^{k} (n-1)^{\lfloor \log \ell \rfloor} \leq k(n-1)^{\lfloor \log k \rfloor} \quad \square$$



(a) Calculated number of PHIs. (b) Upper bound on PHIs.

Fig. 4: Enumeration of projected harmonic intervals.

We note that this derived upper bound is not tight, because if $\ell \neq 2^a$ for $a \in \mathbb{N}$, then $p_\ell < \lfloor \log \ell \rfloor$. Furthermore, many integers have repeated prime factors. To better analyze this bound, we compute $h_{n,k}$ for values of n from 2–30 and values of k from 1–100. Results are plotted in Fig. 4a, and compared in Fig. 4b with the upper bound from Thm. 4. For n=30 and k=100, the upper bound is $\sim 5.9\text{e}10$, whereas the real count is under 1.5e7, almost $4000\times$ fewer.

Polynomial Online Adjustment: We now describe a more efficient algorithm to reassign task periods in response to runtime changes to available utilization. It assigns the same periods as the enumeration-based approach described above, finding an optimal solution to the ordered harmonic elastic problem if one exists. Through offline enumeration of $\mathbf{P}$, it constructs a lookup table over the space of utilizations that can accommodate the task system. It associates ranges of utilization with the PHI achieving the lowest elastic objective. Binary search enables polynomial-time online period selection.

The procedure, outlined in Alg. 3, takes a set Γ of n elastic tasks, as well as the set $\mathbf{P}$ of all PHIs. It then creates a list $\mathbf{R}$ sorted over contiguous and continuous regions of utilization, where each region $R_k = (R_k^{\min}, R_k^{\max}, P_j)$ captures the PHI P_j that achieves the lowest elastic objective for utilizations in $[R_k^{\min}, R_k^{\max}]$. To do so, it creates a region for the first PHI, then iterates over all remaining PHIs P_j. Each one is compared to all existing regions in order; those for which $R_k^{\max} \leq U_{P_j}^{\min}$ are skipped. When comparing a PHI P_j to a region R_k, the procedure finds points within the region where it may need to be split, i.e., those distinct sub-regions where the elastic objective value achieved by P_j may be less than that of the region's PHI P_k. These split points are of two types. In the first considered region for which $R_k^{\max} > U_{P_j}^{\min}$, if $R_k^{\min} < U_{P_j}^{\min}$, then it splits at $U_{P_j}^{\min}$. It may also split at those points U where the elastic objectives intersect, i.e., where $O_{P_j}(U) = O_{P_k}(U)$.

Because we are concerned only with *minimizing* the objective, we eliminate constant terms:

$$O_{P_j}^*(U_D) = \begin{cases} A_{P_j} U_D^2 - B_{P_j} U_D & \text{if } U_D \leq U_{P_j}^{\max} \\ A_{P_j} (U_{P_j}^{\max})^2 - B_{P_j} U_{P_j}^{\max} & \text{if } U_D > U_{P_j}^{\max} \end{cases} \tag{15}$$

$$A_{P_j} = \frac{1}{Y_j^2} \sum_i \frac{y_{i,j}^2}{E_i} \tag{16}$$

$$B_{P_j} = \frac{2}{Y_j} \sum_i \frac{y_{i,j} U_i^{\max}}{E_i} \tag{17}$$

Algorithm 3: GENERATE-LOOKUP-TABLE($\Gamma, \mathbf{P}$)

Inputs: A set Γ of n elastic tasks, the set $\mathbf{P}$ of all PHIs over Γ
Output: A sorted list $\mathbf{R}$ over continuous, contiguous regions $R_k = (R_k^{\min}, R_k^{\max}, P_k)$ indicating that the projected harmonic interval P_k achieves the minimum elastic objective in the utilization range $[R_k^{\min}, R_k^{\max}]$
▷ Compute PHI parameters
forall $P_j \in \mathbf{P}$ **do**
  Compute $U_{P_j}^{\min}$, $U_{P_j}^{\max}$, Y_{P_j}, A_{P_j}, B_{P_j}, $O_{P_j}^*(U_{P_j}^{\max})$ according to Eqns. 12, 15, 16, 17
▷ Construct regions
$\mathbf{R} \leftarrow \{(U_1^{\min}, \infty, P_1)\}$
forall $P_j \in \mathbf{P}, j > 1$ **do**
  forall $R_k \in \mathbf{R}$ **do**
    $(R_k^{\min}, R_k^{\max}, P_k) \leftarrow R_k$
    if k *is* 1 *and* $U_j^{\min} < R_k^{\min}$ **then**
      Insert $(U_j^{\min}, R_k^{\min}, P_j)$ into $\mathbf{R}$ before R_k
    if $U_j^{\max} \leq R_k^{\min}$ **then continue**
    if $R_k^{\min} < U_j^{\min}$ **then**
      Insert $(R_k^{\min}, U_j^{\min}, P_k)$ into $\mathbf{R}$ before R_k
      $R_k^{\min} \leftarrow U_j^{\min}$
    ▷ Parabola/parabola intersection
    if $A_j \neq A_k$ *and* $B_j \neq B_k$ **then**
      $q \leftarrow (B_j - B_k)/(A_j - A_k)$
    else $q \leftarrow 0$
    if $q \notin R_k$ *or* $q > U_j^{\max}$ *or* $q > U_k^{\max}$ **then** $q \leftarrow 0$
    ▷ Line/parabola intersection
    if $O_j^{\min} < O_k^{\min}$ **then**
      $\ell \leftarrow \frac{B_j - \sqrt{B_j^2 + 4A_j O_k^{\min}}}{2A_j}$
      if $\ell \notin R_k$ *or* $\ell < U_k^{\max}$ *or* $\ell > U_j^{\max}$ **then** $\ell \leftarrow 0$
    else
      $\ell \leftarrow \frac{B_k - \sqrt{B_k^2 + 4A_k O_j^{\min}}}{2A_k}$
      if $\ell \notin R_k$ *or* $\ell < U_j^{\max}$ *or* $\ell > U_k^{\max}$ **then** $\ell \leftarrow 0$
    ▷ Split region
    REGIONS $\leftarrow R_k$
    if $q > 0$ **then** Split R_k at q
    if $\ell > 0$ **then** Split R_k at ℓ
    Replace R_k with REGIONS
    ▷ Associate region with better PHI
    forall $R_\ell \in$ REGIONS **do**
      $U_\ell \leftarrow (R_\ell^{\min} + R_\ell^{\max})/2$
      $O_k \leftarrow \max\left\{O_\ell^{\min}, A_k \cdot U_\ell^2 - B_k \cdot U_\ell\right\}$
      $O_j \leftarrow \max\left\{O_j^{\min}, A_j \cdot U_\ell^2 - B_j \cdot U_\ell\right\}$
      if $O_j < O_i$ **then** R_ℓ points to P_j
      ▷ Merge redundant regions
      if P_ℓ *matches* $P_{\ell-1}$ **then**
        $R_\ell^{\min} \leftarrow R_{\ell-1}^{\min}$
        Delete $R_{\ell-1}$
return $\mathbf{R}$

Then there are two points where the objectives may intersect. The first is where the quadratic terms are equal:

$$A_{P_j} U^2 - B_{P_j} U = A_{P_k} U^2 - B_{P_k} U$$

If $A_{P_j} \neq A_{P_k}$ and $B_{P_j} \neq B_{P_k}$, then this is:[2]

$$U = \frac{B_{P_j} - B_{P_k}}{A_{P_j} - A_{P_k}} \tag{18}$$

[2]They also trivially intersect at $U = 0$, but this point is not considered.

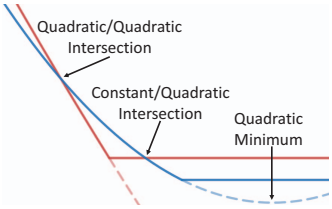


Fig. 5: The set of possible elastic objective intersections.

This value is calculated, then checked to ensure it is within the considered region and does not exceed $U_{P_j}^{\max}$ and $U_{P_k}^{\max}$ (in which case it lies outside the quadratic objective component).

The other intersection point is where the larger of the two PHI objectives' constant components intersects the quadratic component of the other. This can only happen at a single point not exceeding the utilization for which the quadratic component is minimized, illustrated in Fig. 5. Without loss of generality, assume $O_{P_j}^*(U_{P_j}^{\max}) < O_{P_k}^*(U_{P_k}^{\max})$. Then the intersection occurs where:

$$A_{P_j}U^2 - B_{P_j}U = O_{P_k}^*(U_{P_k}^{\max})$$

Solving for the smaller value of U:

$$U = \frac{B_{P_j} - \sqrt{B_{P_j}^2 + 4A_{P_j}O_{P_k}^*(U_{P_k}^{\max})}}{2A_{P_j}} \tag{19}$$

Again, this value is checked to ensure it is both within the considered region, and within the range of utilizations for which $O_{P_j}^*$ is quadratic and $O_{P_k}^*$ is constant — or vice versa, if $O_{P_k}^*(U_{P_k}^{\max}) < O_{P_j}^*(U_{P_j}^{\max})$. Once intersections have been identified, the region is split at those points, and each sub-region is associated with the PHI with the lower objective value according to Eqn. 15. Adjacent sub-regions with the same PHI are merged before replacing the region R_k.

The produced set of regions $\mathbf{R}$ is a sorted lookup table over the entire utilization range that can accommodate the task set. Each region R_k is associated with the PHI P_j that minimizes the elastic objective for the corresponding utilization interval. Thus, for a given utilization bound U_D, binary search finds the optimal PHI in polynomial time, as we now prove.

Theorem 5. *For a set of regions $\mathbf{R}$ constructed over $|\mathbf{P}| = h$ projected harmonic intervals, $|\mathbf{R}|$ does not exceed h^2.*

Proof. We prove this by induction. Clearly if $h = 1$, the number of elements of $\mathbf{R}$ is 1, which is h^2.

The algorithm generates regions by iterating over the set of PHIs. Assume that it is constructing a set of regions for $h+1$ PHIs. Further assume that after iterating over the first h PHIs, it holds that for the number x of elements in $\mathbf{R}$, $x \leq h^2$.

The insertion of PHI P_{h+1} can add a region to $\mathbf{R}$ for every point at which its objective $O_{P_{h+1}}^*$ intersects the objective $O_{P_i}^*$ for each PHI i, $1 \leq i \leq h$. As there are two such possible intersections for each PHI, this can produce up to $2h$ additional regions. Furthermore, an additional region might be added if $U_{h+1}^{\min}$ is less than $R_1^{\min}$, the current minimum utilization covered by $\mathbf{R}$. Alternatively, an additional region might be added if $U_{h+1}^{\min} > R_j^{\min}$ for the first region R_j where $U_{h+1}^{\min} < R_j^{\max}$. As both conditions cannot be true, it follows that only up to $2h+1$ additional regions may be added.

Then the total number of regions after insertion of P_{h+1} does not exceed $x + 2h + 1 \leq h^2 + 2h + 1 = (h+1)^2$. □

Corollary 1. *For n tasks and $k = T_n^{\max}/T_1^{\min}$ as in §IV, the number of regions in $\mathbf{R}$ does not exceed $k(n-1)^{2\lfloor \log k \rfloor}$.*

Proof. From Thm. 4, we know that for the number of PHIs h does not exceed $k(n-1)^{\lfloor \log k \rfloor}$. Then from Thm. 5, for h PHIs the number of regions does not exceed h^2. □

Corollary 2. *For n tasks and $k = T_n^{\max}/T_1^{\min}$ as in §IV, binary search over the set of regions $\mathbf{R}$ takes time $\mathcal{O}(\log^2 k + \log k \cdot \log n)$*

Proof. For a sorted list of x elements, binary search requires $\mathcal{O}(\log x)$ time. Then from Corollary 1, $|\mathbf{R}| \leq k(n-1)^{2\lfloor \log k \rfloor}$. So binary search proceeds in time:

$$\log\left(k(n-1)^{2\lfloor \log k \rfloor}\right) = 2\lfloor \log k \rfloor \left(\log k + \log(n-1)\right)$$

This is $\mathcal{O}(\log^2 k + \log k \cdot \log n)$, which is polynomial in the length of the input. □

VI. Evaluation

A. Characterizing Elasticity

The elasticity constant E_i assigned to task τ_i is intended to represent "the flexibility of the task to vary its utilization" [1]. The elastic objective (Eqn. 1a) as formulated by Chantem et al. in [14], [15] suggests the first-order error induced by individually decreasing the rate R_i of task τ_i from its fastest desired value $R_i^{\max}$ can be described as:

$$\mathcal{L} = w_i(R_i^{\max} - R_i)^2 \tag{20}$$

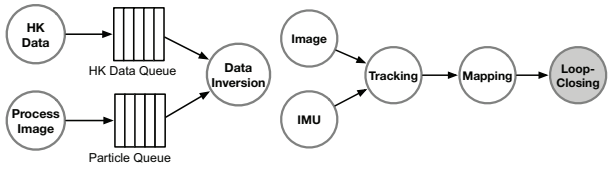
where $w_i = C_i^2/E_i$ and $\mathcal{L}$ is some measure of loss or error. Then for a given application, if the set of weights $\{w_i\}$ are obtained, each task's elasticity can be assigned as:

$$E_i = C_i^2/w_i \tag{21}$$

We use this interpretation to assign elasticity constants to each task in our evaluated applications.

B. FIMS

The Fast Integrated Mobility Spectrometer (FIMS) [6] is a flown instrument that characterizes atmospheric aerosols. Recent efforts to enable real-time measurements of aerosol particle sizes (e.g., to instruct the aircraft to follow an aerosol plume) achieved the desired performance on a Raspberry Pi 4 [37]. The improved computational pipeline captures and processes images to detect particles, grouping them into fixed-duration windows of particle inlet time. Using matrix inversion, it converts instrument responses from particle spatial coordinates to determine the particle size distribution within each window. "Housekeeping" (HK) data readouts from other sensors (e.g., temperature and pressure) are synchronized with the inversion process, and are used to determine if a new data inversion matrix must be computed. Harmonic execution



(a) FIMS pipeline. (b) ORB-SLAM3 pipeline.

Fig. 6: Dataflow pipelines of the tested applications.

guarantees a fixed, integral number of time windows associated with each job. This stabilizes the fraction of particles lost during the inversion process, ensuring a consistent representation of particle distribution and maintaining measurement accuracy, while also bounding the latency between a particle's inlet time and its subsequent inclusion in the reported size distribution.

In our evaluation, we consider a future deployment where FIMS runs concurrently with other applications (e.g., flight control, SLAM, and telemetry) atop SWaP-constrained hardware, e.g., on a UAV. To examine its portability across embedded platforms, we experiment with constraining FIMS to just a fraction of the bandwidth of a *single* core of the Pi 4 and verify that, by using harmonic elastic scheduling to adjust the FIMS task periods, we can avoid missing deadlines.

Experimental Setup: We run FIMS on a Raspberry Pi 4 Model B, which has a 4-core, 64-bit Cortex-A72 (ARM v8) CPU running at 1.50GHz and 4GB of RAM. We test with Linux 5.10.103 and disable CPU throttling. Image processing, HK data reading, and data inversion are all pinned to the same CPU core and run under the `SCHED_FIFO` real-time scheduling class using rate monotonic priorities. Lacking direct access to the prototype FIMS instrument, we instead use an offline dataset consisting of camera images from its particle chamber and readouts from its other sensors. To avoid delays from disk I/O, we load these into program memory prior to execution. In a complete implementation, separate threads will perform the memory transfer from attached USB devices asynchronously. A complete run with our experimental dataset handles $12\,000$ images, processed at a period of 100 ms. HK data is read every 500 ms and data inversion runs every second.

Profiling Execution Times: To measure execution times associated with each task, we make calls to `getrusage()` when each job completes, measuring the total CPU time (user and system) consumed by the task since the end of the prior job. This accounts for execution of the task's function plus the overhead of context switching and timer handling. To capture worst-case conditional behavior, we force recalculation of the inversion matrix with each iteration of data inversion. Profiling results over 10 complete runs are plotted in Fig. 7d – 7f.

Assigning Elasticity Values: We assign elasticity values to tasks per the methodology in §VI-A. We define loss as $1000 \cdot (1 - \theta)$, where θ is the cosine similarity between the distribution of particle sizes produced by a period-adjusted run of FIMS and the ground-truth values. We measure the loss associated with adjusting task periods individually. To reflect the real instrument's behavior, when increasing the image processing period we stack successive image frames. When increasing the HK data period by a factor of $n\times$, we sample every n^{th} data point, interpolating over the most recent two. For data inversion, we change the time bin over which particle size distributions are measured to remain equal to the period. Results are plotted in Fig. 7a – 7c. Using linear regression over the measured error values for each task τ_i, with $(R_i^{\max} - R_i)^2$ as the independent variable, we obtain values of w_i according to Eqn. 20, then use these to compute values of E_i per Eqn. 21. The following table summarizes the assigned parameters:

	$T_i^{\min}$	$T_i^{\max}$	C_i	E_i
Process Image	100	1000	43.0	2.11
HK Data	500	5000	0.747	0.012
Data Inversion	1000	10000	55.3	1.23

Evaluating Scalability: With these parameters, FIMS demands a maximum utilization of 0.487. To test under tighter constraints, we run a highest priority *interference task* that limits CPU utilization by FIMS: it registers an interval timer with a period of 50 ms and spins for a programmable length of time. We run FIMS concurrently with the interference task using different busy loop durations, adjusting the FIMS task periods according to our harmonic elastic model. We then measure each FIMS job's latency (elapsed wallclock time) from task release to completion; results are shown in the following table. $L^{\max}$ indicates the longest measured latency over 3 complete runs; this value never exceeds the corresponding task period, indicating that no deadlines were missed.

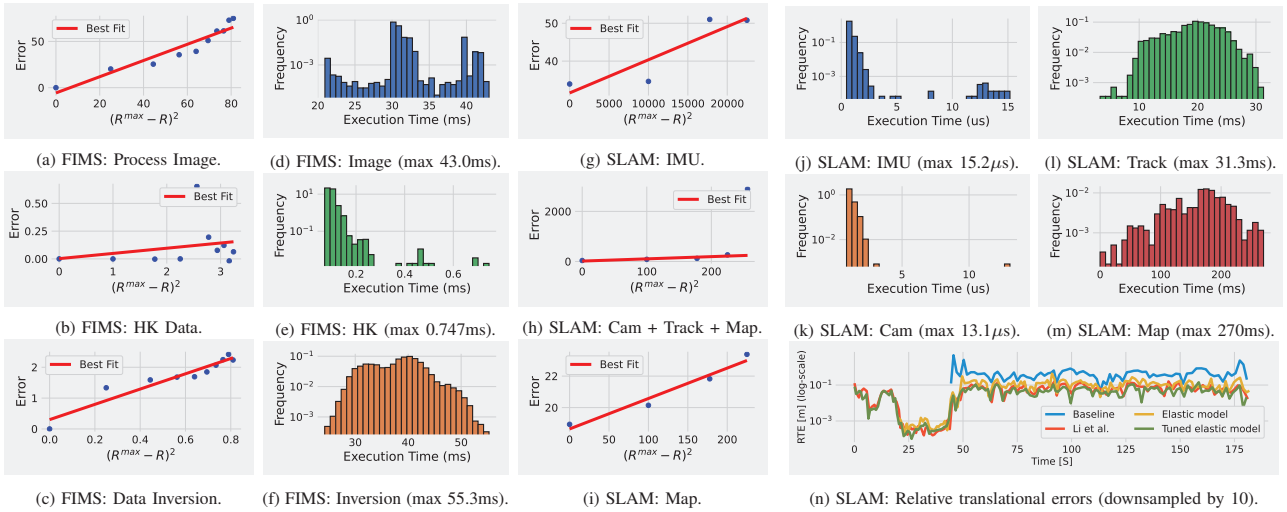
U_D	T_{IMAGE}	T_{HK}	T_{INV}	$L^{\max}_{\text{IMAGE}}$	$L^{\max}_{\text{HK}}$	$L^{\max}_{\text{INV}}$
0.5	100	500	1000	67	305	305
0.4	115	575	2298	94	489	1275
0.3	147	881	9682	137	709	822
0.2	222	3325	9973	222	1324	1327
0.1	458	3205	9615	348	2784	2785

C. ORB-SLAM3

ORB-SLAM3 [18] is a visual-inertial simultaneous localization and mapping system widely used in autonomous vehicle applications. Object tracking fuses captured image frames with interstitial inertial measurements to detect feature points and generate descriptions. These are then matched against a map database of prior descriptions to determine the system's position. If the current environment differs sufficiently from the existing map, the map is updated within the mapping task.

Our goal is to consider the portability of ORB-SLAM3 to SWaP-constrained hardware, where it may execute concurrently on a single core with other applications; the utilization U_D available to it may change during runtime. While missed deadlines do *not* necessarily result in system failure, we show that adjusting task periods in a principled way offers better localization than its baseline implementation.

Experimental Setup: We run ORB-SLAM3 on a 6-core Intel i7-4960X running at 3.6GHz with 12GB of RAM using Linux 4.9.30 built with LITMUSˆRT [38]. We disable HyperThreading and CPU throttling. We evaluate in simulation using the EuRoC MAV dataset [39] from drones in real-world environments. Realistic timing is achieved by using ROS [40] to deliver image frames and inertial measurement unit (IMU)



(a) FIMS: Process Image. (d) FIMS: Image (max 43.0ms). (g) SLAM: IMU. (j) SLAM: IMU (max 15.2μs). (l) SLAM: Track (max 31.3ms).

(b) FIMS: HK Data. (e) FIMS: HK (max 0.747ms). (h) SLAM: Cam + Track + Map. (k) SLAM: Cam (max 13.1μs). (m) SLAM: Map (max 270ms).

(c) FIMS: Data Inversion. (f) FIMS: Inversion (max 55.3ms). (i) SLAM: Map. (n) SLAM: Relative translational errors (downsampled by 10).

Fig. 7: Experimental results on FIMS and ORB-SLAM3.

data as messages. IMU data arrives every 5 ms, and camera frames every 50 ms; a single trace includes 187 seconds worth of data. Tracking executes with as period of 50 ms, and by default, mapping executes only when tracking identifies a keyframe (with a minimum period of 50 ms).

Profiling Execution Times: We measure the execution times of each task by successive calls to `clock_gettime()` using `CLOCK_THREAD_CPUTIME_ID`. Results for each task, profiled over a complete trace, are shown in Fig. 7j – 7m.

Assigning Elasticity Values: For ORB-SLAM3, we define loss as $10^4 X$, where X is the relative translational error (RTE) of the completed map in units of meters from the ground truth. We again capture the first-order effects of adjusting task periods individually. Here, we treat the two stereo image processing threads and the tracking thread as a single task (the application's dataflow pipeline requires these to run at the same rate). We simulate increases in the IMU and image task periods by dropping message frames (e.g., for an IMU period of 10 ms, we drop every other IMU message).

To achieve more deterministic execution time behavior, we modify tracking to categorize every image frame as a keyframe – in a non-degraded state, mapping will perform updates for every frame. However, if utilizations are compressed due to overload, keyframes are selected according to the harmonic relationship between the mapping and tracking periods. To characterize the impact on loss, we hold other periods constant while decreasing the number of frames selected as keyframes (thus increasing the mapping period). We slow the replay of the EuRoC dataset to allow mapping to process every frame as a keyframe without overrunning its deadline.

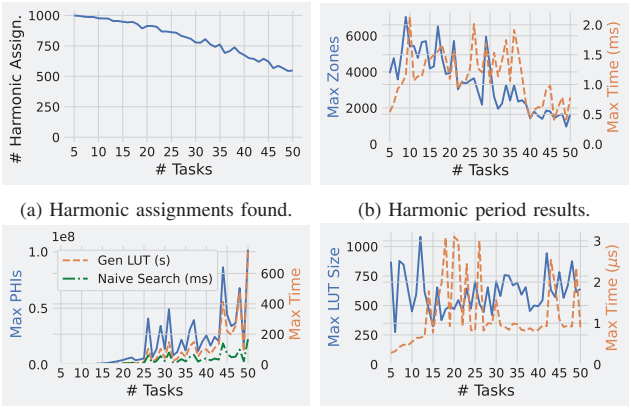
Results are plotted in Fig. 7g – 7i. We again use linear regression to fit values of w_i (Eqn. 20). Observe that in Fig. 7h, the last point (corresponding to a period of 250 ms) diverges sharply from the linear trend. This suggests unstable behavior, so we limit the fit (and $T_i^{\max}$) to 200 ms. Note also that when increasing the image and tracking periods, the mapping period is also increased (its period cannot be *less* than these tasks); we adjust the first-order weight that characterizes the image and tracking task accordingly. After computing values of E_i (Eqn. 21), we obtain the following task parameters:

	$T_i^{\min}$	$T_i^{\max}$	C_i	E_i
IMU	5	20	0.015	0.263
Camera + Tracking	50	200	31.3	4006
Mapping	50	1200	270	1.14e5

Evaluation of Online Adjustment: We artificially limit the CPU utilization available to ORB-SLAM3 by pinning all of its threads on a single core and restrict its total bandwidth with Linux cgroups. We modify the beginning of its main loop to (1) select a random amount of available utilization between 0.50–0.75, (2) enforce this by updating the cgroup, (3) run the online algorithm described in §V to search the lookup table (generated during program initialization) for the best PHI corresponding to that utilization, then (4) update task periods accordingly. This is invoked every second to allow completion of at least one hyperperiod, and incurs an overhead of $< 22\mu$s to select and update the period assignments.

We run ORB-SLAM3 in this manner over EuRoC trace MH_01, comparing its accuracy to an unmodified (non-adaptive) baseline version of the program. Results are plotted in Fig. 7n. In our constrained execution environment, the mean RTE of the baseline implementation (Baseline) is 89mm, and only succeeds in mapping the final 139s of the flight. In contrast, the adaptive implementation (Elastic model) achieves a mean RTE of 13mm, a 6.6× reduction, and maps the final 164s. Using insights from past experience, we also try using the mean observed execution times for the mapping, camera, and tracking workloads (Tuned elastic model). This achieves even better accuracy: the mean RTE is only 8.6mm, a 10.4× reduction, and maps the entire flight.

Finally, we compare our results to the adaptive approach of Li et al. in [19] where ORB-SLAM3 is integrated with an online machine learning model that predicts budget constraints and adapts execution via selective frame dropping, achieving a mean RTE of 9.3mm. Not only does our elastic model achieve better localization, it is more suited for broad use across real-time applications than the approach in [19]: on a hard real-



(a) Harmonic assignments found. (b) Harmonic period results.

(c) Harmonic elastic results. (d) Fast LUT results.

Fig. 8: Evaluation results for synthetic task sets.

time system, task dropping may be unacceptable, so our elastic model extends task periods to avoid missed deadlines. Furthermore, our approach enables straightforward adaptation of execution parameters using only a basic characterization of the impacts on control performance and does not require highly-tailored ML-based integration with the existing application.

D. Evaluation with Larger Synthetic Task Sets

The ORB-SLAM3 and FIMS task sets are relatively small: in this subsection, we explore how task set size impacts the performance of the algorithms discussed in §IV-B and §V using randomly generated task parameters.

Experimental Setup: We generate 1000 task sets each of sizes 5–50. The minimum period $T_i^{\min}$ of each task τ_i is sampled from the log-uniform distribution described in [41] over integers in the range $[1, 100]$. A maximum period $T_i^{\max}$ is obtained by multiplying each $T_i^{\min}$ by a value selected uniformly in $[1, 10]$; this bounds the ratio k of the largest to smallest periods at 1000. Each task set is assigned a total maximum utilization of 1; individual task utilizations $U_i^{\max}$ are then assigned using the UUniSort algorithm [42]. C_i is derived as $U_i^{\max} \cdot T_i^{\min}$. Weights w_i are selected uniformly in $[0, 1]$, from which elasticities E_i are computed according to Eqn. 21. Tasks are then sorted by $T_i^{\min}$. We implement the algorithms in C++, and compile them with GCC optimization level `-O3`. All experiments are run on a single core of an AMD EPYC 9754 with 128GB of RAM running Linux 5.14.0.

The Harmonic Period Problem: We first evaluate the proposed approach to the harmonic period problem described in §IV-B. For each number of tasks, we count how many of the 1000 corresponding sets had a feasible harmonic period assignment, how long it took to find such an assignment, and how many harmonic zones were projected onto the last interval. In Fig. 8a, we see that as the number of tasks increases, the proportion of task sets for which a solution exists goes down. We see in Fig. 8b that the maximum number of harmonic zones projected onto the last interval also *decreases* as the number of tasks increases, even though our theoretical bound increases. We conjecture that this is because adding more period intervals will typically impose tighter constraints on the harmonic search. We also see in Fig. 8b that the algorithm is efficient, executing in under 2.2 ms. Its observed maximum execution times do not have a tight relationship with the number of harmonic zones projected onto the last interval.

The Ordered Harmonic Elastic Problem: We then evaluate both the naïve and efficient approaches to the ordered harmonic elastic problem described in §V. For those task sets where a harmonic assignment is possible, we count how many projected harmonic intervals (PHIs) are found, how long it takes to generate the lookup table (LUT), and how many entries it has. We also compare the time it takes to iterate over all PHIs to find the optimal period assignment, versus performing binary search. In Fig. 8c, we observe that the maximum number of PHIs grows with the number of tasks. The time to generate the lookup table, as well as for the naïve search, remain proportional to the number of PHIs. The naïve search does not exceed 170 ms; it is reasonably efficient, and provides greater flexibility as it allows for online admission of new harmonic tasks, though it might be too slow for online use with larger task systems. Times to generate the LUT remain under 10 ms for up to 8 tasks, but reach as high as 12.6 minutes for up to 50 tasks. This suggests that for smaller task systems, recomputing a set of harmonic assignments *online* if a new elastic task is admitted to the system may be feasible. For larger task sets, slow offline computation of the lookup table nonetheless allows for rapid online adaptation of existing task periods: searching the lookup table and assigning new periods does not exceed 3.2 μs, as shown in Fig. 8d.

VII. Conclusions and Future Work

In this work, we have considered how to extend elastic scheduling to systems of tasks with harmonic periods. We argue that, if every task's period is constrained to a continuous interval, then finding a set of harmonic periods is unlikely to have a polynomial time solution. However, we outline a pseudo-polynomial algorithm to solve the problem, and show it is efficient even for large task sets. We demonstrate that our approach allows applications such as FIMS and SLAM to be deployed effectively atop SWaP-constrained hardware platforms. Quantifying the impact on result error of increasing individual task periods lets us successfully assign values to each task, and for ORB-SLAM3, this approach allowed for a $10.4\times$ decrease in localization error compared to a baseline implementation on a dynamic resource-constrained system.

As future work, we will extend to tasks with elastic execution times, and consider the case where workloads depend on the selected period ratios. We will also explore the implications of adjusting periods during (rather than between) hyperperiods.

Acknowledgment

This research was supported by NSF grants CPS-2229290, CNS-2141256, CNS-2038995, and CNS-2238635; a Washington University CSE/EECE seed grant; and Swedish Research Council grant 2018-04446.

REFERENCES

[1] G. C. Buttazzo, G. Lipari, and L. Abeni, "Elastic task model for adaptive rate control," in *Proc. of IEEE Real-Time Systems Symposium*, 1998. [Online]. Available: https://doi.org/10.1109/REAL.1998.739754

[2] G. C. Buttazzo, G. Lipari, M. Caccamo, and L. Abeni, "Elastic scheduling for flexible workload management," *IEEE Transactions on Computers*, vol. 51, no. 3, pp. 289–302, Mar. 2002. [Online]. Available: http://dx.doi.org/10.1109/12.990127

[3] H. Li, J. Sweeney, K. Ramamritham, R. Grupen, and P. Shenoy, "Real-time support for mobile robotics," in *The 9th IEEE Real-Time and Embedded Technology and Applications Symposium, 2003. Proceedings.*, 2003, pp. 10–18.

[4] J. Orr, J. C. Uribe, C. Gill, S. Baruah *et al.*, "Elastic scheduling of parallel real-time tasks with discrete utilizations," in *Proc. of 28th International Conference on Real-Time Networks and Systems*. ACM, 2020, pp. 117–127. [Online]. Available: https://doi.org/10.1145/3394810.3394824

[5] J. Leonard and H. Durrant-Whyte, "Simultaneous map building and localization for an autonomous mobile robot," in *Proceedings IROS '91:IEEE/RSJ International Workshop on Intelligent Robots and Systems '91*, 1991, pp. 1442–1447 vol.3.

[6] J. Wang, M. Pikridas, S. R. Spielman, and T. Pinterich, "A fast integrated mobility spectrometer for rapid measurement of sub-micrometer aerosol size distribution, part i: Design and model evaluation," *Journal of Aerosol Science*, vol. 108, pp. 44–55, 2017. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0021850216304426

[7] A. Li, J. Wang, S. Baruah, B. Sinopoli, and N. Zhang, "An empirical study of performance interference: Timing violation patterns and impacts," in *2024 Real-Time and Embedded Technology and Applications Symposium (RTAS)*. IEEE, 2024.

[8] V. Brocal, P. Balbastre, R. Ballester, and I. Ripoll, "Task period selection to minimize hyperperiod," in *ETFA2011*, 2011, pp. 1–4.

[9] I. Ripoll and R. Ballester-Ripoll, "Period selection for minimal hyperperiod in periodic task systems," *IEEE Transactions on Computers*, vol. 62, no. 9, pp. 1813–1822, 2013.

[10] H. Kopetz, "On the design of distributed time-triggered embedded systems," *Journal of Computing Science and Engineering*, vol. 2, no. 4, pp. 340–356, 2008.

[11] 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 Syst.*, vol. 2, no. 4, p. 301–324, oct 1990. [Online]. Available: https://doi.org/10.1007/BF01995675

[12] M. Nasri, G. Fohler, and M. Kargahi, "A framework to construct customized harmonic periods for real-time systems," in *2014 26th Euromicro Conference on Real-Time Systems*, 2014, pp. 211–220.

[13] M. Nasri and G. Fohler, "An efficient method for assigning harmonic periods to hard real-time tasks with period ranges," in *2015 27th Euromicro Conference on Real-Time Systems*, 2015, pp. 149–159.

[14] T. Chantem, X. S. Hu, and M. D. Lemmon, "Generalized elastic scheduling," in *Proc. of IEEE International Real-Time Systems Symposium*, 2006, pp. 236–245. [Online]. Available: https://doi.org/10.1109/RTSS.2006.24

[15] ——, "Generalized elastic scheduling for real-time tasks," *IEEE Transactions on Computers*, vol. 58, no. 4, pp. 480–495, Apr. 2009. [Online]. Available: https://doi.org/10.1109/TC.2008.175

[16] G. Bunting, P. Lindsay, A. Maghareh, A. Prakash, and S. Dyke, "Using multi-time stepping in finite element models to meet real time constraints," in *EMI/PMC 2012 Joint Conference of the Engineering Mechanics Institute and the 11th ASCE Joint Specialty Conference on Probabilistic Mechanics and Structural Reliability*, 2012.

[17] R. Mur-Artal, J. M. M. Montiel, and J. D. Tardós, "Orb-slam: A versatile and accurate monocular slam system," *IEEE Transactions on Robotics*, vol. 31, no. 5, pp. 1147–1163, 2015.

[18] C. Campos, R. Elvira, J. J. G. Rodríguez, J. M. M. Montiel, and J. D. Tardós, "Orb-slam3: An accurate open-source library for visual, visual–inertial, and multimap slam," *IEEE Transactions on Robotics*, vol. 37, no. 6, pp. 1874–1890, 2021.

[19] A. Li, H. Liu, J. Wang, and N. Zhang, "From timing variations to performance degradation: Understanding and mitigating the impact of software execution timing in slam," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2022.

[20] M. Sudvarg, C. Gill, and S. Baruah, "Linear-time admission control for elastic scheduling," *Real-Time Systems*, vol. 57, no. 4, pp. 485–490, Oct 2021. [Online]. Available: https://doi.org/10.1007/s11241-021-09373-4

[21] J. Orr and S. Baruah, "Multiprocessor scheduling of elastic tasks," in *Proc. of 27th International Conference on Real-Time Networks and Systems*. ACM, 2019, pp. 133–142. [Online]. Available: https://doi.org/10.1145/3356401.3356403

[22] S. Baruah, "Improved uniprocessor scheduling of systems of sporadic constrained-deadline elastic tasks," in *Proceedings of the 31st International Conference on Real-Time Networks and Systems*, ser. RTNS '23. New York, NY, USA: Association for Computing Machinery, 2023, p. 67–75. [Online]. Available: https://doi.org/10.1145/3575757.3575759

[23] M. Sudvarg, S. Baruah, and C. Gill, "Elastic scheduling for fixed-priority constrained-deadline tasks," in *2023 IEEE 26th International Symposium on Real-Time Distributed Computing (ISORC)*, 2023, pp. 11–20.

[24] J. Li, J. J. Chen, K. Agrawal, C. Lu, C. Gill, and A. Saifullah, "Analysis of federated and global scheduling for parallel real-time tasks," in *Proc. of 26th Euromicro Conference on Real-Time Systems*, 2014, pp. 85–96. [Online]. Available: https://doi.org/10.1109/ECRTS.2014.23

[25] J. Orr, C. Gill, K. Agrawal, J. Li, and S. Baruah, "Elastic scheduling for parallel real-time systems," *Leibniz Transactions on Embedded Systems*, vol. 6, no. 1, p. 05:1–05:14, May 2019. [Online]. Available: https://ojs.dagstuhl.de/index.php/lites/article/view/LITES-v006-i001-a005

[26] J. Orr, C. Gill, K. Agrawal, S. Baruah *et al.*, "Elasticity of workloads and periods of parallel real-time tasks," in *Proc. of 26th International Conference on Real-Time Networks and Systems*. ACM, 2018, pp. 61–71. [Online]. Available: https://doi.org/10.1145/3273905.3273915

[27] T.-W. Kuo and A. Mok, "Load adjustment in adaptive real-time systems," in *[1991] Proceedings Twelfth Real-Time Systems Symposium*, 1991, pp. 160–170.

[28] C.-C. Han and H.-Y. Tyan, "A better polynomial-time schedulability test for real-time fixed-priority scheduling algorithms," in *Proceedings Real-Time Systems Symposium*, 1997, pp. 36–45.

[29] C.-C. Han and K.-J. Lin, "Scheduling distance-constrained real-time tasks," in *[1992] Proceedings Real-Time Systems Symposium*, 1992, pp. 300–308.

[30] C.-S. Shih, S. Gopalakrishnan, P. Ganti, M. Caccamo, and L. Sha, "Scheduling real-time dwells using tasks with synthetic periods," in *RTSS 2003. 24th IEEE Real-Time Systems Symposium, 2003*, 2003, pp. 210–219.

[31] M. Fan and G. Quan, "Harmonic semi-partitioned scheduling for fixed-priority real-time tasks on multi-core platform," in *2012 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, 2012, pp. 503–508.

[32] N. Min-Allah, I. Ali, J. Xing, and Y. Wang, "Utilization bound for periodic task set with composite deadline," *Computers & Electrical Engineering*, vol. 36, no. 6, pp. 1101–1109, 2010. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0045790610000406

[33] D. Ferry, G. Bunting, A. Maghareh, A. Prakash, S. Dyke, K. Agrawal, C. Gill, and C. Lu, "Real-time system support for hybrid structural simulation," in *Proceedings of the 14th International Conference on Embedded Software*, ser. EMSOFT '14. New York, NY, USA: Association for Computing Machinery, 2014. [Online]. Available: https://doi.org/10.1145/2656045.2656067

[34] M. Mohaqeqi, M. Nasri, Y. Xu, A. Cervin, and K.-E. Årzén, "Optimal harmonic period assignment: complexity results and approximation algorithms," *Real-Time Systems*, vol. 54, no. 4, pp. 830–860, Oct 2018. [Online]. Available: https://doi.org/10.1007/s11241-018-9304-0

[35] I. Pavić and H. Džapo, "Optimal harmonic period assignment with constrained number of distinct period values," *IEEE Access*, vol. 8, pp. 175 697–175 712, 2020.

[36] P. L. Montgomery, "A survey of modern integer factorization algorithms," *CWI quarterly*, vol. 7, no. 4, pp. 337–366, 1994.

[37] D. Wang, J. Zhang, J. Buhler, and J. Wang, "Real-time analysis of aerosol size distributions with the fast integrated mobility spectrometer (FIMS)," in *41st Conference of American Association for Aerosol Research (AAAR)*, Oct. 2023. [Online]. Available: https://aaarabstracts.com/2023/view_abstract.php?pid=752

[38] J. M. Calandrino, H. Leontyev, A. Block, U. C. Devi, and J. H. Anderson, "$LITMUS^{RT}$: A testbed for empirically comparing real-time multiprocessor schedulers," in *2006 27th IEEE International Real-Time Systems Symposium (RTSS'06)*, 2006, pp. 111–126.

[39] M. Burri, J. Nikolic, P. Gohl, T. Schneider, J. Rehder, S. Omari, M. W. Achtelik, and R. Siegwart, "The euroc micro aerial vehicle datasets," *The International Journal of Robotics Research*, vol. 35, no. 10, pp. 1157–1163, 2016.

[40] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, A. Y. Ng *et al.*, "Ros: an open-source robot operating system," in *ICRA workshop on open source software*, vol. 3, no. 3.2. Kobe, Japan, 2009, p. 5.

[41] P. Emberson, R. Stafford, and R. Davis, "Techniques for the synthesis of multiprocessor tasksets," in *WATERS workshop at the Euromicro Conference on Real-Time Systems*, Jul. 2010, pp. 6–11.

[42] E. Bini and G. C. Buttazzo, "Measuring the performance of schedulability tests," *Real-Time Syst.*, vol. 30, no. 1–2, p. 129–154, May 2005.