A Fully Polynomial Time Approximation Scheme for Adaptive Variable Rate Task Demand

Aaron Willcock
willcock@wayne.com
Wayne State University
Detroit, MI, USA

Nathan Fisher
fishern@wayne.edu
Wayne State University
Detroit, MI, USA

Thidapat Chantem
tchantem@vt.edu
Virginia Tech
Arlington, VA, USA

ABSTRACT

The Adaptive Variable Rate (AVR) task model defines a task where job WCET and period are a function of engine speed. Motivated by a lack of tractable AVR task demand methods, this work uses predefined job sequences for the Bounded Precedence Constraint Knapsack Problem inherent in AVR task demand calculation instead of enumerating all considered speeds as in existing work. A new, exact approach is proposed and approximated, enabling the derivation of a Fully Polynomial Time Approximation Scheme that outperforms the state-of-the-art in runtime (7,800x improvement) and RAM use (99% reduction) with less than 8% demand overestimate.

CCS CONCEPTS

• **Computer systems organization** → **Embedded software**; **Real-time system specification**.

KEYWORDS

Adaptive variable rate, task demand, bounded precedence constraint knapsack problem, fully polynomial time approximation scheme

ACM Reference Format:
Aaron Willcock, Nathan Fisher, and Thidapat Chantem. 2024. A Fully Polynomial Time Approximation Scheme for Adaptive Variable Rate Task Demand. In *The 32nd International Conference on Real-Time Networks and Systems (RTNS 2024), November 7–8, 2024, Porto, Portugal.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3696355.3696367

1 INTRODUCTION AND MOTIVATION

This research aims to add demand characterization tools to the real-time community toolbox for the effective deployment of real-time, safety-critical cyber-physical systems (CPSs). CPSs are defined by the tight integration of physical dynamics, computation, and control [46]. In these systems, Model Based Systems Engineering (MBSE) is used to construct a system model and implementation [41, 51]. Example MBSE tools include PRISM [40] and UPPAAL [2, 53]. When mapping computational loads to real-time tasks, conventional task models (e.g., periodic and sporadic [42, 45]) do not always align with the modeled CPS and cause overprovisioning. Consider variable sampling rate systems like wearable devices which increase sampling frequency when user activity is detected [3, 16, 18], Brushless DC (BLDC) motors whose control sampling increases with speed [30–32, 35], or satellite magnetorquers whose actuation frequency increases with positional error [14, 15]. Using a periodic task to model the workload with the highest possible frequency (e.g., during activity in the wearable, at high speed in the BLDC motor controller, or high positional error in the satellite) is a safe, valid configuration. However, this results in overprovisioning when the actual sampling frequency is smaller. This inefficiency manifests whenever the system is not operating at maximum frequency. Ideally, custom task models are created during MBSE to mitigate overprovisioning. These models require specialized schedulability analysis tools (e.g., response time analysis, utilization bounds) to allow predictable mingling with other tasks. One tool is the demand bound function, which offers an algorithm-agnostic upper bound on computational workload.

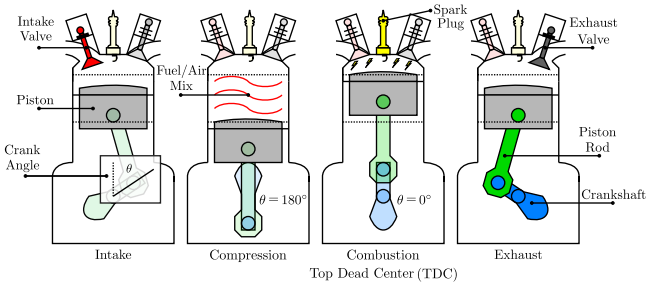

Figure 1: Relevant piston/crankshaft angles for AVR tasks

Consider an Engine Control Unit (ECU) in modern internal combustion engine (ICE) vehicles in which a task releases a job each time the piston reaches Top Dead Center within the cylinder bore, as illustrated in Figure 1. In an ICE, pistons are connected to a crankshaft rotating at a given speed (hereafter *engine speed*). As engine speed increases, release frequency increases and vice versa. Each job has an associated Worst-Case Execution Time (WCET) which corresponds with engine speed as illustrated in Figure 2. Finding the maximum computational workload over some time interval using this model is inherently difficult since engine acceleration (and deceleration) allow an infinite number of job release sequences [8]. This problem is further complicated by changing task parameters [43, 47] where demand recharacterization occurs repeatedly during operation or where Design Space Exploration (DSE) is performed offline, testing hardware-software combinations for feasible systems (e.g., [1, 26, 37]).

RTNS 2024, November 7–8, 2024, Porto, Portugal

ACM ISBN 979-8-4007-1724-6/24/11.
https://doi.org/10.1145/3696355.3696367

1.1 Related Work

The original engine model was proposed as the Rhythmic Task Model [39]. Further investigation was spurred by Buttle [13]. The Rhythmic Task Model was then adopted via Adaptive Variable Rate (AVR) [6, 11] and Variable Rate Behavior (VRB) [17, 22] models.

Varied task models and parameter assumptions can be reviewed in Feld et al. [20] and Shambharkar et al. [50]. Since job interarrival time depends on engine speed and acceleration, different acceleration assumptions are explored (e.g., constant acceleration between speeds [8, 9, 11], maximum acceleration between modes [25], and variable acceleration between speeds [5, 44]).

Existing analysis for variable-rate tasks cover utilization [12, 25], response time [9, 10, 19, 21, 23, 48], interference [22, 49] and demand characterization [5, 8, 44]. Existing works also examine variable-rate tasks under EDF [8, 12, 25, 29, 44], Fixed Priority [11, 19, 21–23, 48], and Dynamic Priority [7].

To our knowledge, the only exact demand characterizations for variable period engine models are Biondi et al. [8], Mohaqeqi et al. [44] and Bijinemula et al. [5]. All other works provide a different analysis (see above). Of the near-peer, exact demand characterization works, only two allow variable acceleration between job releases: Mohaqeqi et al. [44] and Bijinemula et al. [5] – both AVR-based works. The former uses a search technique based on the Digraph Real-Time model [52] while the latter formulates and solves the problem as a Bounded Precedence Constraint Knapsack Problem (BPCKP) [24, 36] - yielding faster runtimes.

Despite its relative speed, the BPCKP approach is a precision-sensitive, numeric method (i.e., increasing precision greatly increases runtime and memory use), making it intractable when task parameters change online and demand must be recalculated frequently (e.g., Connected Autonomous Vehicles changing WCET to engine speed ratios online [43, 47]) or when system designers employ DSE to quickly generate hardware-software combinations to test feasibility in advance (e.g., [1, 26, 37]). To provide a more tractable solution, we seek an approximation of AVR task demand.

Fortunately, existing approximations of AVR demand via utilization are found in Guo and Baruah [25] and by applying Stigge et al.'s concept of the "most dense cycle" [52]. Guo and Baruah offer a speedup factor proof to bound the maximum "wasted" processor capacity at 13% for AVR tasks. However, this bound does not hold when the maximum AVR task utilization is large. Moreover, a linear approximation using Stigge et al.'s digraph-based "most dense cycle" can result in over 40% overestimation (see Appendix for both examples). To mitigate overestimation and pursue tractable runtime, we consult existing BPCKP approximations.

Given that the BPCKP approach offered by Bijinemula et al. is a knapsack problem variant [5], existing knapsack approximations appear as obvious solutions. For context, Keller et al. cover the knapsack problem and variants thereof [38]. Ibarra and Kim [33] give a Fully Polynomial Time Approximation Scheme (FPTAS) for the 0-1 knapsack problem for which Vazirani [54] offers a simplified version. Ibarra and Kim [34] also define and solve the MAXPROFIT problem, a knapsack variant where precedence-constrained processes must be scheduled to maximize profit over a time interval. Already, the MAXPROFIT problem is very similar to computing maximum AVR task demand. Garey and Johnson formalize this MAXPROFIT problem generally as the Partially Ordered Knapsack (POK) and prove its NP-Completeness [24]. Although NP-Complete, POK with precedence constraints *in tree form* admits pseudopolynomial runtime per Johnson and Niemi's [36]. Thankfully, the BPCKP approach *does* produce a precedence graph as an out-tree. However, the approach leaves input parameters in terms of kinematic values (e.g., engine speed, acceleration). In other words: the BPCKP out-tree is not bounded by the problem size (i.e., the number of modes). Instead, the out-tree is bounded by the engine speed, acceleration, and interval size *values*. Thus, the FPTAS offered by Johnson and Niemi is insufficient without a kinematic-independent approach.

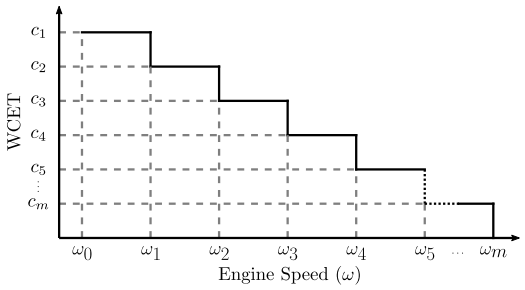

Figure 2: WCET vs. Engine Speed adapted from [11]

1.2 Contributions and Outline

To improve AVR task demand tractability, this work contributes:

(1) a BPCKP AVR task demand formulation which is polynomial in kinematic parameters (Sections 3 and 4),
(2) an exact dynamic programming solution and FPTAS for the above demand formulation (Section 5 and 6), and
(3) a comparison of the proposed FPTAS against the state-of-the-art, exact AVR task demand approach (Section 7) yielding over 7800x runtime improvement with less than 8% demand overestimate and 99% RAM usage reduction.

The following section covers the preliminary background, a formal problem statement, and a solution overview. The remaining sections cover the contributions above in the order listed.

2 PRELIMINARY BACKGROUND

This section presents the fundamentals of the AVR task model, relevant kinematic equations, and the BPCKP. It concludes with a formal problem definition and solution overview.

2.1 The AVR Task Model

First introduced by Biondi et al. [11], the Adaptive Variable Rate task model defines a real-time task with a variable period designed for engine control. In ICEs, linearly traveling pistons drive a rotating crankshaft [27, 28]. Shown in Figure 1, a piston reaches Top Dead Center (TDC) when the crankshaft angle is $\theta = 0°$. In the AVR model, jobs are released each time the piston reaches TDC — once per revolution. Thus, engine speed dictates release frequency such that lower speed increases job interarrival time and higher speed decreases job interarrival time. The AVR model also has discrete

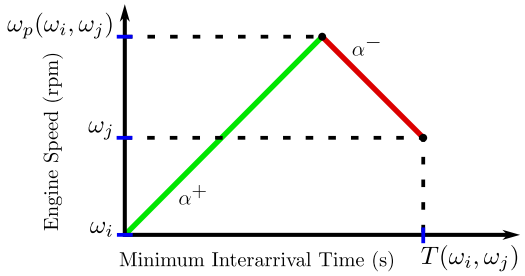

Figure 3: Speed pattern not limited by ω_m, adapted from [5]

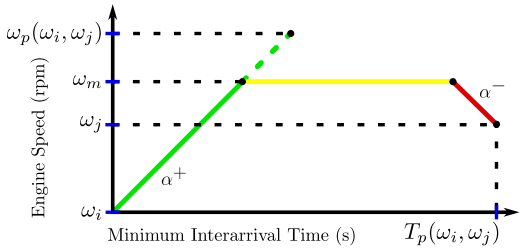

Figure 4: Speed pattern limited by ω_m, adapted from [5]

modes with job WCET non-increasing with engine speed given by:

$$\mathbf{C} = \{c_1, c_2, \ldots, c_m\} \mid c_i \in \mathbb{Z}_1^+, c_i > c_{i+1} \forall i \in \mathbb{Z}_1^m \tag{1}$$

where c_i is the WCET in microseconds (μs), c_1 is the largest WCET of any mode as shown in Figure 2, and $m = |\mathbf{C}|$ is the number of modes [1]. The *boundary speeds* at which the modes change are:

$$\mathbf{\Omega} = \{\omega_0, \omega_1, \omega_2, \ldots \omega_m\} \tag{2}$$

where ω_m is the maximum speed. The WCET of any speed is then:

$$C(s) = \begin{cases} c_1 & \text{if } s = \omega_0 \\ c_i & \text{if } \omega_{i-1} < s \leq \omega_i \end{cases} \tag{3}$$

where s is the speed in revolutions per minute (rpm). Note the first case handles the slowest possible speed, ω_0, which has WCET c_1.

To properly calculate the minimum interarrival time (MIAT) for an AVR task, the *maximum acceleration*, α^+, and *maximum deceleration*, α^-, of the engine must be defined. These values may be symmetric (i.e., in Bijinemula et al. [5]) or asymmetric (i.e., in Mohaqeqi et al. [44]). For this work, the AVR task parameters are:

$$T_{avr} = (m, \mathbf{C}, \mathbf{\Omega}, \alpha^+, \alpha^-) \tag{4}$$

where acceleration is symmetric ($\alpha^+ = -\alpha^-$) [2].

2.2 Kinematic Definitions

In the AVR model, engine speed is measured and jobs are released at TDC. Since the piston is TDC once per revolution, the rotational distance between two consecutive job releases is strictly $\theta = 1$ revolution. The following kinematic equations relate MIAT, engine speed, and distance. The equations are provided in Mohaqeqi et al. and Bijinemula et al. [5, 44] and derivable from kinematic equations (e.g., [56]). Figures 3 and 4 illustrate the two possible speed versus time graphs of variable acceleration producing MIAT as calculated by equations below. In these figures, the green, yellow, and red lines have slopes α^+, zero, and α^- respectively.

The distance traveled under uniform acceleration from starting speed, ω_i, to ending speed, ω_j, using acceleration α^+ is:

$$\theta(\omega_i, \omega_j) = \frac{\omega_j^2 - \omega_i^2}{2\alpha^+} \tag{5}$$

[1] We let $\mathbb{Z}_a^b = \{x \in \mathbb{Z} \mid a \leq x \leq b\}$, $\mathbb{Z}_a^+ = \mathbb{Z}_a^{+\infty}$, and $\mathbb{R}_a^b = \{x \in \mathbb{R} \mid a \leq x \leq b\}$

[2] Assuming symmetry simplifies underlying kinematics, maintains problem intuition, and allows FPTAS comparison without converting extant work to support asymmetry.

The MIAT between two speeds with variable acceleration is:

$$T(\omega_i, \omega_j) = \frac{\sqrt{2\omega_j^2 + 4\alpha^+ + 2\omega_i^2} - \omega_j - \omega_i}{\alpha^+} \tag{6}$$

During variable acceleration (e.g., Figure 3), a peak speed is reached while accelerating from ω_i to ω_j and is given by:

$$\omega_p(\omega_i, \omega_j) = \sqrt{\frac{\omega_i^2 + 2\alpha^+ + \omega_j^2}{2}}. \tag{7}$$

When peak speed exceeds ω_m, Equation 6 is unusable (ω_m cannot be exceeded). Instead, the speed pattern in Figure 4 with MIAT:

$$T_p(\omega_i, \omega_j) = \frac{\omega_m - \omega_i}{\alpha^+} + \frac{\omega_i^2 - 2\omega_m^2 + \omega_j^2}{2\alpha^+\omega_m} + \frac{1}{\omega_m} + \frac{\omega_m - \omega_j}{\alpha^+} \tag{8}$$

is used. Incorporating the dependency on peak speed gives the MIAT between any two speeds in microseconds [3]:

$$\overline{T}(\omega_i, \omega_j) = \begin{cases} T(\omega_i, \omega_j) \cdot 6.0 \times 10^7 & \text{if } \omega_p(\omega_i, \omega_j) \leq \omega_m \\ T_p(\omega_i, \omega_j) \cdot 6.0 \times 10^7 & \text{otherwise.} \end{cases} \tag{9}$$

Note these equations require speeds to be within one revolution of one another (i.e., $\theta(\omega_i, \omega_j) \leq 1$).

2.3 Speed Sequences and Demand

Since WCET is determined by engine speed measured at TDC (when jobs are released), an in-order WCET sequence is equivalent to a sequence of engine speeds, a *speed sequence*, written as:

$$S = (s_1, s_2, \ldots, s_n) \mid s_i \in \mathbb{R}_0^{\omega_m} \ \forall \ i \in \mathbb{Z}_1^n \tag{10}$$

where s_1 is the *starting speed* and s_n the ending speed.

Recall our goal is to maximize the demand (and thus WCET) over some interval. With speed sequences, our goal is now to find the speed sequence which maximizes WCET over some interval. However, engines have minimum and maximum accelerations (α^- and α^+) meaning not every speed is *reachable* in one revolution from every other speed. To explain, we simplify the *reachable* definition in Bijinemula et al. [5]: s_b is **reachable** from s_a if $\sqrt{s_a^2 + 2\alpha^-} \leq s_b \leq \sqrt{s_a^2 + 2\alpha^+}$. By definition, a speed is reachable from itself.

Putting our original goal in context: since not every pair of speeds is reachable, not every speed sequence is *kinematically feasible* (i.e., not every sequence can be produced by an ICE without exceeding acceleration limits). If we let $\mathbf{S}$ be the set of all speed sequences,

[3] One minute is 6.0×10^7 μs

then $\mathbf{S_F} \subset \mathbf{S}$ is the set of all kinematically feasible sequences such that any two consecutive speeds in $S \in \mathbf{S_F}$ is reachable:

$$S \in \mathbf{S_F} \implies \sqrt{s_i^2 - 2\alpha^+} \leq s_{i+1} \leq \sqrt{s_i^2 + 2\alpha^+} \qquad \forall\, i \in \mathbb{Z}_1^{n-1} \quad (11)$$

Thus, we are only interested in finding a *feasible* sequence that maximizes demand over an interval. If we restrict our analysis to $\mathbf{S_F}$, then the MIAT of $S \in \mathbf{S_F}$ (as derived in previous works) is:

$$T(S) = \begin{cases} \sum_{x=1}^{|S|-1} \overline{T}(s_x, s_{x+1}) + \overline{T}(s_{|S|}, \hat{s}_{|S|}) & \text{if } |S| > 1 \\ \overline{T}(s_{|S|}, \hat{s}_{|S|}) & \text{if } |S| = 1 \\ 0 & \text{if } |S| = 0 \end{cases} \quad (12)$$

where $\hat{s}_{|S|} = \min(\omega_m, \sqrt{s_{|S|}^2 + 2\alpha^+})$ and the term $\overline{T}(s_{|S|}, \hat{s}_{|S|})$ gives the MIAT of the last job release while accelerating maximally.

To conclude this section, we restate the demand of $S \in \mathbf{S_F}$:

$$D(S) = \begin{cases} \sum_{s \in S} C(s) & \text{if } |S| \geq 1 \\ 0 & \text{if } |S| = 0. \end{cases} \quad (13)$$

In this work, we assume $D(S) \leq T(S)\ \forall\ S$. Practically, this means WCET values are such that utilization cannot exceed one - an assumption consistent with existing AVR task sets [5, 11, 44, 49]. We now describe the AVR task demand problem, the existing BPCKP-based approach, and present our solution.

2.4 The AVR Task Demand Problems

The AVR task demand problem covered in prior works is as follows:

- Given an AVR task, T_{avr}, with m modes and an interval, δ, find an algorithm, **E**, to calculate the exact maximum demand that can be generated over an interval of size δ.

In this work, we focus on the approximation variation:

- Given an AVR task, T_{avr}, with m modes and an interval, δ, find an *FPTAS*, **A**, *to approximate* the exact maximum demand that can be generated over an interval of size δ.

To be an FPTAS, **A**, given a fixed $\epsilon > 0$, must have a runtime polynomially bounded in the number of modes, m, and $\frac{1}{\epsilon}$ while producing demand at most $(1 - \epsilon)$ times the exact maximum demand.

2.5 AVR Task Demand as a BPCKP

Bijinemula et al. view AVR task demand as a BPCKP [5] since not all speeds are *reachable* from one another – creating precedence constraints among speeds. These constraints are represented as an out-tree with vertices as speeds and edges linking reachable speeds. Figure 5 illustrates one such tree. In the BPCKP approach, interarrival times are viewed as "weight", WCET as "profit", and the demand window δ as "capacity". The resulting formulation has $O(j \cdot M_\delta)$ decision variables where j is the number of unique speeds and M_δ the maximum number of jobs over interval δ. Per [5, 44], $j = O(m \cdot \frac{\omega_m^2 - \omega_0^2}{2\alpha^+})$ unique speeds must be considered. Furthermore, since utilization cannot exceed one and the smallest WCET $c_m = 1$, we know $M_\delta = O(\delta)$. Therefore, $O(m \cdot \frac{\omega_m^2 - \omega_0^2}{2\alpha^+} \cdot \delta)$ decision variables exist. This is dependent on *values* of kinematic parameters (ω_0, ω_m, and α^+) — making an FPTAS for this BPCKP impossible since an FPTAS must be polynomial in only the problem size (m) and $1/\epsilon$.

2.6 Solution Overview

To solve the approximation AVR task demand problem, we provide an exact dynamic programming solution dependent on m and δ. An FPTAS of the exact solution is then provided without pseudopolynomial dependence on δ using three components:

(1) a BPCKP formulation using predefined speed sequences that is polynomial in m and δ,
(2) an exact dynamic programming solution pseudopolynomial in m and δ, and
(3) a three-part approximation of the exact solution that is polynomial in m, δ, and $\frac{1}{\epsilon}$.

This gives the desired FPTAS polynomially bounded in m and $\frac{1}{\epsilon}$.

3 PREDEFINED SEQUENCES

To cover predefined speed sequences (PDSes), we differentiate the set and sequence notation. Unordered sets use conventional notation (e.g., $S = \{a, b, c\}$) and operators (e.g., $\cup$ and $\cap$). Sequences use parenthesis and a concatenation operator ($+\!\!+$) as follows:

Let a sequence be given by $S = (s_1, s_2, \ldots, s_n)$ or by iterative construction as in $S = (k)_{k=0}^{n} = (s_1 = 0, s_2 = 1, \ldots, s_{n+1} = n)$ Furthermore, let us define a concatenation operation, $+\!\!+$, as follows: $S_a +\!\!+ S_b = (a_1, a_2, b_1, b_2) \iff S_a = (a_1, a_2), S_b = (b_1, b_2)$ which also functions iteratively as: $+\!\!+_{x=1}^{n} S_x = S_1 +\!\!+ S_2 +\!\!+ \cdots +\!\!+ S_n$. With this notation, we now cover predefined speed sequences.

Recall that our goal is to find the speed sequence S which maximizes demand $D(S)$ over the interval δ (i.e., $T(S) \leq \delta$). Given an AVR task, T_{avr}, there are infinitely many sequences to consider [43]. Fortunately, previous works identify *dominant sequences* [5, 8, 10], a subset of $\mathbf{S_F}$ which provably maximize demand compared to peers. Thus, these are the only sequences that must be searched to find one maximizing demand. A dominant sequence S must:

(1) begin at a boundary speed (i.e., $s_1 \in S \subset \Omega$), **AND**
(2) use variable acceleration to reach a boundary speed **OR** use constant maximum acceleration to reach the next speed (i.e., $(s_{i+1} \in \Omega) \vee (s_{i+1} = \sqrt{s_i^2 + 2\alpha^+})$).

Hereafter, let $\mathbf{S_D} \subset \mathbf{S_F}$ be the set of all dominant sequences.

The BPCKP approach models dominant sequences as an out-tree - shown in Figure 5. Note for simplicity of presentation, any node in this tree representation may act as a terminal node as in [5]. Unfortunately, this representation makes the number of nodes pseudopolynomial in kinematic parameters (i.e., $O(m \cdot \frac{\omega_m^2 - \omega_0^2}{2\alpha^+} \cdot \delta)$ from earlier), preventing the formation of an FPTAS.

To avoid this, consider condensing the out-tree from Figure 5 into the recursive tree in Figure 6. Consecutive nodes with the same speed (e.g., ω_i) could be combined into a single node representing a desired number of repetitions. For example, let:

$$(\omega_i)_{\circlearrowleft}^{r} = \begin{cases} (\omega_i)_{k=1}^{r} & \text{if } r > 0 \\ \emptyset & \text{otherwise} \end{cases} \quad (14)$$

give a repeating sequence of ω_i, a *Repeating Boundary* (RB) sequence. We then condense repeating ω_i into a set of $(\omega_i)_{\circlearrowleft}^{r}$ nodes - one for each possible value of r.

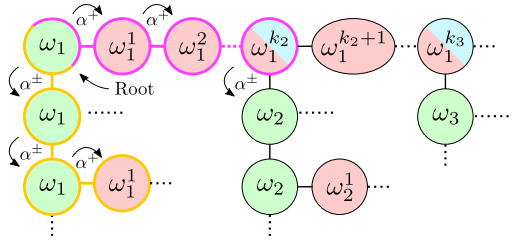

Figure 5: An example BPCKP out-tree adapted from [5] with root ω_1 where ω_x^y is the y^{th} speed reached via max acceleration (i.e., $\omega_x^y = \sqrt{\omega_x^2 + 2\alpha^+ y}$ where $\omega_x^0 = \omega_x$), k_z is defined such that ω_z is *reachable* from $\omega_x^{k_z}$ (e.g. a sequence accelerating from boundary ω_1 to ω_2 is $S = (\omega_1^0, \omega_1^1, \ldots, \omega_1^{k_2})$), and α^+ denotes max acceleration while $\alpha^\pm$ denotes variable acceleration.

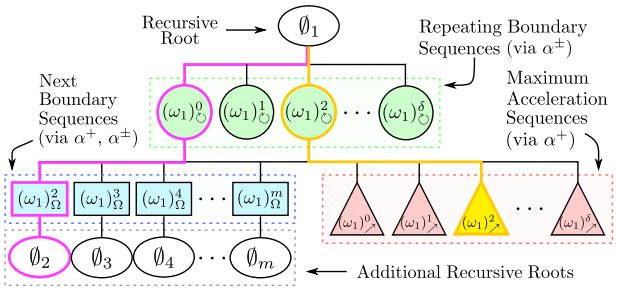

Figure 6: A condensed, recursive speed sequence tree

Similarly, consecutive nodes representing maximum acceleration could also be combined. We could let:

$$(\omega_i)^f_{\nearrow} = \begin{cases} \left(\sqrt{\omega_i^2 + 2\alpha^+(k-1)}\right)_{k=1}^{f} & \text{if } f \in \mathbb{Z}_1^{R(i,m)} \wedge i \in \mathbb{Z}_1^{m-1} \\ \emptyset & \text{otherwise} \end{cases} \tag{15}$$

give a speed sequence generated by accelerating maximally for f revolutions (i.e., job releases), a *Maximum Acceleration* (MA) sequence. Note that no speed is returned when $i = m$ as speeds cannot exceed ω_m and $(\omega_m)^r_{\circlearrowright}$ can produce any number of repeated ω_m speeds, if necessary. $R(i, j)$, defined below, gives the number of job releases required to accelerate maximally from ω_i to ω_j. Requiring $f \leq R(i, m)$ prevents sequences exceeding ω_m from being produced. The subtraction by one makes the sequences start with ω_i in the MA PDS definition. The release count is then:

$$R(i,j) = \begin{cases} \lceil \theta(\omega_i, \omega_j) \rceil & \text{if } 0 < i < j \leq m \\ 0 & \text{otherwise} \end{cases} \tag{16}$$

An MA sequence may also end such that another boundary speed is *reachable* from the last MA sequence speed (e.g., $\omega_i^{k_z}$ in Figure 5). We refer to this special case of MA sequence as a *Next Boundary* sequence in Figure 6 and represent it as:

$$(\omega_i)^z_{\Omega} = \begin{cases} (\omega_i)^{R(i,z)}_{\nearrow} & \text{if } i \in \mathbb{Z}_1^{m-1} \wedge z \in \mathbb{Z}_{i+1}^{m} \\ \emptyset & \text{otherwise .} \end{cases} \tag{17}$$

Finally, to simplify presentation we define a PDS combining RB and MA sequences (hereafter RB-MA sequences):

$$\Omega_i^{r,f} = (\omega_i)^r_{\circlearrowright} + (\omega_i)^f_{\nearrow} \tag{18}$$

Note that when $f = R(i, z)$ the MA sequence is equivalent to $(\omega_i)^z_{\Omega}$ per Equation 17. Thus, any dominant sequence producible by the BPCKP out-tree (Figure 5) is producible by the condensed, recursive out-tree (Figure 6) and representable as series of $\Omega_i^{r,f}$ PDSes.

To illustrate why, consider the following cases. The pink path in the BPCKP tree produces the sequence $S = (\omega_1, \omega_1^1, \ldots, \omega_1^{k_2})$. In the condensed, recursive tree, an equivalent path is highlighted in pink. Using the RB-MA definition, this equivalent sequence is: $\Omega_1^{0,R(1,2)} = (\omega_1)^0_{\circlearrowright} + (\omega_1)^{R(1,2)}_{\nearrow} = \emptyset + (\omega_1)^2_{\Omega} = (\omega_1, \omega_1^1, \ldots, \omega_1^{k_2})$. Note that although the pink path in the BPCKP tree stops at $\omega_1^{k_2}$, alternatively it could expand to include ω_2 as an RB sequence or may continue as an MA sequence. The same property is true for the recursive tree representation in which the recursive root $\emptyset_2$ is highlighted as the recursive roots alow for another RB-MA sequence beginning at ω_2 to follow. This mapping of the $\omega_i^{k_z}$ speeds to recursive roots allows for the concatenation of multiple RB-MA sequences in the recursive tree just as the original BPCKP tree representation allows sequences to arrive at and then proceed from new boundary speeds. This concatenation uses the variable acceleration referenced in Figure 5 and visualized in Figures 3 and 4 thus allowing the condensed tree to incorporate variable acceleration.

The gold path in the BPCKP tree produces the sequence $S = (\omega_1, \omega_1, \omega_1^0, \omega_1^1)$. In the condensed, recursive tree, an equivalent path is highlighted in gold as well. This equivalent sequence can be described with the RB-MA definition as: $\Omega_1^{2,2} = (\omega_1)^2_{\circlearrowright} + (\omega_1)^2_{\nearrow} = (\omega_1, \omega_1, \omega_1^0, \omega_1^1)$. Unlike the pink path, this sequence terminates at ω_1^1. This is also true in the recursive tree where the sequence ends at a terminal leaf (instead of a recursive root).

The demand and MIAT of $\Omega_i^{r,f}$ is then $D(\Omega_i^{r,f})$ and $T(\Omega_i^{r,f})$, respectively. With this condensed representation, we now create the PDS-based BPCKP.

4 AVR TASK DEMAND WITH PDSES

A new AVR Task Demand BPCKP is now presented where decision variables represent PDSes and a solution gives our desired result: the maximum demand over an interval δ. The function $B(\mathbf{x})$, defined later, constructs a speed sequence producing maximum demand.

Let $G_P = (V_P, A_P)$ be a precedence constraint graph where V_P represents the set of all $\Omega_i^{r,f}$ for all possible combinations of i, r, f as vertices and A_P represents the set of all PDS pairs that form valid, dominant sequences as edges. That is, let $((i, r_i, f_i), (j, r_j, f_j)) \in A_P$

indicate $S = (\Omega_i^{r_i,f_i} +\!\!+ \Omega_j^{r_j,f_j}) \in \mathbf{S_D}$. The new BPCKP is then:

$$\underset{x}{\text{maximize}} \quad \sum_{i=1}^{m}\sum_{r=0}^{R_\delta}\sum_{f=0}^{F_\delta} D(\Omega_i^{r,f}) \cdot x_i^{r,f} \tag{19}$$

$$\text{subject to} \quad T(B(\mathbf{x})) \le \delta \tag{20}$$

$$\sum_{r=0}^{R_\delta}\sum_{f=0}^{F_\delta} x_i^{r,f} \le 1 \qquad \forall\, i \in \mathbb{Z}_1^m \tag{21}$$

$$((i,r_i,f_i),(j,r_j,f_j)) \in A_P$$

$$\forall\, x_i^{r_i,f_i} = 1 \mid x_j^{r_j,f_j} = N(x_i^{r_i,f_i}). \tag{22}$$

where $x_i^{r,f}$ indicates whether $\Omega_i^{r,f}$ is in the solution sequence, $B(\mathbf{x})$, defined later, concatenates selected PDSes into one, contiguous speed sequence, and $N(x_i^{r,f})$ gives the next selected PDS in order of index i. Formally, $N(x_i^{r,f}) = x_j^{r_j,f_j} \mid j > i \wedge x_j^{r_j,f_j} = 1 \wedge \nexists k \mid i < k < j \wedge x_k^{r_k,f_k} = 1$.

The objective function, Equation 19, specifies maximal demand where the terms R_δ and F_δ refer to the maximum RB and MA sequence length in time δ, respectively. Since δ and $\mathbf{C}$ are expressed in microseconds (μs) and the smallest WCET is c_m, the maximum number of job releases that fit in any interval of size δ is $\lfloor \delta / c_m \rfloor$. Thus, $R_\delta = F_\delta = \lfloor \delta / c_m \rfloor$. Since the smallest c_m value is one, $R_\delta = F_\delta = \delta$ is a safe upper bound [4]. Equation 20 requires all jobs have deadlines within δ. Equation 21 requires at most one PDS per mode to enforce condensing repeating boundaries (e.g., a solution having $x_1^{2,0} = 1$ and $x_1^{4,0} = 1$ is equivalent to $x_1^{6,0} = 1$ since $\Omega_1^{2,0} +\!\!+ \Omega_1^{4,0} = \Omega_1^{6,0}$). Equation 22 requires solutions to honor precedence constraints from graph G_P (i.e., the solution is a dominant sequence). Specifically, the $N(x_i^{r,f})$ term forces selected PDSes, concatenated in order of index, to honor precedence constraints of A_P.

To construct the solution, let $\mathbf{x}$ be the array $x_i^{r,f}$. The sequence formed from the $\mathbf{x}$ is:

$$B(\mathbf{x}) = \mathop{+\!\!+}_{i=1}^{m}\mathop{+\!\!+}_{r=0}^{\delta}\mathop{+\!\!+}_{f=0}^{\delta} b(x_i^{r,f}) \mid b(x_i^{r,f}) = \begin{cases} \Omega_i^{r,f} & \text{if } x_i^{r,f} = 1 \\ \emptyset & \text{otherwise.} \end{cases} \tag{23}$$

Since this formulation uses PDSes represented by $\Omega_i^{r,f}$, the number of unique PDSes (i.e., vertices of G_P) is $O(m\delta^2)$ as i is at most m while r and f are at most δ. We now have a BPCKP independent of kinematic terms (e.g., ω_0, α^+, or ω_m) — a foundation for the FPTAS. Moreover, this formulation does not require pre-computation of PDSes - simply iteration through i, r, f values (as opposed to individual speeds as in [5]).

5 AN EXACT DEMAND CALCULATION

This section begins with notation for sets of PDSes. We then establish the AVR task demand calculation's optimal substructure. An exact, recursive dynamic programming solution with pseudo-polynomial runtime is then introduced.

[4] A large δ greatly increases problem size. This is true for prior work [5] (as evident by the experiments section) and a necessary weakness for a kinematic-agnostic BPCKP. Practical limitations (e.g., $c_m \gg 1$) make execution tractable without approximation.

5.1 PDS Subset Notation

To discuss optimal substructure, we introduce restricted set notation. Let $\mathbf{S_{\Omega,i}}$ be the set of PDSes which begin with speed ω_i (i.e., $\mathbf{S_{\Omega,i}} \subset \mathbf{S_D} \mid \forall S \in \mathbf{S_{\Omega,i}}, s_1 = \omega_i$), $\mathbf{S_\Omega^b}$ be the set of PDSes which produce at least b demand ($\mathbf{S_\Omega^b} \subset \mathbf{S_D} \mid \forall S \in \mathbf{S_\Omega^b}, D(S) \ge b$), and $\mathbf{S_{\Omega,i}^b}$ be the set of PDSes which begin ω_i *and* produce at least b demand ($\mathbf{S_{\Omega,i}^b} \subset (\mathbf{S_{\Omega,i}} \cap \mathbf{S_\Omega^b}) \mid \forall S \in \mathbf{S_{\Omega,i}^b}, S \in (\mathbf{S_{\Omega,i}} \cap \mathbf{S_\Omega^b})$).

5.2 Proof of Optimal Substructure

To provide a foundation for the dynamic programming solution, proof of the demand calculation's optimal substructure is presented. In the following proof, $S_{i,b}^*$ denotes an optimal speed sequence (i.e., $S_{i,b}^*$ gives the MIAT of all PDSes beginning with ω_i and producing demand at least b). More formally, $S_{i,b}^* \in \mathbf{S_{\Omega,i}^b} \mid T(S_{i,b}^*) \le T(S')\ \forall S' \in \mathbf{S_{\Omega,i}^b}$). Additionally, the symbol $\rightsquigarrow$ denotes any additional PDSes that may or may not be present (i.e., the trailing end of a sequence not relevant to the proof).

Lemma 1 (AVR Task Demand Optimal Substructure). *Given an AVR task T_{avr} with m modes and boundary speeds $\mathbf{\Omega}$, if $S_{i,b}^* = (\Omega_i^{r_i,R(i,x)} +\!\!+ \Omega_x^{r_x,f_x} \rightsquigarrow)$, then $(\Omega_x^{r_x,f_x} \rightsquigarrow)$ is necessarily optimal (i.e., $(\Omega_x^{r_x,f_x} \rightsquigarrow) = S_{x,b_r}^*$ where $b_r = b - D(\Omega_i^{r_i,R(i,x)})$).*

Proof. Suppose $S_{i,b}^* = (\Omega_i^{r_i,R(i,x)} +\!\!+ \Omega_x^{r_x,f_x} \rightsquigarrow)$. We claim the subsequence $(\Omega_x^{r_x,f_x} \rightsquigarrow)$ is also optimal (i.e., $(\Omega_x^{r_x,f_x} \rightsquigarrow) = S_{x,b_r}^*$ where $b_r = b - D(\Omega_i^{r_i,R(i,x)})$). To prove so, suppose $\exists S' = \Omega_{x'}^{r_{x'},f_{x'}}$ such that $D(S') \ge b - D(\Omega_i^{r_i,R(i,x')}) \wedge T(\Omega_i^{r_i,R(i,x')} +\!\!+ S') < T(S_{i,b}^*)$. We have a contradiction, since either $S_{i,b}^*$ is optimal or $\Omega_i^{r_i,R(i,x')} +\!\!+ S'$ is optimal. Thus, there is an optimal substructure in which $S_{i,b}^*$ may be constructed with S_{x,b_r}^* where $x > i$. □

By this lemma, an optimal speed sequence having MIAT with demand at least b may be constructed only by considering all possible subsequent PDSes beginning with indices from $i+1$ to m as part of the sequence $S_{i,b}^*$. That is, to produce $S_{i,b}^*$ requires solving S_{x,b_r}^* where $b_r = b - D(\Omega_i^{r_i,R(i,x)})$ for all $x \in \mathbb{Z}_{i+1}^m$. Using this strategy, the dynamic programming solution is presented.

5.3 Dynamic Programming Solution

To make the dynamic programming solution, let $T(i,b)$ be:

$$T(i,b) = \min_{S \in S(i,b)} T(S) \quad | \quad S(i,b) = \bigcup_{x=1}^{i} \mathbf{S_{\Omega,x}^b} \tag{24}$$

where $T(i,b)$ gives the MIAT of all sequences in $S(i,b)$. The function combines sets of $\mathbf{S_{\Omega,x}^b}$, sequences which begin with ω_i and produce demand at least b, by iterating through all boundary speeds. Iterating through all boundary speeds starting with ω_i tests each boundary as a possible starting point when generating demand b. With all sets aggregated, $T(i,b)$ finds the sequence with MIAT and demand at least b.

We now recursively define $\mathbf{S}^{\mathbf{b}}_{\Omega,\mathbf{i}}$ for dynamic programming:

$$\mathbf{S}^{\mathbf{b}}_{\Omega,\mathbf{i}} = \begin{cases} \emptyset & \text{if } b \le 0 \\ \Omega_m^{r,0} \mid r = \lceil \frac{b}{c_i} \rceil & \text{if } b > 0 \wedge i = m \\ \left\{ \bigcup_{f=0}^{\lceil b/c_m \rceil} \left(\Omega_i^{r_d,f} \right) \mid r_d = b - D(\Omega_i^{0,f}) \right\} & \text{if } b > 0 \wedge i < m \\ \bigcup \left\{ \bigcup_{r=0}^{\lceil b/c_i \rceil} \bigcup_{j=i+1}^{m} \bigcup_{S \in \mathbf{S}^{\mathbf{br}}_{\Omega,\mathbf{j}}} \left(\Omega_i^{r,R(i,j)} + S \right) \right\} & \end{cases}$$

where $b_r = b - D(\Omega_i^{r,R(i,j)})$. The cases are as follows:

In the first case, requested demand $b \le 0$ so no sequence is returned since $D(\emptyset) = 0 \ge b$.

In the second case, requested demand $b > 0$ and $i = m$. Since $i = m$, no PDS $\Omega_j^{r_j,f_j}$ with $j > m$ will be concatenated afterward, and no subsequent PDSes are considered. Moreover, $(\omega_m)^{f}_{\nearrow} = \emptyset \; \forall f$ so no f values other than $f = 0$ are needed. This leaves only a value of r to produce $D(S) \ge b$. Thus, $\lceil b/c_i \rceil$ gives the minimum r to produce demand b.

In the third case, requested demand $b > 0$, and the sequence must start at a boundary speed less than ω_m (i.e., $i < m$). In this case, the terms before the big union ($\bigcup$) iterate through all combinations of f and r_d in which the single PDS $\Omega_i^{r_d,f}$ produces at least demand b (e.g., the right side of the condensed, recursive tree in Figure 6). Note, the term r_d is selected to make up the difference between the requested demand b and the demand produced by the f term alone (i.e., $D(\Omega_i^{0,f})$).

The terms after the union ($\bigcup$) iterate through all combinations of r *and* possible subsequent PDSes (e.g., the left side of the condensed, recursive tree in Figure 6). The j term iterates through all boundaries from which subsequent PDSes must be considered. Replacing f with $R(i,j)$ ensures that $\Omega_i^{r,R(i,j)}$ produces *only* sequences which are feasible when concatenated with a subsequent PDS starting at ω_j. Lastly, the union with term $S \in \mathbf{S}^{\mathbf{br}}_{\Omega,\mathbf{j}}$ iterates through every possible dominant sequence beginning with ω_j and generating at least demand $b_r = b - D(\Omega_i^{r_i,R(i,j)})$ (i.e., the remaining demand to generate after the demand from $\Omega_i^{r_i,R(i,j)}$ is subtracted from the total requested demand). This term implements the dynamic programming property.

Informally, $\mathbf{S}^{\mathbf{b}}_{\Omega,\mathbf{i}}$ enumerates all dominant sequences starting with ω_i producing demand at least b. $S(i,b)$ aggregates $\mathbf{S}^{\mathbf{b}}_{\Omega,\mathbf{1}}$ through $\mathbf{S}^{\mathbf{b}}_{\Omega,\mathbf{i}}$. Finally, $T(i,b)$ finds the sequence in $S(i,b)$ with MIAT, giving the MIAT to generate demand b.

We now cover the dynamic programming algorithm for exact AVR task demand which we call Algorithm **E**, given below:

The overall runtime is $O(\lg \delta \cdot m \cdot \delta^2)$ as follows. The primary loop in Algorithm 1 is $O(\delta)$. However, this may be improved with a binary search to build results in the MIAT[] table making the primary loop runtime $O(\lg \delta)$. Inside this loop, each call to $T(m,b)$ is a single call to $S(m,b)$ (Equation 24). This single call to $S(m,b)$ uses $\mathbf{S}^{\mathbf{b}}_{\Omega,\mathbf{i}}$ to calculate $\Omega_i^{r,f}$ for every combination of i, r, and f. With memoization, Algorithm **E** only needs to compute $\Omega_i^{r,f}$ once for every i, r, f combination. Given that i is $O(m)$ while r and f are $O(b)$, we know $S(m,b)$ (and thus $T(m,b)$) has runtime $O(mb^2)$.

Algorithm 1 Exact DP $T(i,b)$ (Algorithm **E**)

```
function AVR-Demand-Exact(T_avr, δ):
    MIAT[0, 1, ..., δ] ← ∞                    ▷ Init. MIAT table
    for b ← 0 to δ do:                        ▷ Iterate through all demands
        MIAT[b] ← T(m, b)                     ▷ Save MIAT
    end for
    return max{b | MIAT[b] ≤ δ}               ▷ Return max demand
end function
```

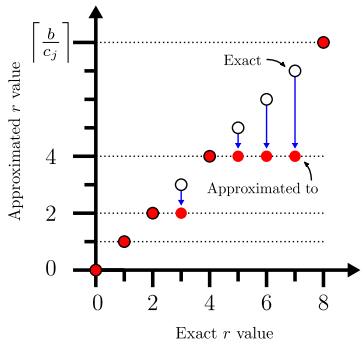

Figure 7: Example Approximation of ω_j Repetition Quantity

Recalling that b is at most δ, we now have a primary loop with runtime $O(\lg \delta)$ and loop contents, $T(m,b)$, with runtime $O(m\delta^2)$. This gives the overall runtime: $O(\lg \delta \cdot m \cdot \delta^2)$.

6 DEMAND APPROXIMATIONS

Algorithm **E**'s runtime is pseudopolynomial in δ. To build an FPTAS, we must shed the pseudopolynomial dependence on δ (e.g, by replacing it with $\ln \delta$ which is polynomial in the size of the input δ when represented in bits). This requires changing the implementation of $\mathbf{S}^{\mathbf{b}}_{\Omega,\mathbf{i}}$ - the function contributing δ^2 to overall runtime.

This section covers the approximation of δ for r and f and an accompanying Algorithm **A** — the FPTAS. An approximation ratio proof is then presented along with a definition of *safe demand*, D_{safe} — the demand a user may safely rely on after approximation.

6.1 RB Sequence Approximation

The dynamic programming function $\mathbf{S}^{\mathbf{b}}_{\Omega,\mathbf{i}}$ relies on b to quantify the maximum repeated jobs at boundary ω_i. Since b is at most δ, this means δ unique values of r. To facilitate an FPTAS, let $\epsilon_r > 0$ be the basis for a log scale representation of r at a given ω_i. Specifically, let the set of values for r instead be:

$$r'(b,i,\epsilon_r) = \left\{ \left\lfloor \left\lceil \frac{b}{c_i} \right\rceil \cdot (1-\epsilon_r)^k \right\rfloor \qquad \forall k \in \mathbb{Z}_0^{\lfloor \ell_r \rfloor + 1} \right\} \tag{25}$$

where ℓ_r is the value of k at which $\left\lceil \frac{b}{c_i} \right\rceil \cdot (1-\epsilon_r)^k = 1$:

$$\ell_r = \frac{\ln(1) - \ln\left\lceil \frac{b}{c_i} \right\rceil}{\ln(1-\epsilon_r)} = -\frac{\ln\left\lceil \frac{b}{c_i} \right\rceil}{\ln(1-\epsilon_r)} < \frac{\ln b}{\epsilon_r} \tag{26}$$

Figure 7 illustrates this approximation. This approach creates a smaller set of searched repetitions at each boundary, bounded polynomially in the problem size as there are at most $\frac{\ln \delta}{\epsilon_r}$ unique values

of r to test when $b = \delta$, as opposed to $\left\lceil \frac{\delta}{c_i} \right\rceil$. To explain the effect on MIAT calculation by $T(i, b)$, consider the following lemma.

Lemma 2. *If q is the repetition quantity r selected at any boundary by $S(i, b)$, when approximating the set of repetitions with ϵ_r, then $q \geq (1 - \epsilon_r) \cdot OPT_r$.*

Proof. Let OPT_r be the optimal number of repetitions at some boundary ω_i and q be the approximated number of repetitions. Furthermore, let OPT_r be defined such that $\left\lceil \frac{b}{c_i} \right\rceil \cdot (1 - \epsilon_r)^{x+1} \leq OPT_r \leq \left\lceil \frac{b}{c_i} \right\rceil \cdot (1 - \epsilon_r)^x$ (i.e., OPT_r is between two repetition values of the approximated version). If index $x + 1$ is selected (i.e., $q = \left\lceil \frac{b}{c_i} \right\rceil \cdot (1 - \epsilon_r)^x$), OPT_r is at most $\left\lceil \frac{b}{c_i} \right\rceil \cdot (1 - \epsilon_r)^x$. The maximum underestimate, therefore, is given by:

$$\frac{\left(\left\lceil \frac{b}{c_i} \right\rceil \cdot (1 - \epsilon_r)^{x+1}\right)}{\left(\left\lceil \frac{b}{c_i} \right\rceil \cdot (1 - \epsilon_r)^x\right)} = (1 - \epsilon_r) \Rightarrow q \geq (1 - \epsilon_r) \cdot OPT_r$$

The lemma is proven. □

A similar approximation, covered below, may be applied to the set of values for f to limit the number of MA sequences to search.

6.2 MA Sequence Approximation

In Equation 25, f is used to search all possible consecutive release quantities via $(\omega_i)^{\nearrow f}$. Like r, f is bounded by b (i.e., there are b unique values of f). Let $\epsilon_f > 0$ be the basis for a log-scale representation of f. Then, the set of approximated values for f is:

$$f'(b, \epsilon_f) = \left\{ \left\lfloor \left\lceil \frac{b}{c_m} \right\rceil \cdot (1 - \epsilon_f)^k \right\rfloor \;\; \forall\, k \in \mathbb{Z}_0^{\lfloor \ell_f \rfloor + 1} \right\} \tag{27}$$

where ℓ_f is the value of k at which $\left\lceil \frac{b}{c_m} \right\rceil \cdot (1 - \epsilon_f)^k = 1$:

$$\ell_f = -\frac{\ln \left\lceil \frac{b}{c_m} \right\rceil}{\ln(1 - \epsilon_f)} < \frac{\ln b}{\epsilon_f} \tag{28}$$

We then establish a similar lemma for approximation of f. The proof is identical in form to Lemma 2 and omitted for space.

Lemma 3. *If q is the number of consecutive job releases at maximum acceleration at any boundary by $S(i, b)$ when approximating consecutive releases with ϵ_f, then $q \geq (1 - \epsilon_f) \cdot OPT_f$.*

The terms r and f, which individually contribute $O(b)$ time in the exact case, are now $O(\ln b/\epsilon_r)$ and $O(\ln b/\epsilon_f)$ respectively. The final approximation below will remove the MIAT table size dependence on δ shown in Algorithm **E**.

6.3 Approximation of MIAT Table Size

Recall that in Algorithm **E**, $T(m, b)$ must calculated for $b \in \mathbb{Z}_0^{\delta}$ (ignoring binary search). Since demand is at most $b = \delta$, this approximation is required to remove dependence on δ. To remove the dependency, we propose scaling PDS WCET (and thus the maximum demand we must search) via a scaling factor K and scaling function $D_K(S)$ — just as the 0-1 knapsack utilizes a scaling factor [54]. Let a scaling function and factor, K, be defined as:

$$D_K(S) = \left\lfloor \frac{D(S)}{K} \right\rfloor \mid K = \frac{\epsilon_b \cdot b}{m} \tag{29}$$

where $\epsilon_b > 0$ and $D_K(S)$ is a scaled version of $D(S)$ which scales demand by K for use in the approximation of Algorithm **E**. Note ϵ_b scales the maximum value of b (i.e., δ) that must be searched. The maximum MIAT table size is now $b' = \left\lfloor \frac{b}{K} \right\rfloor = \frac{m}{\epsilon_b}$. We now incorporate all three approximations of r, f, and b as:

$$\mathbf{S'^{b}_{\Omega,i}} = \begin{cases} \emptyset & \text{if } b \leq 0 \\ \bigcup\limits_{r \in r'} \left(\Omega_i^{r,0} \right) \mid D_K(\Omega_i^{r,f}) \geq b & \text{if } b > 0 \wedge i = m \\ \left\{ \bigcup\limits_{r \in r'} \bigcup\limits_{f \in f'} \left(\Omega_i^{r,f} \right) \mid D_K(\Omega_i^{r,f}) \geq b \right\} & \text{if } b > 0 \wedge i < m \\ \bigcup \left\{ \bigcup\limits_{r \in r'} \bigcup\limits_{j=i+1}^{m} \bigcup\limits_{S \in \mathbf{S'^{b_r}_{\Omega,j}}} \left(\Omega_i^{r,R(i,j)} + S \right) \right\} & \end{cases}$$

where $r' = r'(b, i, \epsilon_r)$, $f' = f'(b, i, \epsilon_f)$, and $b_r = b - D_K(\Omega_i^{r,R(i,j)})$. We then use the apostrophe to denote approximated function variants as: $T'(m, b) = \min_{S \in S'(i,b)} T(S)$, and $S'(i, b) = \bigcup_{i=1}^{m} \mathbf{S'^{b}_{\Omega,i}}$. The approximation, Algorithm **A** is then Algorithm **E** with $T(m, b)$ replaced by $T'(m, b')$. We now prove Algorithm **A**'s results are within $(1 - \epsilon_b)$ of optimal.

Lemma 4. *If S gives MIAT $T'(m, b)$ and O gives MIAT $T(m, b)$, $S \geq (1 - \epsilon_b) \cdot OPT \mid OPT = T(m, b)$*

Proof. Let us define $S \mid T(S) = T'(m, b)$ and $O \mid T(O) = T(m, b)$. We assert:

$D(S) \geq K \cdot D'(S)$ ▶ By Equation 29

$D'(S) \geq D'(O) \implies K \cdot D'(S) \geq K \cdot D'(O)$ ▶ By construction

$D(O) - KD'(O) \leq mK$

$\implies K \cdot D'(O) \geq D(O) - mK$ ▶ By scaling at most m times

Combining the above inequalities gives:

$D(S) \geq K \cdot D'(S) \geq K \cdot D'(O) \geq D(O) - m \cdot K$ ▶ By above

$\Leftrightarrow D(S) \geq D(O) - \epsilon_b \cdot \delta$ ▶ By $K = \frac{\epsilon_b \cdot \delta}{m}$

$\Leftrightarrow D(S) \geq OPT - \epsilon_b \cdot OPT = (1 - \epsilon_b)OPT$ ▶ By $\delta \geq OPT$

where $D(O) = OPT$. Thus, the lemma is proven. □

6.4 Overall Approximation Ratio

Since the approximation of MIAT table size (i.e., $b = \delta$) applies to the sequences *after* approximating quantities r and f for $\Omega_i^{r,f}$, the overall approximation is given by:

$$(1 - \epsilon) \leq (1 - \epsilon_r)(1 - \epsilon_f)(1 - \epsilon_b) \tag{30}$$

where ϵ is the overall approximation value, ϵ_r is the approximation factor for unique values of r, ϵ_f is the approximation factor for unique values of r, and ϵ_b is the approximation factor for MIAT table size (i.e., for $b = \delta$). Since $(1 - \epsilon_r)$, $(1 - \epsilon_f)$, $(1 - \epsilon_b)$ must all exceed $(1 - \epsilon)$, a simple solution for selecting values of ϵ_r, ϵ_f, and ϵ_b given a desired overall ϵ is: $\epsilon_r = \epsilon_f = \epsilon_b = \epsilon/3$.

6.5 Approximation Runtime

With bounded search quantities for r and f, we derive a runtime for Algorithm **A**. The worst-case runtime of the exact approach, Algorithm **E**, is $O(\lg \delta \cdot m\delta^2)$. The approximation of r gives a bound of $O(\frac{\ln b}{\epsilon_r})$ and approximation of f gives a bound of $O(\frac{\ln b}{\epsilon_f})$. The approximation of b, gives $\lfloor \delta/K \rfloor = \lfloor m/\epsilon_b \rfloor = O(m/\epsilon_b)$. Since $b = O(\delta)$, the runtime is: $O\left(\lg(m/\epsilon_b) \cdot m \cdot (\ln \delta/\epsilon_r) \cdot (\ln \delta/\epsilon_f)\right)$.

6.6 Deriving a Safe Demand

Approximated demand is only guaranteed to exceed $(1-\epsilon) \cdot OPT$. However, it may be lower than the exact maximum demand (i.e., unsafe). To convert approximated demand to a pessimistic (i.e., safe) demand for use in real-time systems, users must instead use:

$$D_{\text{safe}}(\delta) = \frac{K \cdot b}{(1-\epsilon)} \mid b = \max\{\, b \mid T(m,b) \leq \delta\} \tag{31}$$

as the safe demand in a window of size δ where K is the scaling factor defined in Equation 29, and b is the largest value for which $T(m,b)$ has a MIAT less than δ.

7 EVALUATION AND RESULTS

In the absence of readily available ICEs with compatible, open-source engine control units implementing AVR tasks, all experiments are simulations as with prior works [4, 5, 44]. The source code and publication data are available online [55].

7.1 Setup and Experiments

7.1.1 Setup. The Bijinemula et al. artifact [4] (KAVR), our nearest peer, is used for comparison. The exact dynamic programming (EXACT) and FPTAS implementations (APX) are compared with KAVR. Linux on an AMD 74f3 3.2GHz CPU and 254 GB RAM is used for single-threaded simulation. Python 3.10.12 is used for consistency with the KAVR artifact. The canonical (CAN) and generalized (GEN) task sets are from literature [5, 11, 44]. APX parameters default to $\epsilon_b = \epsilon_r = \epsilon_f = 0.025$ with $(1-\epsilon) = 0.927$ and solutions overestimate demand at most 7.8% (1/0.927) per Equation 30. APX and EXACT use precision 12 for all experiments.

7.1.2 Experiment 1 - Variable Precision. In this experiment, the CAN and GEN 6-mode task sets from the literature are used. with simulation parameters identical to [5]. The results in Figure 8 highlight a KAVR weakness: increasing precision increases runtime. APX dominates runtime for all precisions, EXACT dominates when KAVR precision exceeds *five*. Minimum and maximum runtime improvement of APX to KAVR is: 6x and 7,761x respectively.

7.1.3 Experiment 2 - Varied Demand Window Sizes. In this experiment, the literature task sets are used with varied values for δ. Figure 9 shows the effect of δ on runtime where "-PXX" specifies KAVR precision XX. Note the log scale on both axes. Around $\delta = 1.3 \times 10^6 \mu s$, APX begins to outperform KAVR. At $\delta = 10e7$, APX is more than 6,100x faster. This is expected as KAVR is more sensitive to δ than APX given their asymptotic analysis.

Figure 10 illustrates RAM usage for the varied duration experiment versus precision. APX requires 99.99% less RAM than KAVR which also comports with the KAVR sensitivity to δ.

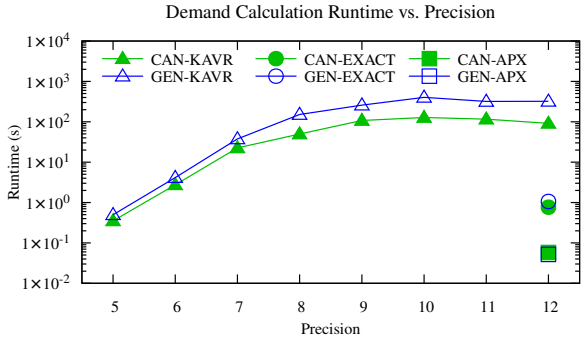

Figure 8: Variable Precision - Runtime

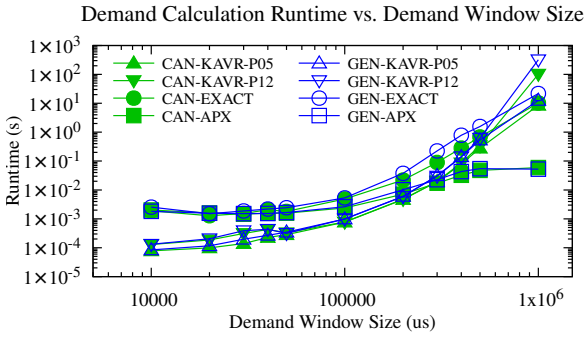

Figure 9: Variable Duration - Runtime

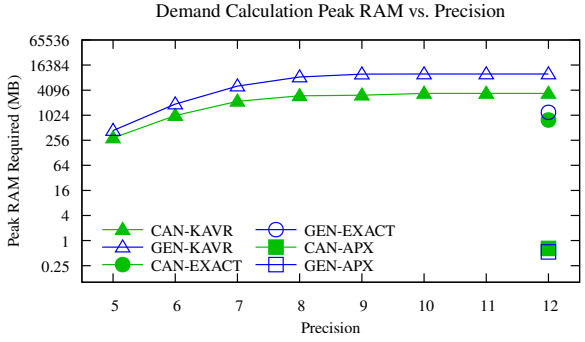

Figure 10: Variable Duration - Peak RAM Usage

7.1.4 Experiment 3 - Varied Acceleration. In this experiment, literature task sets are used with varied α^+ and α^- illustrating the effect of acceleration on runtime. Acceleration is varied in the range $[1\times10^4, 1\times10^6]$ rpm^2 with results in Figure 11. Note that low acceleration prevents speed sequences from reaching subsequent boundary speeds, thereby reducing search complexity (and runtime). At larger acceleration, feasible boundary speed combinations increase (increasing runtime). APX runtime dominates for both literature task sets. The minimum and maximum runtime improvement of APX over KAVR is 3.91x and 7,827x respectively.

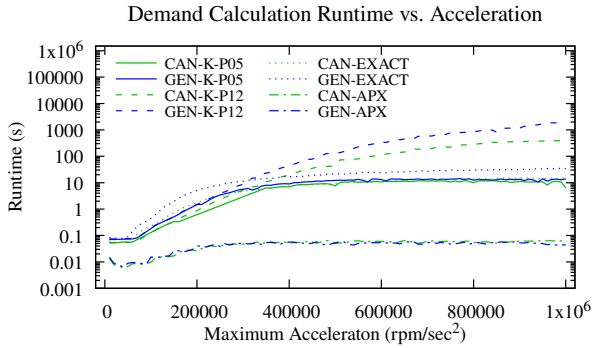

Figure 11: Varied Acceleration - Runtime

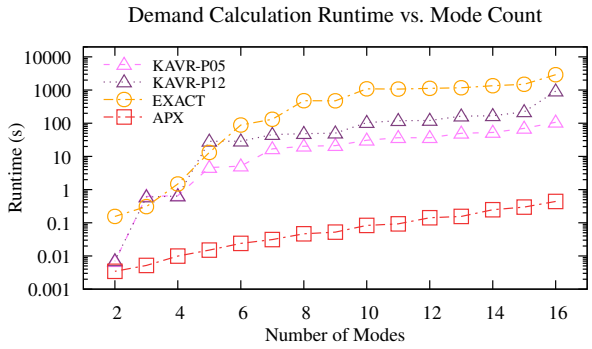

Figure 12: Varied Mode Count - Runtime

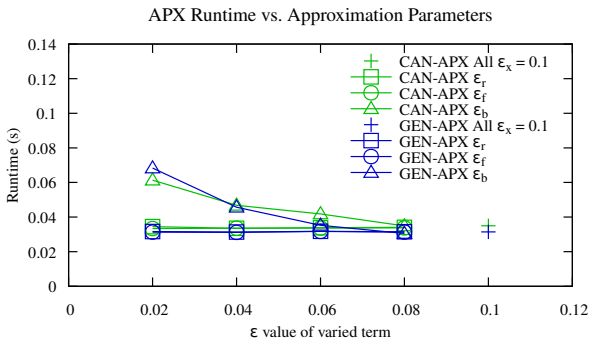

Figure 13: Varied Approximation Ratios - Runtime

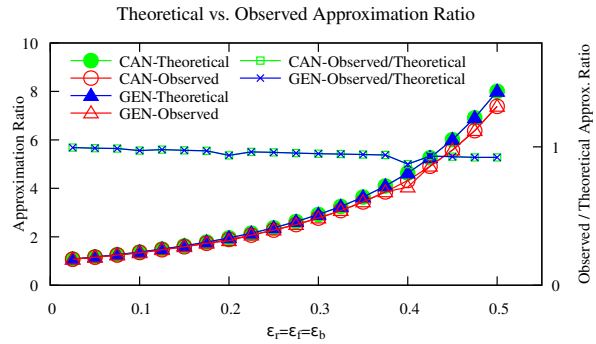

Figure 14: Solution Quality

7.1.5 Experiment 4: Varied Mode Count. In this experiment, a randomly generated AVR task with 16 modes was created. From the 16-mode task, 14 other AVR tasks are created by repeatedly merging the middle-most modes. This merging maintains the minimum speed, maximum speeds, $\alpha^+ = 7.5e5$ rpm^2, and $\delta = 750$ms. Thus, only mode quantity (m) changes. Figure 12 shows the results. APX runtime dominates especially for larger m. The minimum and maximum improvement over KAVR is 1.955x and 2,888x respectively.

An observant reader may ask: "*why does KAVR outperform EXACT?*" and given that KAVR *does* outperform EXACT, "*why does APX outperform KAVR?*". Recall KAVR has $O(m \cdot \frac{\omega_m^2 - \omega_0^2}{2\alpha^+} \cdot \delta)$ decision variables where EXACT has $O(\ln \delta \cdot m \cdot \delta^2)$. Using the CAN task set, $m = 6, \omega_m = 6500, \omega_0 = 500, \alpha^+ = 6.0e5$. Thus, KAVR is $O(35m\delta)$ and EXACT is $O(\ln \delta \cdot m \cdot \delta^2)$. For a large δ and m (e.g., $m = 6, \delta \geq 10.4$us), KAVR outperforms. Figures 9 and 12 illustrate this possibility. Moreover, the APX runtime with all $\epsilon_r = \epsilon_f = \epsilon_b = 0.025$, is: $O\left(\ln(m/0.025) \cdot m \cdot (2 \ln \delta / 0.025)\right)$. For a large m and δ (e.g., $m = 6, \delta \geq 4.073e4$us), APX outperforms. Practically, implementation alters exact m and δ required to outperform.

7.1.6 Experiment 5 - Varied Approximation Ratios. In this experiment, CAN and GEN task sets are used with varied values of ϵ_r, ϵ_f, and ϵ_b for APX. The results in Figure 13 illustrate effects of ϵ_r, ϵ_f, and ϵ_b on runtime. The crosshairs with labels "All $\epsilon_x = 0.1$" indicate reference runtime values where all parameters are 0.1 (i.e., $\epsilon_r = \epsilon_f = \epsilon_b = 0.1$). The graph plots each epsilon value varying from the baseline value 0.01 while others remain constant (e.g., $\epsilon_r = [0.02, 0.08]$ while $\epsilon_f = \epsilon_b = 0.1$). While all three contribute equally to the approximation ratio, increases in ϵ_b yield greater runtime reduction. This suggests users seeking faster runtimes should increase ϵ_b to maximize runtime improvement per increase in demand overestimate. Results also suggest ϵ_r and ϵ_f may be decreased with little runtime penalty to reduce maximum overestimate.

7.1.7 Experiment 6 - Solution Quality. To illustrate the ratio of observed to theoretical approximation ratio, the CAN and GEN literature task sets are used for demand calculation with a fixed $\delta = 1.0$s and varying values of equivalent $\epsilon_r = \epsilon_f = \epsilon_b$. Figure 14 compares the theoretical upper bound on demand overestimation (i.e., $1/(1-\epsilon)$) to the observed overestimate (i.e., APX D_{safe} / KAVR demand). The observed overestimate closely tracks the theoretical upper bound with a minimum ratio of 0.91 at $\epsilon_r = \epsilon_f = \epsilon_b = 0.5$ and a maximum of 0.99 at $\epsilon_r = \epsilon_f = \epsilon_b = 0.025$.

8 DISCUSSION AND LIMITATIONS

8.1 Effects of K on Runtime, Demand

Unlike the 0-1 knapsack problem, the scaling factor K is not a value we *directly* control. Instead, we control it via ϵ_b. As shown in Figure 13, the term ϵ_b has the greatest effect on runtime relative to ϵ_r and ϵ_f. Per Equation 29, we know a larger ϵ_b yields a larger K. This comports with the varied approximation ratio experiment (Figure

13) which shows that runtime decreases as ϵ_b increases. Accordingly, runtime decreases as K increases. This behavior matches our expectation for a scaling factor in the 0-1 knapsack.

Furthermore, an increased K also means an increased ϵ_b (all other parameters constant), a larger $D_{\text{safe}}(\delta)$ (Equation 31), and larger approximation ratio (Equation 30). These relationships also align to expected behavior of a scaling factor in the 0-1 knapsack.

8.2 Experimental Limitations

Experiments focus on two metric groupings: literature metrics and approximation-relevant metrics. In literature, runtime versus mode count of randomly generated tasks, Figure 12, and runtime of literature tasks, Figure 8, were the primary distinguishing metrics. Since this work is approximation-focused, we isolate parameters with the greatest effect on approximation runtime: duration, Figure 9, acceleration, Figure 11, and approximation parameters, Figures 13 and 14. Additional comparison via randomly generated tasks would be beneficial in revealing combined parameter effects outside those revealed here (i.e., precision, duration, acceleration, mode count) and are not included here for time and space limitations.

9 CONCLUSION AND FUTURE WORK

This work provides a BPCKP for AVR task demand calculation based on predefined job sequences. An exact dynamic programming solution and FPTAS are presented. Compared to the state-of-the-art, the proposed FPTAS demonstrates a 7,800x runtime improvement with less than 8% demand overestimate and a 99.99% reduction in RAM usage. This approach is well-suited for modern, resource-limited CPS with variable WCET and period behavior (e.g., wearable devices, brushless DC motors, and satellite magnetorquers in the introduction) as well as the original, motivating system: ICEs.

Future work includes proving NP-hardness for AVR task demand calculation as the approximated approach hints at and replacing the single job release with multiple job releases throughout a single rotation to enable porting to other variable-period systems.

ACKNOWLEDGMENTS

This research was supported in part by the US National Science Foundation (Grant Nos. CNS-2211641, CPS-2038609, CPS-2038726, and IIS-1724227).

A RELATED WORK OVERESTIMATION

Via Speedup Factor. Equation 13 of Guo and Baruah [25], restated here, gives a AVR task set uniprocessor speedup factor $s = 1/(1-\beta+\beta/\eta(\omega))$ where β is the AVR task utilization ratio - the ratio of max AVR utilization to the sum of AVR utilization and non-AVR task (i.e., periodic) utilization, and $\eta(\Omega)$ is the ratio of the single-job MIAT to the same boundary speed over the MIAT under maximum acceleration (e.g., $\max_i\{T(\omega_i,\omega_i)/T(\omega,\sqrt{\omega_i^2+2\alpha^+})\}$). By this equation, if an AVR task is the only task (i.e., $\beta = 1$) and $T_{avr} = (\{9.0e5 \text{ us}, 600 \text{ us}\}, \{100 \text{ rpm}, 3200 \text{ rpm}\}, 6.0e5 \text{ rpm}^2, -6.0e5 \text{ rpm}^2)$ is used, then $\eta(\Omega) = 1.3620$ making $s = 1.3620$. This indicates we are "wasting" $(1 - 1/1.3620) > 26\%$ processor capacity.

Via Linear Approximation. If used as a linear approximation, the "most dense cycle" (from the Digraph Real-Time approach [52]) of T_{avr} above gives 90% utilization. At $\delta = 20$ ms, the linearly approximated demand would be 20 ms · 0.9 = 18 ms despite the actual, exact demand being 0.6 ms (a 2,900% overestimate). and $\delta = 2$ s, 1.9 s despite the actual, exact demand being 0.6312 s (a 42% overestimate). Overestimation in both cases far exceeds 8% offered by APX, even for large δ.

REFERENCES

[1] Meena Belwal and T. S. B. Sudarshan. 2014. A survey on design space exploration for heterogeneous multi-core. In *2014 International Conference on Embedded Systems (ICES)*. IEEE, Coimbatore, India, 80–85. https://doi.org/10.1109/EmbeddedSys.2014.6953095

[2] Johan Bengtsson, Kim Larsen, Fredrik Larsson, Paul Pettersson, and Wang Yi. 1996. UPPAAL—a tool suite for automatic verification of real-time systems. In *Proceedings of the DIMACS/SYCON workshop on Hybrid systems III : verification and control: verification and control*. Springer-Verlag, Berlin, Heidelberg, 232–243.

[3] Gerald Bieber, Thomas Kirste, and Michael Gaede. 2014. Low sampling rate for physical activity recognition. In *Proceedings of the 7th International Conference on PErvasive Technologies Related to Assistive Environments (PETRA '14)*. Association for Computing Machinery, New York, NY, USA, 1–8. https://doi.org/10.1145/2674396.2674446

[4] Sandeep Kumar Bijinemula, Aaron Willcock, Thidapat Chantem, and Nathan Fisher. 2018. Code for the paper-Efficient knapsack-based approach for calculating the worst-case demand of AVR tasks. https://github.com/bsk1410/Efficient-Knapsack-for-AVR-tasks-RTSS2018

[5] Sandeep Kumar Bijinemula, Aaron Willcock, Thidapat Chantem, and Nathan Fisher. 2018. An Efficient Knapsack-Based Approach for Calculating the Worst-Case Demand of AVR Tasks. In *2018 IEEE Real-Time Systems Symposium (RTSS)*. IEEE, Nashville, TN, USA, 384–395. https://doi.org/10.1109/RTSS.2018.00053 ISSN: 2576-3172.

[6] Alessandro Biondi and Giorgio Buttazzo. 2015. Engine control: Task modeling and analysis. In *2015 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, Grenoble, France, 525–530. https://doi.org/10.7873/DATE.2015.0147 ISSN: 1558-1101.

[7] Alessandro Biondi and Giorgio Buttazzo. 2018. Modeling and Analysis of Engine Control Tasks Under Dynamic Priority Scheduling. *IEEE Transactions on Industrial Informatics* 14, 10 (Oct. 2018), 4407–4416. https://doi.org/10.1109/TII.2018.2791939 Conference Name: IEEE Transactions on Industrial Informatics.

[8] Alessandro Biondi, Giorgio Buttazzo, and Stefano Simoncelli. 2015. Feasibility Analysis of Engine Control Tasks under EDF Scheduling. In *2015 27th Euromicro Conference on Real-Time Systems*. IEEE, Lund, Sweden, 139–148. https://doi.org/10.1109/ECRTS.2015.20 ISSN: 2377-5998.

[9] Alessandro Biondi, Marco Di Natale, and Giorgio Buttazzo. 2015. Response-time Analysis for Real-time Tasks in Engine Control Applications. In *Proceedings of the ACM/IEEE Sixth International Conference on Cyber-Physical Systems (ICCPS '15)*. ACM, New York, NY, USA, 120–129. https://doi.org/10.1145/2735960.2735963 event-place: Seattle, Washington.

[10] Alessandro Biondi, Marco Di Natale, and Giorgio Buttazzo. 2018. Response-time analysis of engine control applications under fixed-priority scheduling. *IEEE Trans. Comput.* 67, 5 (2018), 687–703. https://doi.org/10.1109/TC.2017.2777826 Publisher: IEEE.

[11] Alessandro Biondi, Alessandra Melani, Mauro Marinoni, Marco Di Natale, and Giorgio Buttazzo. 2014. Exact Interference of Adaptive Variable-Rate Tasks under Fixed-Priority Scheduling. In *2014 26th Euromicro Conference on Real-Time Systems*. IEEE, Madrid, Spain, 165–174. https://doi.org/10.1109/ECRTS.2014.38 ISSN: 2377-5998.

[12] Giorgio C. Buttazzo, Enrico Bini, Darren Buttle, Scuola Superiore, and Sant Anna. 2014. Rate-Adaptive Tasks: Model, Analysis, and Design Issues. In *Design, Automation & Test in Europe Conference & Exhibition (DATE), 2014*. IEEE Conference Publications, New Jersey, 1–6. https://doi.org/10.7873/DATE.2014.266

[13] Darren Buttle. 2012. Real-Time in the Prime-Time. In *2012 24th Euromicro Conference on Real-Time Systems*. IEEE, Pisa, Italy, xii–xiii. https://doi.org/10.1109/ECRTS.2012.7 ISSN: 2377-5998.

[14] Yi Cao and Wen-Hua Chen. 2009. Automatic differentiation based nonlinear model predictive control of satellites using magneto-torquers. In *2009 4th IEEE Conference on Industrial Electronics and Applications*. IEEE, Xi'an, China, 913–918. https://doi.org/10.1109/ICIEA.2009.5138329 ISSN: 2158-2297.

[15] Yi Cao and Wen-Hua Chen. 2014. Variable sampling-time nonlinear model predictive control of satellites using magneto-torquers. *Systems Science & Control Engineering* 2, 1 (Dec. 2014), 593–601. https://doi.org/10.1080/21642583.2014.956841 Publisher: Taylor & Francis _eprint: https://doi.org/10.1080/21642583.2014.956841.

[16] Shih-Lun Chen, Jocelyn Flores Villaverde, Ho-Yin Lee, Danny Wen-Yaw Chung, Ting-Lan Lin, Chih-Hao Tseng, and Kuei-An Lo. 2017. A Power-Efficient Mixed-Signal Smart ADC Design With Adaptive Resolution and Variable Sampling

Rate for Low-Power Applications. *IEEE Sensors Journal* 17, 11 (June 2017), 3461–3469. https://doi.org/10.1109/JSEN.2017.2680472 Conference Name: IEEE Sensors Journal.

[17] Robert I. Davis, Timo Feld, Victor Pollex, and Frank Slomka. 2014. Schedulability tests for tasks with Variable Rate-dependent Behaviour under fixed priority scheduling. In *Real-Time Technology and Applications - Proceedings*, Vol. 2014-Octob. IEEE, Berlin, Germany, 51–62. https://doi.org/10.1109/RTAS.2014.6925990

[18] William R. Dieter, Srabosti Datta, and Wong Key Kai. 2005. Power reduction by varying sampling rate. In *Proceedings of the 2005 international symposium on Low power electronics and design (ISLPED '05)*. Association for Computing Machinery, New York, NY, USA, 227–232. https://doi.org/10.1145/1077603.1077658

[19] Timo Feld. 2020. *Response time analyses of adaptive variable-rate-tasks*. Dissertation. Universität Ulm. https://doi.org/10.18725/OPARU-24369 Accepted: 2020-01-23T13:51:11Z ISBN: 9781688526969.

[20] Timo Feld, Alessandro Biondi, Robert I. Davis, Giorgio Buttazzo, and Frank Slomka. 2018. A survey of schedulability analysis techniques for rate-dependent tasks. *Journal of Systems and Software* 138 (April 2018), 100–107. https://doi.org/10.1016/j.jss.2017.12.033

[21] Timo Feld and Frank Slomka. 2015. Sufficient response time analysis considering dependencies between rate-dependent tasks. In *2015 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, Grenoble, France, 519–524. https://doi.org/10.7873/DATE.2015.0150 ISSN: 1558-1101.

[22] Timo Feld and Frank Slomka. 2018. Exact Interference of Tasks With Variable Rate-Dependent Behavior. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 37, 5 (May 2018), 954–967. https://doi.org/10.1109/TCAD.2017.2729459 Conference Name: IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems.

[23] Timo Feld and Frank Slomka. 2019. A Sufficient Response Time Analysis Considering Angular Phases Between Rate-Dependent Tasks. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 38, 11 (Nov. 2019), 2008–2021. https://doi.org/10.1109/TCAD.2018.2878163 Conference Name: IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems.

[24] Michael R. Garey and David S. Johnson. 1979. *Computers and Intractability; A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co., USA.

[25] Zhishan Guo and Sanjoy K. Baruah. 2015. Uniprocessor EDF scheduling of AVR task systems. In *ACM/IEEE 6th International Conference on Cyber-Physical Systems, ICCPS 2015*. Association for Computing Machinery, Inc, Seattle, Washington, 159–168. https://doi.org/10.1145/2735960.2735976

[26] Marius Herget, Faezeh Sadat Saadatmand, Martin Bor, Ignacio González Alonso, Todor Stefanov, Benny Akesson, and Andy D. Pimentel. 2022. Design Space Exploration for Distributed Cyber-Physical Systems: State-of-the-art, Challenges, and Directions. In *2022 25th Euromicro Conference on Digital System Design (DSD)*. IEEE, Maspalomas, Spain, 632–640. https://doi.org/10.1109/DSD57027.2022.00090 ISSN: 2771-2508.

[27] John Heywood. 1988. *Internal Combustion Engine Fundamentals*. McGraw-Hill Education, New York, NY, USA. Google-Books-ID: u9FSAAAAMAAJ.

[28] John B. Heywood. 2018. *Internal Combustion Engine Fundamentals* (2nd edition ed.). McGraw-Hill Education, New York, NY, USA. https://www.accessengineeringlibrary.com/content/book/9781260116106

[29] Wen-Hung Huang and Jian-Jia Chen. 2015. Techniques for Schedulability Analysis in Mode Change Systems under Fixed-Priority Scheduling. In *2015 IEEE 21st International Conference on Embedded and Real-Time Computing Systems and Applications*. IEEE, Hong Kong, China, 176–186. https://doi.org/10.1109/RTCSA.2015.36 ISSN: 2325-1301.

[30] Chung-Wen Hung, Jhih-Han Chen, and Hsuan T. Chang. 2011. A Minimal Fuzzy Gain Scheduling Speed Controller and Torque Compensation for the Variable Sampling System of BLDC Motors. In *2011 Fifth International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing*. IEEE, Seoul, Korea (South), 434–437. https://doi.org/10.1109/IMIS.2011.85

[31] Chung-Wen Hung, Cheng-Tsung Lin, Chih-Wen Liu, and Jia-Yush Yen. 2007. A Variable-Sampling Controller for Brushless DC Motor Drives With Low-Resolution Position Sensors. *IEEE Transactions on Industrial Electronics* 54, 5 (Oct. 2007), 2846–2852. https://doi.org/10.1109/TIE.2007.901303 Conference Name: IEEE Transactions on Industrial Electronics.

[32] Chung-Wen Hung and Jia-Yush Yen. 2013. A Robust Variable Sampling Time BLDC Motor Control Design Based upon μ-Synthesis. *The Scientific World Journal* 2013 (Nov. 2013), 236404. https://doi.org/10.1155/2013/236404

[33] Oscar H. Ibarra and Chul E. Kim. 1975. Fast Approximation Algorithms for the Knapsack and Sum of Subset Problems. *J. ACM* 22, 4 (Oct. 1975), 463–468. https://doi.org/10.1145/321906.321909

[34] Oscar H. Ibarra and Chul E. Kim. 1978. Approximation Algorithms for Certain Scheduling Problems. *Mathematics of Operations Research* 3, 3 (1978), 197–204. http://www.jstor.org/stable/3689490 Publisher: INFORMS.

[35] Jia-Yush Yen, Yang-Lin Chen, and M. Tomizuka. 2002. Variable sampling rate controller design for brushless DC motor. In *Proceedings of the 41st IEEE Conference on Decision and Control, 2002.*, Vol. 1. IEEE, Las Vegas, NV, USA, 462–467 vol.1. https://doi.org/10.1109/CDC.2002.1184539 ISSN: 0191-2216.

[36] D. S. Johnson and K. A. Niemi. 1983. On Knapsacks, Partitions, and a New Dynamic Programming Technique for Trees. *Mathematics of Operations Research* 8, 1 (1983), 1–14. http://www.jstor.org/stable/3689406 Publisher: INFORMS.

[37] Prachi Joshi. 2018. *Design Space Exploration for Embedded Systems in Automotives*. Ph. D. Dissertation. Virginia Tech, Blacksburg, Virginia. https://vtechworks.lib.vt.edu/items/f2162aeb-b0e4-46fd-a868-3f72deaf5772

[38] Hans Kellerer, Ulrich Pferschy, and David Pisinger. 2004. *Knapsack Problems*. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-540-24777-7

[39] Junsung Kim, Karthik Lakshmanan, and Ragunathan Rajkumar. 2012. Rhythmic tasks: A new task model with continually varying periods for cyber-physical systems. In *Proceedings - 2012 IEEE/ACM 3rd International Conference on Cyber-Physical Systems, ICCPS 2012*. IEEE, Beijing, China, 55–64. https://doi.org/10.1109/ICCPS.2012.14

[40] Marta Kwiatkowska, Gethin Norman, and David Parker. 2011. PRISM 4.0: verification of probabilistic real-time systems. In *Proceedings of the 23rd international conference on Computer aided verification (CAV'11)*. Springer-Verlag, Berlin, Heidelberg, 585–591.

[41] Edward A. Lee and Sanjit A. Seshia. 2017. *Introduction to Embedded Systems* (second ed.). MIT Press, Berkeley, CA, USA. https://ptolemy.berkeley.edu/books/leeseshia/

[42] C. L. Liu and James W. Layland. 1973. Scheduling Algorithms for Multiprogramming in a Hard-Real-Time Environment. *J. ACM* 20, 1 (Jan. 1973), 46–61. https://doi.org/10.1145/321738.321743

[43] Yu Liu, Chao Peng, Yecheng Zhao, Yangyang Li, and Haibo Zeng. 2020. Schedulability Analysis of Engine Control Systems With Dynamic Switching Speeds. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 39, 10 (Oct. 2020), 2067–2080. https://doi.org/10.1109/TCAD.2019.2951124 Conference Name: IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems.

[44] Morteza Mohaqeqi, Jakaria Abdullah, Pontus Ekberg, and Wang Yi. 2017. Refinement of Workload Models for Engine Controllers by State Space Partitioning. In *29th Euromicro Conference on Real-Time Systems (ECRTS 2017) (Leibniz International Proceedings in Informatics (LIPIcs), Vol. 76)*, Marko Bertogna (Ed.). Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, Dagstuhl, Germany, 11:1–11:22. https://doi.org/10.4230/LIPIcs.ECRTS.2017.11 ISSN: 1868-8969.

[45] A. K. Mok. 1983. *FUNDAMENTAL DESIGN PROBLEMS OF DISTRIBUTED SYSTEMS FOR THE HARD-REAL-TIME ENVIRONMENT*. Technical Report. Massachusetts Institute of Technology, USA.

[46] National Science Foundation. 2021. Cyber-Physical Systems (CPS) | NSF - National Science Foundation. https://new.nsf.gov/funding/opportunities/cyber-physical-systems-cps

[47] Chao Peng, Yecheng Zhao, and Haibo Zeng. 2018. Schedulability Analysis of Adaptive Variable-Rate Tasks with Dynamic Switching Speeds. In *2018 IEEE Real-Time Systems Symposium (RTSS)*. IEEE, Nashville, TN, USA, 396–407. https://doi.org/10.1109/RTSS.2018.00054 ISSN: 2576-3172.

[48] Victor Pollex, Timo Feld, Frank Slomka, Ulrich Margull, Ralph Mader, and Gerhard Wirrer. 2013. Sufficient real-time analysis for an engine control unit with constant angular velocities. In *2013 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, Grenoble, France, 1335–1338. https://doi.org/10.7873/DATE.2013.275 ISSN: 1530-1591.

[49] Mohammadreza Sadeghi, Marco Philippi, Amir Mahdian, and Frank Slomka. 2022. MIAT Efficient analysis of adaptive variable-rate tasks. *Journal of Systems Architecture* 127 (June 2022), 102472. https://doi.org/10.1016/j.sysarc.2022.102472

[50] Prashant Giridhar Shambharkar, Siddhant Bhambri, Arnav Goel, and M. N. Doja. 2019. A Survey on Schedulability Analysis of Rate-Adaptive Tasks. In *2019 International Conference on Machine Learning, Big Data, Cloud and Parallel Computing (COMITCon)*. IEEE, Faridabad, India, 277–282. https://doi.org/10.1109/COMITCon.2019.8862266

[51] Albert Solberg. 2018. Model Based Systems Engineering (MBSE). http://www.nasa.gov/consortium/ModelBasedSystems

[52] Martin Stigge, Pontus Ekberg, Nan Guan, and Wang Yi. 2011. The Digraph Real-Time Task Model. In *2011 17th IEEE Real-Time and Embedded Technology and Applications Symposium*. IEEE, Chicago, IL, USA, 71–80. https://doi.org/10.1109/RTAS.2011.15

[53] Uppsala Universitet and Aalborg University. 2024. Home | UPPAAL. https://uppaal.org/

[54] Vijay Vazirani. 2003. *Approximation Algorithms*. Springer, New York, NY, USA. http://link.springer.com/book/10.1007/978-3-662-04565-7

[55] Aaron Willcock. 2024. aarontwillcock/RTNS24-AVR-FPTAS. https://github.com/aarontwillcock/RTNS24-AVR-FPTAS original-date: 2024-08-20T13:46:19Z.

[56] Hugh D. Young, Roger A. Freedman, A. Lewis Ford, and Hugh D. Young. 2012. *Sears and Zemansky's University physics*. Pearson Learning Solutions, San Francisco, CA, USA.

Received 20 February 2007; revised 12 March 2009; accepted 5 June 2009