Demand Characterization of CPS with Conditionally-Enabled Sensors

Aaron Willcock
Department of Computer Science
Wayne State University
Detroit, MI
aaron.willcock@wayne.edu

Nathan Fisher
Department of Computer Science
Wayne State University
Detroit, MI
fishern@wayne.edu

Thidapat Chantem
Department of Electrical and Computer Engineering
Virginia Tech
Arlington, VA
tchantem@vt.edu

***Abstract*—Characterizing computational demand of Cyber-Physical Systems (CPS) is critical for guaranteeing that multiple hard real-time tasks may be scheduled on shared resources without missing deadlines. In a CPS involving repetition such as industrial automation systems found in chemical process control or robotic manufacturing, sensors and actuators used as part of the industrial process may be conditionally enabled (and disabled) as a sequence of repeated steps is executed. In robotic manufacturing, for example, these steps may be the movement of a robotic arm through some trajectories followed by activation of end-effector sensors and actuators at the end of each completed motion. The conditional enabling of sensors and actuators produces a sequence of Monotonically Ascending Execution times (MAE) with lower WCET when the sensors are disabled and higher WCET when enabled. Since these systems may have several predefined steps to follow before repeating the entire sequence each unique step may result in several consecutive sequences of MAE. The repetition of these unique sequences of MAE result in a repeating WCET sequence. In the absence of an efficient demand characterization technique for repeating WCET sequences composed of subsequences with monotonically increasing execution time, this work proposes a new task model to describe the behavior of real-world systems which generate large repeating WCET sequences with subsequences of monotonically increasing execution times. In comparison to the most applicable current model, the Generalized Multiframe model (GMF), an empirically and theoretically faster method for characterizing the demand is provided. The demand characterization algorithm is evaluated through a case study of a robotic arm and simulation of 10,000 randomly generated tasks where, on average, the proposed approach is 231 and 179 times faster than the state-of-the-art in the case study and simulation respectively.**

***Index Terms*—real-time systems, control systems, cyber-physical systems**

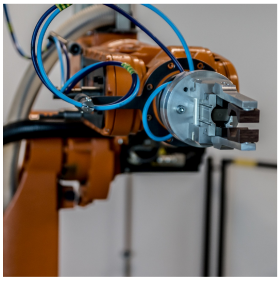
I. Introduction

Characterizing computational demand of Cyber-Physical Systems (CPSs) is critical for guaranteeing multiple hard real-time tasks may be scheduled on a shared processor without missing deadlines. Demand characterization using Demand Bound Functions (DBFs) is important as it bounds the maximum demand a particular task may place on a processor and is commonly used in real-time systems [9], [18], [23]. Precise demand characterization also helps avoid overprovisioning of processing resources as pessimistic characterization can cause designers to use more processors, higher power processors, or both - resulting in an unnecessary increase in the size, weight, and power of computing systems. In modern manufacturing, robotic systems are used to execute repeated motions to process parts. The end effectors of these systems are typically interchangeable tools such a deburrers, drills, welders, grinders, or other manipulators used to process parts being manufactured [12], [22].

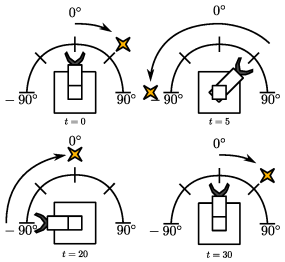
When using a particular end effector, the controller responsible for motion and operation enables the tool as necessary. Consider, for example, a robot like the one depicted in Figure 1a required to move in a pattern shown in Figure 1b. Suppose that upon arrival to the desired locations shown in Figure 1b, the end effector is activated. Since the end effector is enabled as needed, the sensors and actuators of the end effector are not always in use. Thus, the real-time workload associated with sensor sampling, fusion, and control of end-effector actuators is only present during some of the repeated trajectories the robot moves through. The real-time task controlling the robot and its end effector would show a predictably variable sequence of WCETs where lower WCET would be expected without end-effector sensors and actuators enabled but higher WCET expected when in motion and enabling the end effector. This pattern would manifest as a sequence of Monotonically Ascending Execution times (MAE). Moreover, since the MAE sequence would repeat for each unique movement of the robot to a new location where the end effector will be enabled before repeating entirely, several MAE sequences would be concatenated to form a repeating WCET sequence. Figure 2 shows an example repeating WCET sequence composed of two subsequences with Monotonically Ascending Execution times. The repeating WCET sequence repeats at time $t = 16$. Over the intervals $[0, 6]$ and $[6, 16]$ the WCET of jobs is monotonically increasing. In practice, the solid jobs may represent the constant workload of a robotic arm such as the use of accelerometers and gyroscopes to locate the arm in space. The striped jobs would represent the conditionally-enabled workload of the end effector. For deburring end effector, the striped jobs may represent increased workload from encoders and torque sensors regulating speed and force [16].

Research Need: The most applicable demand characterization for systems that demonstrate a repeating WCET sequence

This work is supported in part by the National Science Foundation (CNS-1618185, CNS-1618979, CPS-2038609, CPS-2038726, and IIS-1724227)



(a) An example robot arm with a pneumatic-powered gripper as an end effector.



(b) An example motion pattern for a robot arm which rotates to three positions before repeating.

Fig. 1. A robot arm and an example motion pattern.

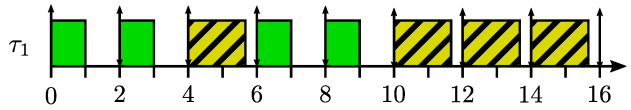


Fig. 2. A repeating WCET sequence composed of two subsequences of MAEs

is the Generalized Multiframe (GMF) model by Baruah et al. [3]. Extending upon the Multiframe model of Mok and Chen [20], the GMF approach provides a Demand Bound Function (DBF) for tasks with a repeating sequence of execution times, such as the robot arm with a conditionally-enabled end effector described above. The DBF is a fundamental tool in real-time systems which describes the maximum demand a set of tasks may place on a processor in a given time interval. While the GMF DBF algorithm in Baruah et al. [3] is an exact characterization of demand, it is designed for arbitrary patterns of frame execution times and thus computationally expensive for systems that exhibit patterns such as repeatable WCET sequences composed of MAE. This computational expense can restrict offline control synthesis and make online schedulability analysis difficult. For example, in the hardware-software co-design process, where physical systems, control laws, and real-time systems are modeled and simulated together, schedulability analysis is performed on many different system implementations. Furthermore, for control systems tasked with creating new trajectories or using new end effectors after deployment (i.e. new motion paths for new parts or retooling for new manufacturing processes), the latency required to reassess schedulability online using the GMF approach is impractical. Restricted to the current computationally expensive approach, designers have the choice of either computing every possible change to the sequence of setpoints offline or using a fast, inexact approach (such as assuming every job executes at the highest WCET) to reassess schedulability online. Thus, any improvements in schedulability runtime make offline synthesis and online recharacterization more practical.

Approach: In the absence of an efficient and exact demand characterization technique for repeating WCET sequence with Monotonically Ascending Execution, this work proposes a task model and DBF calculation for a special case of the GMF task model in which repeated monotonic ascension of execution times are exploited. In the proposed approach, the repeating WCET sequence is modeled as a series of repeating MAE sequences. A user-defined function, called a *driving function*, is used to represent the change in WCET over time. In practice, such as in Figure 1b, this driving function may represent tracking error over time of an asymptotically stable control system. Properties of the driving function, its relationship with WCET, and the monotonically ascending subsequences of the repeating WCET sequence are used to limit the number of intervals that must be searched to obtain an upper bound on the demand of the repeating WCET sequence task. The main contributions of this work are:

1) a new method of modeling real-time workloads as repeating sequences of WCET derived from a generic function definition,
2) an exact DBF calculation for the model - theoretically and empirically faster than the GMF approach,
3) a case study of a robotic arm implementation that exhibits the repeating WCET described, and
4) a simulation of 10,000 randomly generated tasks where, on average, the proposed approach is 179 times faster than the GMF approach.

This work is one of several along a pathway exploiting physical system dynamics (in this case, through control laws) to reduce demand characterization computation time. This work aims to demonstrate that known system dynamics may be leveraged for improving the efficiency of analysis. Our longer term vision is to use the insights into control-based demand characterization from this work and apply it to increasingly complicated control models to improve the efficiency and efficacy of CPS.

II. Related Work

Existing real-time works address scheduling in the context of control systems by focusing on adapting periods or WCET [8], [19]. Works like Branickey et al. [7], Baruah et al. [2], and Yoshimoto and Ushio [25] address the possibility of skipping jobs while maintaining stability. These approaches do not offer a precise bounding function for changing WCET, however, and cannot directly be applied to exploiting predictably repeating variations in WCET.

Real-time models which address variation in WCET as a function of the environment include works like Chen et al. [10] and Lee et al. [17] in which Dynamic Voltage and Frequency Scaling (DVFS) is adapted as a function of known workload or thermal constraints. Others like Biondi and Buttazzo [6] and Kim et al. [15] base task period and WCET on the speed of an internal combustion engine. These works all alter task parameters as a function of some changing system dynamic and provide methods for modeling the demand generated by systems with adaptive WCET. These works, however, do not define a predictably repeating variation in the sequence of WCETs that could be produced and therefore do not apply to the repeating WCET sequence.

In terms of real-time analysis models, the approach most applicable to analyzing a repeating WCET sequence is the

Generalized Multiframe (GMF) model. The Multiframe task model by Mok and Chen describes tasks with WCETs that repeat in a cycle throughout system operation [20]. The GMF task model by Baruah et al. further extends on Mok and Chen by allowing deadlines of frames to differ and varying minimum interarrival times [3]. These models provide an exact DBF when applied to repeating WCET sequence problem. However, the time required to compute the DBF using the GMF model for repeating WCET sequence problem is significantly longer and, as will be shown, can be improved by leveraging properties repeating WCET sequence, namely the monotonicity of the MAE subsequences.

III. System Model

In the GMF task model, a sequence of frames with WCETs are explicitly defined. The GMF DBF algorithm is then used to characterize the maximum demand the sequence of frames may generate over any interval. In this work, we provide a task model that is a special case of the GMF model in which the sequence of frames is generated by: a *driving function* representing how WCET changes in time, a *super schedule* which represents how the *driving function* and its sequence of initial values reset in time to simulate unique motions, and an explicit mapping of the *driving function* to WCET. In the model that follows, the driving function, super schedule, and the mapping of driving function to WCET will be used to generate a sequence of jobs, individual units of computation, whose WCET varies in time. This section describes each of these components and concludes with the proposed task model.

A. The Driving Function

For this model, the *driving function* is a generic, user-defined function that serves as the basis for WCET values in time. The driving function is given by:

$$f(t) : \mathbb{R}_0^+ \mapsto \mathbb{R}_0^+ \mid f(t) > f(t+\epsilon)\ \forall\, \epsilon > 0 \tag{1}$$

where t represents the function input which increases with respect to time from some initial value of t. Although $f(t)$ may be any decreasing function which maps to nonnegative reals, practically $f(t)$ may represent the decrease in error over time of an asymptotically stable control system. In the robot example given in the introduction, this function could be derived from the control law responsible for driving the arm and end effector towards the sequence of target locations. Since the motivation behind this work is repeating trajectories of motion, the driving function is accompanied by a *super schedule* to describe repetition.

B. Super Schedule

To define the repetition of the driving function values, we define a *super schedule*, $\mathbb{S}$, as 3-tuple:

$$\mathbb{S} = \{\mathbf{R}, \mathbf{S}, P\}, \tag{2}$$

where $\mathbf{R}$ is the reset sequence, $\mathbf{S}$ is the starting value sequence, and P is the *super period*. Figure 3 illustrates a sequence of reset times and starting values along with a super period.

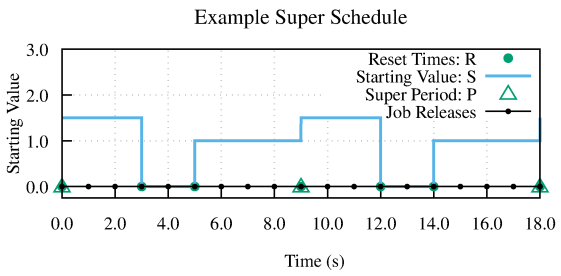


Fig. 3. An Example Super Schedule vs Time. $\mathbb{S} = \{\mathbf{R} = \{0, 3.0, 5.0\}, \mathbf{S} = \{1.5, 0, 1\}\}, P = 9.0\}$ are the parameters used to construct this graph. Along $y = 0$, the black dots indicate job release times. The starting values and reset times repeat after the super period $P = 9.0$.

The following equations provide formal definitions of the sequences above. The reset sequence $\mathbf{R}$ is a sequence of reset times from system start when the starting values of $\mathbf{S}$ are used as initial values for $f(t)$:

$$\mathbf{R} = \{r_0, r_1, ..., r_n\} \mid r_i \in \mathbb{R}_0^+, r_i < r_{i+1},\ \forall\, i < n \tag{3}$$

where n is the number of elements in the sequence and $r_n = P$. Here, P, the super period, is the time at which the sequence of reset values and starting values restarts from r_0 and s_0 respectively. This restart can be seen in Figure 3 at time $t = 9$. The starting value sequence $\mathbf{S}$ is defined as:

$$\mathbf{S} = \{s_0, s_1, ..., s_{n-1}\} \mid s_i + (r_{i+1} - r_i) \geq s_{(i+1) \bmod n} \quad \forall i \leq n-1, s_i \in \mathbb{R}_0^+ \tag{4}$$

where s_i is the initial value given as input to $f(t)$ at time r_i. Specifically, at time $t = r_i$, $f(s_i)$ is the output of the driving function. The constraint $s_i + (r_{i+1} - r_i) \geq s_{(i+1) \bmod n}$ requires that starting values provided at reset times only cause $f(t)$ to increase in value. Since $f(t)$ is a strictly decreasing function, $s_i + (r_{i+1} - r_i) \geq s_{(i+1) \bmod n} \implies f(s_i + (r_{i+1} - r_i)) \leq f(s_{(i+1) \bmod n})$. Informally, the reset sequence identifies when, in time, the driving function will be "reset" to some initial value from which it decays. Practically, a reset sequence may be determined by the set of unique setpoints a robot must drive its end effector towards. The starting value sequence represents the initial value from which the function decays. In practice, these starting values may be derived from the known distance the manipulator must cover before the end effector would be enabled. In examples like the robotic arm and other activities that include motion planning, trajectories are provided to controllers as repeating sequences typically for manufacturing or other automated, high-repetition activities [13], [14], [21]. Although the basis for these terms *may* be rooted in control, there is no implicit requirements for generating the repeating WCET sequence outside of the equation definitions above.

C. Starting Value Index

Since the starting values, S, are used as input to the driving function, $f(t)$, an index function is needed to identify the

current starting value. The *Starting Value Index*, defined below, identifies which starting value, s_i, is used at any given time t.

Definition 1 (Starting Value Index): Let $r_n = P$. At time t the Starting Value Index ($svi(t)$) is:

$$svi(t) = \{i \text{ if } (0 \leq i < n-1) \wedge (r_i \leq t \bmod P < r_{i+1}) \quad (5)$$

This index determines which two consecutive reset times, r_i and $r_{(i+1) \bmod (n+1)}$, the provided time t is between. Note that this function is discontinuous $\forall\, t \mid t \bmod P \in \mathbf{R}$.

Since the driving function "resets" at each reset time, the instantaneous value of the driving function at any time is not strictly a function of $f(t)$.

D. Instantaneous Driving Function Values

To determine the driving function value at any t, a function for modeling the time since the last reset is needed. To model the time since the last reset, let us define a function which increases linearly from zero at and after any reset time.

Property 1 (Time Since Reset): At time t, the time from the last reset is:

$$T(t) = t - r_{svi(t)} - P \cdot \left\lfloor \frac{t}{P} \right\rfloor \quad (6)$$

Note this function, based on Equation 5, is discontinuous at any time $t \,|\, t \bmod P \in \mathbf{R}$.

The time since last reset function, Equation 6, may now be incorporated into the instantaneous driving function. The instantaneous driving function represents the output of the driving function $f(x)$ resetting at each reset time, $r \in \mathbf{R}$.

Property 2 (Instantaneous Driving Function): At time t, the instantaneous value of the driving function, incorporating resets and starting values, is given by:

$$F(t) = f(s_{svi(t)} + T(t)) \quad (7)$$

where $svi(t)$ is the index of the starting value at time t, $s_{svi(t)}$ is the starting value itself, and $T(t)$ is the time since last reset. This function gives the instantaneous value of the driving function at any time t while incorporating the reset times and initial values. The output of the instantaneous driving function may now be used as the basis for generating WCET values.

To represent the changing WCET resulting from conditionally-enabled end-effector sensors and actuators, we now define the sensor boundaries and WCET functions.

1) Sensor Boundaries: As previously mentioned, sensors and actuators for an end effector may only be enabled as necessary. For this work, enabling of sensors and actuators will be based on the value of the driving function, $f(t)$. The values of $f(t)$ at which different WCETs are used are called *boundaries*. The relationship between the driving function, WCET, and sensor boundaries is given by:

$$\mathbf{W} = (\mathbf{C}, \mathbf{B}), \quad (8)$$

where $\mathbf{C}$ is the set of WCETs for the conditionally-enabled sensors and $\mathbf{B}$ is the set of *boundaries*.

Specifically, the WCET set $\mathbf{C}$ is defined as:

$$\mathbf{C} = \{c_0, c_1, ..., c_{m-1}\} \mid c_i \in \mathbb{R}^+_{>0}, c_i > c_{i+1},\ \forall i \leq m-1, \quad (9)$$

where c_i represents the WCET of the task for values of $f(t)$ between boundaries b_i and b_{i+1} and m is the number of unique WCETs. Note that all sets described in the remainder of the work will begin with subscript zero.

The sensor boundary set $\mathbf{B}$ is defined as:

$$\mathbf{B} = \{b_0, b_1, b_2, ..., b_m\} \mid b_i \in \mathbb{R}^+_0, b_i < b_{i+1},\ \forall\, i \leq m, \quad (10)$$

where b_i represents the value of the driving function below which WCET c_{i-1} is used. Whenever $f(t) \leq b_i$ the sensor(s) or actuator(s) with WCET c_{i-1} is executed. Additionally, $b_0 = 0$ which is the minimum for any boundary set. Note that there is one more boundary than unique WCET as the boundaries are the beginning and ends of the ranges. In practice, conditionally-enabled sensors may contribute additional WCET by requiring the sensor data to be sampled, filtered, and compared with other data. Any additional sensor fusion computation, such as fault detection routines, would also increase task WCET when a sensors is enabled. The boundaries for these sensors may be based on system tracking error - only enabling sensors, and thus increasing WCET, when system error is low.

Together, the above equations may be combined into a left continuous piecewise function describing the WCET as a function of the driving function:

$$C(f(t)) = \{c_{i-1} \text{ if } (b_{i-1} < f(t) \leq b_i), i \in \mathbb{N}, i \leq m. \quad (11)$$

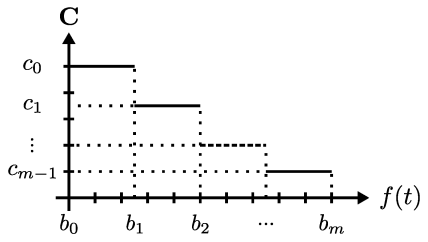


Fig. 4. WCET vs. driving function, $f(t)$, with WCET boundaries, b_i. The x-axis shows boundary values of $\mathbf{B}$. The y-axis shows WCET values of $\mathbf{C}$.

Fig. 4 illustrates the piecewise function in Equation 11. This inverse relationship between the driving function and WCET is similar to the relationship between period and WCET as presented in the Adaptive Variable Rate model by Biondi and Buttazzo [6] and the Rhythmic Task Model by Kim et al. [15].

$F(t)$ may now replace $f(t)$ in Equation 11 to give the sequence of WCETs over time. Since $F(t)$ is strictly decreasing except at times $t \in R$, the final property is now established.

E. Nondecreasing WCET in Time

Given the properties defined above, a final property of nondecreasing WCET in time between consecutive reset times is established. The nondecreasing nature of WCET over time will be exploited in the worst-case demand (WCD) calculation.

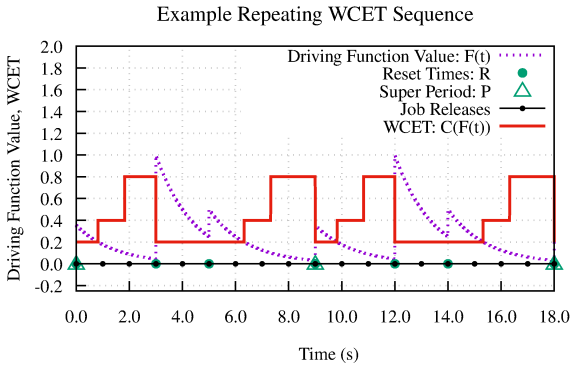


Fig. 5. An Example Repeating WCET Sequence with Monotonically Ascending Execution. The driving function $f(t) = 2^{-t}$ combined with period $p = 1$, reset times $\mathbf{R} = \{0, 3.0, 5.0, 9.0\}$, super period $P = 9$, and starting values $\mathbf{S} = \{1.5, 0, 1\}$ is used as input to the instantaneous driving function $F(t) = f(s_{svi(t)} + T(t))$. The instantaneous driving function is then used as input to the WCET function given by $\mathbf{B} = \{0.0, 0.1, 0.2, 1.0\}$ and $\mathbf{C} = \{0.8, 0.4, 0.2\}$. At time $t = 0$, the system is given the starting value $s_0 = 1.5$ shown above. As $F(t)$ decreases, the WCET of job releases increases according to Equation 11. Along $y = 0$, the black dots indicate job release times. Note the schedule repeats every $P = 9$ time units.

Lemma 1 (Nondecreasing WCET): WCET is nondecreasing between two consecutive reset times.

Proof 1: Let there be an interval $[t, t + \delta]$ where $\delta < P$ and the Starting Value Indices at t and $t + \delta$ are equal ($svi(t) = svi(t+\delta)$) meaning this time interval is between two consecutive reset times. By Equation 7, the driving function value will decrease over the interval $[t, t + \delta)$ (meaning $F(t) > F(t + \delta)$). By Equation 11, a decrease in the driving function value maintains or increases job WCET ($F(t) > F(t+\delta) \Rightarrow C(F(t)) \leq C(F(t+\delta))$). Together, Equations 11 and 7 show that WCET is nondecreasing in time over intervals between two consecutive reset times.

F. Monotonically Ascending Repeating WCET Task

Using the concepts above, a new real-time task model is now proposed which extends the periodic task model:

$$\tau = (p, f, \mathbb{S}, \mathbf{W}, d) \mid p \geq c_i \; \forall i \; \in \; \mathbb{Z}, i \leq m - 1, \qquad (12)$$

where p, the period, specifies the time between job releases, f, the driving function, $\mathcal{S}$, the *super schedule*, defines when f resets and to which values, $\mathbf{W}$, the WCET function, defines the set of WCETs and boundaries, and d, relative deadline, is the duration of time after a job release that the job must be completed. For this work, implicit deadlines are assumed ($d = p$). The key properties of this model are shown in Fig. 5, an example superschedule with WCET values shown.

IV. Constructing the Demand Bound Function

This section describes an analytic approach to constructing a Demand Bound Function (DBF), a function which characterizes the worst-case demand of a real-time task, for the proposed task model. This section also presents methods for reducing the search space for calculating the DBF, and concludes with the DBF algorithm itself. This DBF algorithm constructs a table of DBF values which are used in schedulability analysis. Note that we focus on demand characterization in general and not the underlying systems (i.e. uniprocessor vs multiprocessor). Existing works focus on the application of DBFs to specific systems such as multiprocessors [11].

A. The Demand Bound Function and its Properties

The DBF, introduced by Baruah et al. [4], is a function which characterizes demand by providing the maximum cumulative execution time a task (or set of tasks) may require from a processor over any interval of size δ. The DBF, therefore, provides the maximum execution for any possible legal job arrivals for a task system. For this work, the same definition presented in Baruah et al. [4] is used.

Definition 2 (Demand bound Function): The *demand bound function*, $DBF(\delta)$, gives the maximum cumulative WCET of all jobs of a task with both release times and deadlines within any time interval of length δ.

A task set $\mathrm{T} = \{\tau_1, \tau_2, \ldots, \tau_n\}$ can be scheduled on a preemptive single processor using Earliest Deadline First (EDF) scheduling *if and only if* the sum of all tasks' $DBF(\delta)$ is less than δ for all $\delta > 0$ [4]. The proposed DBF calculation, therefore, can guarantee the schedulability of real-time tasks.

B. The GMF Approach to DBF

The GMF model presented in Baruah et al. [3] gives a DBF for tasks with a repeating sequence of WCETs called frames. The GMF task model is given by $(\vec{E}, \vec{D}, \vec{P})$ where all three parameters are vectors of length N, $\vec{E}$ represents frame WCETs, $\vec{D}$ represents relative deadlines, and $\vec{P}$ minimum separations. Note that for the proposed approach, $\vec{P}$ corresponds to the task periods. In the GMF model, the DBF calculation requires that each frame be the starting point of a search for WCD and has a running time $O(N^2 \log N)$ with N being the number of frames. Since the WCET sequence repeats until the super period, P, the number of frames as interpreted by the GMF approach would be $\left\lceil \frac{P}{p} \right\rceil$. Assuming the system is discretized and scaled such that period $p = 1$, this would give a running time of $O(P^2 \log P)$. In comparison, the approach to be shown has a running time of $O(nP)$. Note that in the case where the number of reset events is equal to the super period (i.e. $n = P$), the the worst case complexity becomes $O(P^2)$. Section V also shows empirically that the proposed approach significantly improves the latency in computing demand over a GMF-based approach for repeating WCET sequences.

C. The Demand Window and its Properties

To create a DBF for the proposed task model, *demand windows* are used to define the time interval over which the worst-case demand should be calculated. Let a demand window be modeled as:

$$w = (a, \delta) \mid a, \delta \in \mathbb{R}_0^+ \qquad (13)$$

which defines the interval $[a, a + \delta]$ where a is the offset from $t = 0$ and δ is the width of the interval. The demand for a

given w is then the sum of WCETs of jobs with both releases and deadlines within the interval.

For a given demand window, w, since implicit deadlines are used ($p = d$), the maximum number of jobs of a task with releases and deadlines within the window is:

$$n_j(\delta) = \left\lfloor \frac{\delta}{p} \right\rfloor. \tag{14}$$

When building a DBF, a safe approach is to search all possible window positions and sizes since $DBF(\delta)$ must always be greater than or equal to the demand of any individual window of the same size. To limit the number of windows that must be examined, the following lemma establishes that any window, w, may be transformed into a new window, w', in which the window size is an integer multiple of the period (i.e. $\delta \bmod p = 0$) while maintaining the same maximum number of jobs that may be contained within the window.

Lemma 2 (Demand Window Discretization): $\forall w = (a, \delta) \mid \delta \bmod p \neq 0$,
$\exists w' = \left(\left\lfloor \frac{a}{p} \right\rfloor p, \left\lfloor \frac{\delta}{p} \right\rfloor p \right) \mid n_j(\delta) = n_j \left(\left\lfloor \frac{\delta}{p} \right\rfloor \cdot p \right)$

Proof 2: Let $\mathbb{W}$ be the set of all demand windows, $w = (a, \delta)$. Let $w = (a, \delta) \in \mathbb{W}$ where $\delta \bmod p \neq 0$. Let $\mathbb{W}'$ be the set of all demand windows, $w = (a, \delta) \mid \delta \bmod p = 0$. Let $w' = \left(\left\lfloor \frac{a}{p} \right\rfloor \cdot p, \left\lfloor \frac{\delta}{p} \right\rfloor \cdot p \right) \in \mathbb{W}'$. The maximum number of jobs of a task with releases and deadlines within w is given by Equation 14. The maximum number of jobs of a task with releases and deadlines within w' is:

$$n_j \left(\left\lfloor \frac{\delta}{p} \right\rfloor \cdot p \right) = \left\lfloor \frac{\left\lfloor \frac{\delta}{p} \right\rfloor \cdot p}{p} \right\rfloor = \left\lfloor \left\lfloor \frac{\delta}{p} \right\rfloor \right\rfloor = \left\lfloor \frac{\delta}{p} \right\rfloor. \tag{15}$$

Since $n_j(\delta) = n_j \left(\left\lfloor \frac{\delta}{p} \right\rfloor \cdot p \right)$, maximum number of jobs of a task with releases and deadlines within w remains constant after reducing a and δ to the closest integer multiple of p releases and deadlines. Thus, the maximum demand either window could contain is equal.

Given the above lemma, any window depicted in the remainder of the work is assumed to have an offset, a, and window size, δ, which are integer multiples of p - meaning the leftmost ($t = a$) and rightmost ($t = a + \delta$) points of the window will align with job releases.

D. Reducing DBF Search Space

To further reduce the search space, two additional methods are presented. First, the WCD of a single super period, P, is shown to be sufficient for calculating demand of windows larger than the super period (i.e. when $\delta > P$). Second, any window not aligned with a reset time is shown to be transformable into a window which is aligned with a reset time and has equal or greater demand. The nondecreasing nature of job WCETs after a reset time enables this second method. These techniques combine to produce the DBF calculation procedure presented in Algorithm 1 (in Section IV-E).

1) Shortening Windows with Super Periods: Suppose the DBF for a given repeating WCET sequence is provided for all values of δ in the range $[0, P]$. For any window $w = (a, \delta)$ in which $\delta > P$, the worst-case demand may be calculated by multiplying the WCD over a single setpoint period (DBF(P)) by the number of setpoint periods, P, that fit within δ ($N = \lceil \frac{\delta}{P} \rceil$). The WCD over N setpoint periods, $DBF(P) \cdot N$, may then be summed with the WCD of the remaining time, $DBF(\delta \bmod P)$, to produce the WCD for δ. Formally, the equation for this is given by:

$$DBF(\delta) = \begin{cases} \lfloor \frac{\delta}{P} \rfloor \cdot DBF(P) + DBF(\delta \bmod P) & \text{if } \delta > P \\ \text{lookup in DBF table produced by Alg. 1} & \text{if } \delta \leq P \end{cases} \tag{16}$$

This approach avoids building a DBF for values larger than $\delta = P$ as any window of size $\delta > P$ may be broken down into N number of setpoint period WCDs plus the WCD of some window size $\delta < P$.

2) Aligning Demand Windows: Another search space reduction comes from exploiting the nondecreasing WCETs described in Lemma 1. Since job WCETs are nondecreasing between two consecutive reset times, this allows any window, $w = (a, \delta)$, in which the rightmost edge of the window ($(a+\delta) \bmod P$) does not align with a reset time ($(a+\delta) \bmod P \neq r \; \forall r \in \mathbf{R}$) to be transformed into a window $w' = (a', \delta')$, in which the rightmost edge of the window ($(a' + \delta') \bmod P$) aligns with a reset time ($\exists r \in \mathbf{R} \mid (a' + \delta') \bmod P = r$) while maintaining or increasing demand. The proof for this is broken into three cases presented in Lemmas 3-5 below.

Case 1 addresses when the driving function value of the job released at the leftmost window edge (at $t = a$) is no less than driving function values of the job released at the rightmost window edge ($F(a) \geq F(a+\delta)$). In Case 1, the demand of the window is maintained or increased by increasing the window offset by one period ($a' = a + p$). Case 2 addresses when the driving function value of the job released at the leftmost window edge is less than the driving function value of the job released at the rightmost edge ($F(a) < F(a + \delta)$) **and** when this inequality would be maintained if the offset was reduced by one period ($F(a - p) \leq F(a - p + \delta)$). In Case 2, the demand of the window is maintained or increased by decreasing the window offset by one period ($a' = a - p$). Case 3 addresses when neither of the above cases are true. In this case, it is shown that the window must be aligned with a reset time ($\exists i \in \mathbb{Z}^+ \mid a + \delta \bmod P = r_i$).

Lemma 3 (Case 1: Increasing Window Offsets): Given a demand window $w = (a, \delta)$ where $F(a) \geq F(a + \delta)$ then $w' = (a + p, \delta)$ has equal or greater demand than w.

Proof 3: Let $w = (a, \delta)$ be a window such that $F(a) \geq F(a + \delta)$. Suppose the offset of w is increased by one period such that $a' = a + p$ creating the new demand window w'. The increased offset will reduce the demand of the window by $C(a)$ but raise the demand of the window by $C(a + \delta)$.

$$F(a) \geq F(a + \delta) \text{ By definition}$$
$$\Rightarrow C(a) \leq C(a + \delta) \text{ By Eqn. 11}$$

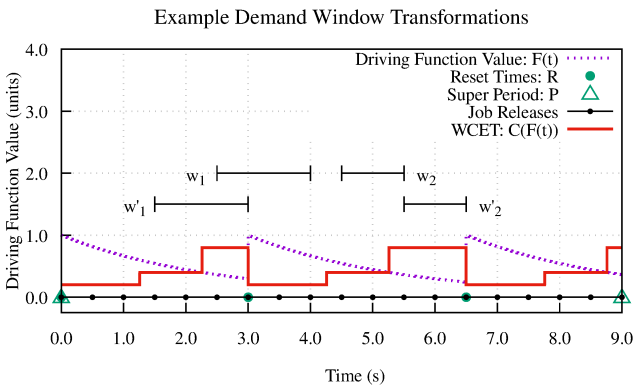


Fig. 6. An example repeating WCET sequence with period $p = 0.5$. The repeated application of Lemma 4 to w_1 results in w_1' which has greater demand and is aligned with the setpoint update at $t = 3$. The repeated application of Lemma 3 to w_2 results in w_2' which has equal demand and is aligned with the setpoint update at $t = 6.5$.

Since $C(a+\delta) \geq C(a)$, w' has no smaller demand than w.

Lemma 4 (Case 2: Decreasing Window Offsets): Given a demand window $w = (a, \delta)$, if $F(a) \leq F(a+\delta)$ and $F(a-p) \leq F(a-p+\delta)$ then $w' = (a-p, \delta)$ has equal or greater demand than w.

Proof 4: The proof is symmetric to the Lemma 3 proof.

Lemma 5 (Case 3: Demand Locally Maximized when Aligned with Reset Times): Given a demand window $w = (a, \delta)$ where $F(a) < F(a+\delta)$ and $F(a-p) > F(a-p+\delta)$ then $(a+\delta) \bmod P \in \mathbf{R}$.

Proof 5: Let $w = (a, \delta)$ be a window such that $F(a) < F(a+\delta)$ and $F(a-p) \geq F(a-p+\delta)$. By Equation 7 the function value decreases as $t \to +\infty$ and increases as $t \to 0$ except at times $t \mid t \bmod P \in R$. Since $F(a-p) \geq F(a-p+\delta)$ and $F(a) < F(a+\delta)$, the function output increased over the interval $[a-p+\delta, a+\delta]$. Therefore, it must be true that $(a+\delta) \bmod P \in \mathbf{R}$ which means the rightmost edge of w, $a+\delta$, is aligned with a reset time.

An illustration of windows to which the above lemmas apply can be found in Fig. 6.

Theorem 1 (Maximization of Window Demand by Alignment with Setpoint Updates): For any window $w \in \mathbb{W}$, the demand of w is maintained or increased when discretized and aligned with a setpoint update - when $(a+\delta) \bmod P \in \mathbf{R}$.

Proof 6: The proof follows directly from Lemmas 3-5.

Corollary 1 (Reset Times as Local Optima): The set of demand windows whose rightmost edges align with a setpoint update, $(a+\delta) \bmod P \in \mathbf{R}$, are local Optima for WCD of windows of the same size δ.

Proof 7: By Theorem 1, any demand window w not aligned with a reset time $(w = (a, \delta) \mid (a+\delta) \bmod P \notin \mathbf{R})$, may be shifted to align with a reset time and maintain or increase demand.

By Theorem 1 and Corollary 1 solving for $DBF(\delta)$ only requires searching for windows of length δ whose rightmost edges $(a+\delta)$ are aligned with a reset time $(\exists i \in \mathbb{Z}^* \mid (a+\delta \bmod P) = r_i)$. Searching for the WCD of a particular window of size δ is now limited to demand windows aligned with reset time. That is, to find $DBF(\delta)$, only n unique window positions must be searched. Having established which finite set of windows need to be searched, the following subsections cover the algorithms for generating the repeating WCET sequence and constructing the DBF.

E. DBF Construction Algorithm

Algorithm 1 gives the procedure for constructing the DBF for this task model. The procedure can be broken into four major steps:

1) Lines 2-4 create the repeating WCET sequence (RWS) by enumerating the WCETs of all jobs within $[0, P)$ using $C(F(t))$.
2) Lines 7-10 calculate the WCD of all windows of size δ that are right-aligned to each reset time for every demand window size in the range $\delta = [1, P]$.
3) Lines 11-12 store the WCD of a fixed right-aligned window and fixed δ.
4) Line 15 stores the WCD of **all** right-aligned windows of fixed δ.

The result is a table of in which any $\delta \leq P \mid \delta \in \mathbb{N}$ has an associated upper bound on demand.

Algorithm 1 MAE repeating WCET sequence DBF

```
1:  procedure MAE-RWS-DBF(p, f(t), 𝕊, W, d, δ)
2:      for t ← 0 to P − 1 do                    ▷ For each job release...
3:          WCETS[t] ← C(F(t)) ▷ Calc. WCET using f(t), 𝕊, W
4:      end for
5:      DBF[0, 1, ..., P] ← 0                    ▷ Init. DBF table
6:      cusum[0, 1, ..., n] ← 0                  ▷ Init. sum vars
7:      for δ ← 1 to P do                        ▷ For each δ size...
8:          maxDemand = 0                        ▷ Init. max demand
9:          for s ← 0 to n do                    ▷ For each reset time...
10:             cusum[s] += WCETS[(r_s − δ)%P]    ▷ Sum WCET.
11:             if cusum[s] > maxDemand then     ▷ Update the max
12:                 maxDemand ← cusum[s]
13:             end if
14:         end for
15:         DBF[δ] ← maxDemand                   ▷ Add entry for δ
16:     end for
17:     Return DBF
18: end procedure
```

Applying the table to Equation 16, gives the complete DBF as the DBF for any $\delta \leq P$ may be found in the table. The DBF may be computed in $O(nP)$ time assuming P is scaled such that $p = 1$. Note that in the case where the number of reset events is equal to the super period (i.e. $n = P$), the the worst case complexity becomes $O(P^2)$. This complexity is a result of exploiting both the homogeneous period and only needing to check windows aligned with reset times - instead of each job release (or frame from the GMF perspective). The GMF approach cannot use either of the above exploits as frames are not guaranteed to have homogeneous periods and each frame is a candidate for beginning a WCD search.

V. Experimental Evaluation and Results

Using the system model and DBF calculations provided, the robotic arm described in the introduction was implemented as a case study except that a conditionally enabled sensors simulated with an ultrasonic rangefinder as opposed to a deburring or welding tool. To explore runtime improvement of the proposed DBF at scale and across random task sets, schedulability analysis was performed on 10,000 randomly generated task sets composed of one RWS task and a random number of periodic tasks. All source code for the experiments and case study may be found online [24]. In both the case study and simulation, analysis is performed for a uniprocessor.

A. Case Study

To evaluate practical application and assess DBF calculation improvement in a real system, a robotic arm, shown in Fig. 7 was implemented. The robotic arm demonstrates conditional enabling of sensors. The robotic arm used FreeRTOS [1] on the Arduino Mega 2560 for computation. The hardware and software setups are provided below.

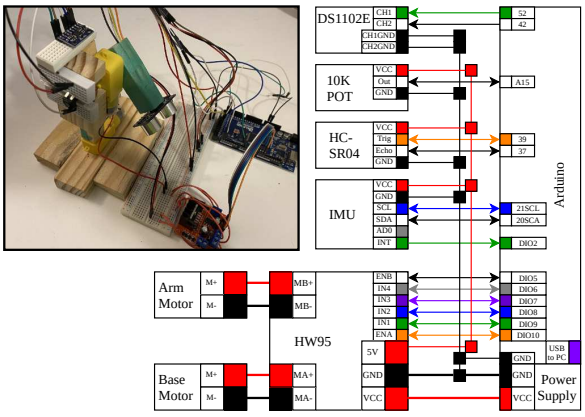


Fig. 7. Case study robot arm and schematic.

1) Hardware Setup: An Arduino Mega 2560 is the real-time controller with a uniprocessor clock speed of 16MHz. For yaw and pitch sensing, an Inertial Measurement Unit (IMU) and potentiometer are used. For yaw and pitch actuation, the arm uses two 6VDC motors with 48:1 transmissions powered by an HW95 breakout with an L298N Dual DC H Bridge. The HC-SR04 Ultrasonic Distance Sensor at the end of the arm simulates the conditionally-enabled sensor (extra computational workload) carried by an end effector. In practice, this may be a tool such as a deburrer, spot welder, drill, or other sensor-enabled end effector.

2) Software Setup: The real-time operating system is a port of FreeRTOS [1] and all code is in C. The AVR-GCC compiler is used (included with the Arduino IDE). Pulse Width Modulation is used to control motors, Inter-Integrated Circuit for the IMU, and Analog-to-Digital Converter for potentiometer sampling. The Simple_MPU6050 library is also used [26]. Two proportional feedback controllers use tracking errors from the IMU and potentiometer to demonstrate repeating motion.

TABLE I
Case Study Model Parameters

Parameter	Value	Units
$\mathbf{C}$	[14.0,6.0,5.0]	ms
$\mathbf{B}$	[0,10,45,180]	degrees
$f(t)$	$120 \cdot e^{-0.0188t}$	N/A
$\mathbf{S}$	[120,0]	degrees
$\mathbf{R}, P$	[0,1080], 2700	ms
p	18	ms
δ	2700	ms

3) Conditional-Enabling Description and Parameterization: The robot arm is provided two setpoints for the arm base to drive to. When the arm base error is below 45 degrees, the end-effector potentiometer is enabled so the wrist may adjust to its setpoint. When the arm base error is below 10 degrees, the ultrasonic rangefinder is enabled. The wrist potentiometer sampling and ultrasonic rangefinder pings activated by the task raise WCET as arm base error decreases. Table I provides the task parameters for the implemented arm. Note that this is a very short sequence of setpoints and the entire motion sequence is completed in 2.7 seconds.

An oscilloscope (DS1102E) was used to verify timing characteristics implemented by the real-time task.

4) Evaluation: Using the empirically derived values as input to the proposed DBF calculation, the runtime improvement of the proposed algorithm was calculated. Recall that the GMF approach and the proposed approach to calculating DBF are both exact; thus the comparison is based on algorithm runtime alone. For the empirically derived task, the GMF DBF calculation completed in $132.930ms$ while the RWS-MAE-DBF completed in $0.574ms$. Thus, proposed DBF calculation was 231.568 times faster than the GMF approach.

B. Simulations

In addition to the case study, simulations were performed to determine the runtime improvement of the proposed DBF at scale and across random task sets. Each task set was composed of a random number of periodic tasks and one RWS task. The periodic tasks were generated using the UUNIFAST approach [5]. The RWS tasks were created using randomly generated values for f, $\mathbb{W}$, $\mathbb{S}$, and $\mathbf{w}$ with uniform distribution. The hyperperiod of the periodic tasks and the RWS tasks were then used to bound the DBF calculation. The bounds on randomly generated parameters are listed in Table II. Note that the target periodic task utilization is constrained to the range $[1 - c_0, 1 - c_{m-1}]$. This constraint prevents the periodic tasks from having utilizations small enough that the task set is always feasible (recall that $p = 1$) or large enough that the task set is never feasible. Keeping periodic task utilizations in the range requires analysis via the DBF and results is a mix of feasible and infeasible task sets. The simulations (including the case study calculation) were developed using Python 3.8.5. The simulating platform was an i7-6700HQ CPU @ 2.60GHz CPU with 16GB RAM.

TABLE II
10,000 TASK SETS RANDOM GENERATION CONSTRAINTS

Parameter	Min. Value	Max. Value
p	1	1
$a, \mathbf{B}$	1	100
$m, n, \mathbf{S}$	1	10
$\mathbf{C}$	1	p
$\mathbf{R}, P$	1	50
δ	p	$10 \cdot P$
Num. Periodic Tasks	1	5
Periodic Task period	1	P
Target Periodic Task utilization	$1 - c_0$	$1 - c_{m-1}$

TABLE III
10,000 TASK SETS AVERAGE STATS

Metric	Average Value
Num Periodic Tasks	3.000
Super Period	193
GMF DBF Calc. Time	657.950 ms
RWS DBF Calc. Time	3.010 ms
RWS DBF Calc. Time Improvement	179.378
Periodic Task Utilization	0.496
Schedulability Analysis Time	0.0004 ms
Schedulable Task Sets Percent	0.455

The average runtime improvement of the DBF calculations are given in Table III. The proposed DBF calculation is, on average, 179.378 times faster than the GMF approach. The schedulability analysis determined that 45.46% of all task sets were feasible. Figure 8 presents all algorithm runtimes. The super period, P, is used as the x-axis since both DBF calculation complexities can be expressed in terms of P. Recall that the GMF DBF runtime complexity is $O(N^2 \log N)$ which translates to $O(P^2 \log P)$ in terms of P. In contrast, the proposed approach has a runtime complexity of $O(n \cdot P)$ which is $O(P^2)$ when $n = P$ in the worst case. Note that in practice $n << P$ since $n \approx P$ implies that the driving function is reset almost as frequently as jobs are released.

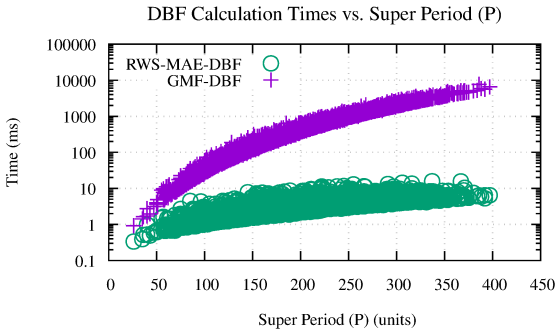


Fig. 8. DBF calculation times for 10,000 randomly generated task sets.

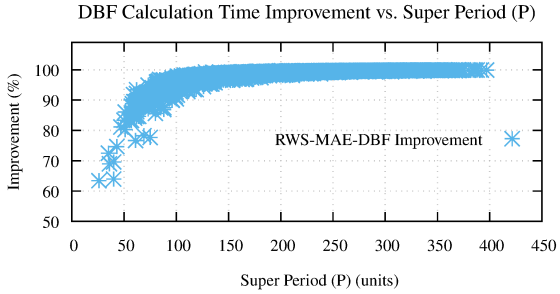


Fig. 9. Calculation time improvement of RWS DBF over GMF

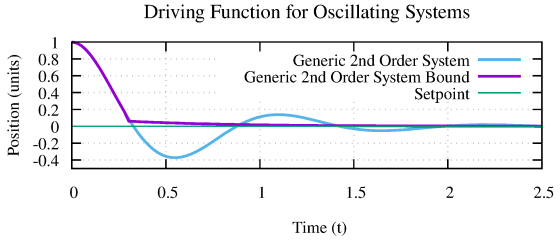


Fig. 10. An example driving function for a generic second order system.

VI. DISCUSSION AND PRACTICAL FACTORS

A. Driving Functions for Oscillating Systems

The model relies on a strictly decreasing driving function. For asymptotically stable systems that do not exhibit oscillation (i.e. first-order systems) the driving function is straightforward. For asymptotically stable systems that exhibit oscillation (i.e. second-order systems), the driving function creates an upper bound on execution time. For example, suppose the following generic second order system is used:

$$x(t) = \left(\frac{e^{-z\omega t}}{\sqrt{1-z^2}}\right) \sin\left(\omega\sqrt{1-z^2}t + \arccos(z)\right) \mid 0 < z < 1, z \in \mathbb{R}_{>0}, \omega \in \mathbb{R}_{\geq 0}$$

One example driving function for this system could be:

$$x_{driving}(t) = \begin{cases} b_1\left(\frac{e^{-z\omega t}}{\sqrt{1-z^2}}\right) & x(t) \leq b_1 \\ b_1\left(\frac{e^{-z\omega t}}{\sqrt{1-z^2}}\right) & t > \frac{\pi - \arccos(z)}{\omega\sqrt{1-z^2}} \\ x(t) & \text{otherwise} \end{cases}$$

since $x_{driving}(t)$ will, by definition, be strictly decreasing, it will provide an upper bound on WCET (since it matches the second order system until $x(t) = b_1$) but not exhibit oscillations. Figure 10 illustrates these equations with $z = 0.3, \omega = 6.0, b_1 = 0.1$.

B. Alternative WCET vs Driving Function Relationships

In this work, WCET is assumed to increase as the driving function decreases. This relationship may be altered and maintain the $O(nP)$ DBF calculation time. For example, suppose the driving function $f(x) = x$, a strictly increasing

function. Keeping the relationship between WCET and the driving function value the same, the WCET sequences would be nonincreasing except at reset times ($t \in R$). Applying a symmetric version of the proofs results in lemmas supporting the search of demand windows *left-aligned* with reset times.

If the relationship between the driving function and WCET were flipped such that lower driving function values implied lower WCET and vice versa. This inverted relationship would cause the sequences of WCETs produced to be nonincreasing except at reset times. Thus, the demand windows which must be searched are again reduced to windows which are *left-aligned* with reset times. While $f(x)$ is defined to be strictly monotone and the function $C(x)$ is monotonic, the only demand windows which must be considered to bound demand will be aligned to reset times.

C. Discretization and Scaling

The model provided accepts continuous-time functions. Modern real-time control, however, is typically performed by discrete-time microprocessors. To discretize the system model, the reset times, $\mathbf{R}$, and super period, P, must be transformed into the smallest integer multiples of the period which exceed the original parameter value. For example, given a reset time r_i, the transformed reset time is $r'_i = \left\lceil \frac{r_i}{p} \right\rceil \cdot p$. This transformation will not alter the original WCET since a controller with period p which receives a starting value at time $t \mid t \bmod p \neq 0$ will be unable act on the new starting value until the next job release at time $\left\lceil \frac{t}{p} \right\rceil \cdot p$. After this transformation, all timing-related values may be scaled such that $p = 1$. The assumption that $p = 1$ is not required but reduces the computational complexity.

VII. Conclusion and Future Work

In this work, a new method of modeling real-time workloads defined by Repeating WCET Sequences with Monotonically Ascending Execution is presented. An exact DBF calculation is provided, implemented, and evaluated through case study and simulation of 10,000 randomly generated task sets. The proposed approach is, on average, 231 and 179 times faster than the state-of-the-art in the case study and simulation respectively. For systems operating in dynamic environments where schedulability is frequently evaluated online, the primary importance of this work is the more tractable characterization of demand. Challenges for future work include allowing multiple driving functions to be used in generating sequences, varying reset times and starting values (for example, as a graph) to incorporate more flexibility.

References

[1] Amazon Web Services, Inc. Market leading RTOS (Real Time Operating System) for embedded systems with Internet of Things extensions. Publication Title: FreeRTOS.

[2] S. Baruah, V. Bonifaci, A. Marchetti-Spaccamela, and V. Verdugo. A Scheduling Model Inspired by Control Theory. In *Proceedings of the 25th International Conference on Real-Time Networks and Systems*, RTNS '17, pages 78–87, New York, NY, USA, 2017. Association for Computing Machinery. event-place: Grenoble, France.

[3] S. Baruah, D. Chen, S. Gorinsky, and A. Mok. Generalized Multiframe Tasks. *Real-Time Systems*, 17(1):5–22, July 1999.

[4] S. Baruah, A. Mok, and L. Rosier. Preemptively scheduling hard-real-time sporadic tasks on one processor. In *[1990] Proceedings 11th Real-Time Systems Symposium*, pages 182–190, Dec. 1990.

[5] E. Bini and G. C. Buttazzo. Measuring the Performance of Schedulability Tests. *Real-Time Systems*, 30(1-2):129–154, May 2005.

[6] A. Biondi and G. Buttazzo. Engine control: Task modeling and analysis. In *2015 Design, Automation Test in Europe Conference Exhibition (DATE)*, pages 525–530, 2015.

[7] M. Branicky, S. Phillips, and Wei Zhang. Scheduling and feedback co-design for networked control systems. In *Proceedings of the 41st IEEE Conference on Decision and Control*, pages 1211–1217, 2003. Issue: December.

[8] A. Cervin, J. Eker, B. Bernhardsson, and K.-E. Årzén. Feedback–Feedforward Scheduling of Control Tasks. *Real-Time Systems*, 23(1):25–53, July 2002.

[9] J.-J. Chen and S. Chakraborty. Resource Augmentation Bounds for Approximate Demand Bound Functions. In *2011 IEEE 32nd Real-Time Systems Symposium*, pages 272–281, Nov. 2011. ISSN: 1052-8725.

[10] J.-J. Chen, S. Wang, and L. Thiele. Proactive Speed Scheduling for Real-Time Tasks under Thermal Constraints. In *2009 15th IEEE Real-Time and Embedded Technology and Applications Symposium*, pages 141–150, Apr. 2009. ISSN: 1545-3421.

[11] R. I. Davis and A. Burns. A survey of hard real-time scheduling for multiprocessor systems. *ACM Computing Surveys*, 43(4):35:1–35:44, Oct. 2011.

[12] EMI Corporation. End of Arm Tooling Components | EOAT Robot Tooling.

[13] A. Gasparetto, P. Boscariol, A. Lanzutti, and R. Vidoni. Trajectory Planning in Robotics. *Mathematics in Computer Science*, 6(3):269–279, Sept. 2012.

[14] B. Hernández and E. Giraldo. A Review of Path Planning and Control for Autonomous Robots. In *2018 IEEE 2nd Colombian Conference on Robotics and Automation (CCRA)*, pages 1–6, Nov. 2018.

[15] J. Kim, K. Lakshmanan, and R. Rajkumar. Rhythmic tasks: A new task model with continually varying periods for cyber-physical systems. *Proceedings - 2012 IEEE/ACM 3rd International Conference on Cyber-Physical Systems, ICCPS 2012*, pages 55–64, 2012.

[16] T. Kubela, A. Pochyly, V. Singule, and L. Flekal. Force-Torque Control Methodology for Industrial Robots Applied on Finishing Operations. In R. Jabloński and T. Březina, editors, *Mechatronics*, pages 429–437, Berlin, Heidelberg, 2012. Springer.

[17] Y. Lee, H. S. Chwa, K. G. Shin, and S. Wang. Thermal-Aware Resource Management for Embedded Real-Time Systems. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 37(11):2857–2868, Nov. 2018. Conference Name: IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems.

[18] M. Mahdiani and A. Masrur. On Bounding Execution Demand under Mixed-Criticality EDF. In *Proceedings of the 26th International Conference on Real-Time Networks and Systems*, RTNS '18, pages 170–179, Chasseneuil-du-Poitou, France, Oct. 2018. Association for Computing Machinery.

[19] P. Martí, C. Lin, S. A. Brandt, M. Velasco, and J. M. Fuertes. Draco: Efficient resource management for resource-constrained control tasks. *IEEE Transactions on Computers*, 58(1):90–105, 2009. Publisher: IEEE.

[20] A. Mok and D. Chen. A multiframe model for real-time tasks. In *17th IEEE Real-Time Systems Symposium*, pages 22–29, Dec. 1996. ISSN: 1052-8725.

[21] S. Y. Nof. *Handbook of industrial robotics*. John Wiley, 1999.

[22] R. O. M. T. POSTED 09/19/2017. EOAT in Robots: A Basic Overview.

[23] N. Serreli, G. Lipari, and E. Bini. The Demand Bound Function Interface of Distributed Sporadic Pipelines of Tasks Scheduled by EDF. In *2010 22nd Euromicro Conference on Real-Time Systems*, pages 187–196, July 2010. ISSN: 2377-5998.

[24] A. Willcock. aarontwillcock/RTCSA21-DC-CES, Apr. 2021. Available at: https://github.com/aarontwillcock/RTCSA21-DC-CES.

[25] T. Yoshimoto and T. Ushio. Optimal arbitration of control tasks by job skipping in cyber-physical systems. *Proceedings - 2011 IEEE/ACM 2nd International Conference on Cyber-Physical Systems, ICCPS 2011*, pages 55–64, 2011. Publisher: IEEE ISBN: 9780769543611.

[26] ZHomeSlice. Simple_mpu6050, Oct. 2019. Publisher: GitHub.