An Efficient Knapsack-Based Approach for Calculating the Worst-Case Demand of AVR Tasks

Sandeep Kumar Bijinemula*, Aaron Willcock†, Thidapat Chantem*, Nathan Fisher†
* Department of Electrical and Computer Engineering, Virginia Tech, Arlington, USA.
†Department of Computer Science, Wayne State University, Detroit, USA.
bsk1410@vt.edu, aaron.willcock@wayne.edu, tchantem@vt.edu, fishern@wayne.edu

***Abstract*—Engine-triggered tasks are real-time tasks that are released when the crankshaft in an engine completes a rotation, which depends on the angular speed and acceleration of the crankshaft itself. In addition, the execution time of an engine-triggered task depends on the speed of the crankshaft. Tasks whose execution times depend on a variable period are referred to as adaptive-variable rate (AVR) tasks. Existing techniques to calculate the worst-case demand of AVR tasks are either inexact or computationally intractable. In this paper, we transform the problem of finding the worst-case demand of AVR tasks over a given time interval into a variant of the knapsack problem to efficiently find the exact solution. We then propose a framework to systematically reduce the search space associated with finding the worst-case demand of AVR tasks. Experimental results reveal that our approach is at least 10 times faster, with an average runtime improvement of 146 times, for randomly generated tasksets when compared to the state-of-the-art technique.**

***Index Terms*—Adaptive variable rate task, demand bound function, worst-case demand, knapsack problem**

I. Introduction

Resource management is a key consideration in any real-time system. Pessimistic assumptions lead to overestimation of workload, which results in underutilization of the resources. On the other hand, workload underestimation can cause deadline misses. For a system with hard real-time requirements, missing deadlines can be catastrophic. An example is the powertrain control module (PCM) of a car. The ignition system and fuel injection tasks, which are managed by the PCM, are initiated based on the crankshaft's relative position with respect to fixed points in its path of rotation. As the crankshaft's angular speed increases, the crankshaft reaches a given angle faster, hence increasing the number of job releases in a given time interval. As a result, at higher speeds, a larger number of jobs are released, and if not properly scheduled, some jobs could miss their deadlines.

An engine's behavior is generally more stable at higher speeds due to frequent sensor and actuation updates. Hence, jobs released at higher speeds may have lower execution times [1]. To reflect this, the so-called engine-triggered tasks are modeled to have smaller execution times as the speed increases. In addition, as the speed increases, the time taken to complete a rotation decreases and hence the inter-arrival time between two consecutive jobs also decreases. Since the traditional periodic task model [2] assumes a constant time period for a task, applying it to define systems such as a vehicle with PCM would result in overly pessimistic utilization. To tackle this, a model called the adaptive variable rate (AVR) task model has been proposed to capture the behavior of engine-triggered tasks. An AVR task is defined by a set of modes, each of which is expressed by a range of speeds [1] and a constant execution time as shown in Fig. 1.

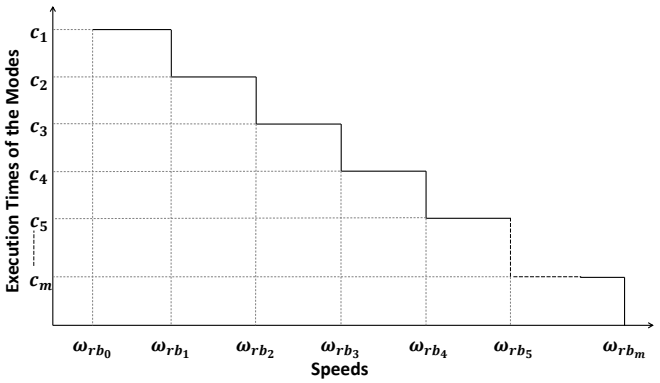


Fig. 1. Different modes of an AVR task where c_i and ω_{rb_i} are the execution time and the right boundary speed of the i^{th} mode, respectively.

To determine whether an AVR task is schedulable using EDF, the demand bound function (dbf) is often used [3], [4] to measure the resource requirement over a given time interval. In a nutshell, dbf determines the worst-case aggregate execution time of the jobs that have both the arrivals and deadlines within a time interval $[t_1, t_2]$. In general, the calculation of the worst-case demand of an AVR task is not straightforward. Let us consider an example. In a time interval $[t_1, t_2]$, assume 15 jobs are released at the highest allowable speed with each job having an execution time of 50 μs. On the other hand, during the same duration, assume 10 job releases are possible at the lowest speed with each job having an execution time of 100 μs. The demand when the jobs are released at the lowest speed (1,000 μs) is greater than when the jobs are released at the highest speed (750 μs). Suppose instead that the jobs that are released at the highest speed have an execution time of 70 μs. In this case, the demand of these jobs is greater than the demand of the jobs that are released at the lowest speed. Hence, the demand depends on the relationship between the task's execution times and speeds, as well as the acceleration profiles of the engine.

Several methods have been proposed to calculate the dbf of

an AVR task. Mohaqeqi et al. [5] proposed an exact analysis, using the Digraph model [6], to transform an AVR task into a digraph to calculate the exact worst-case demand assuming that the crankshaft can have multiple acceleration values during a rotation. While this approach represents the state-of-the-art technique, it is computationally intensive and unlikely to be suitable for large problem instances. In this paper, we propose a knapsack-based method to efficiently calculate the exact worst-case demand of an AVR task.

Contributions: The main contributions of this paper are:

1) To determine the worst-case demand of an AVR task, the search for the dominant job sequence, i.e., one that results in the maximum demand over a given time interval, is modeled as a bounded precedence constraint knapsack problem. A dynamic programming based approach is presented to exactly and efficiently solve the problem.
2) The number of job sequences that need to be considered when calculating the worst-case demand of an AVR task is significantly reduced by exploiting the kinematic properties of the engine.
3) Experimental results based on existing AVR task sets as well as randomly generated AVR task sets reveal that the proposed approach significantly outperforms the state-of-the-art technique [5] in terms of computation time.

The rest of the paper is organized as follows. In Section II, we review key existing work pertaining to AVR tasks. In Section III, we introduce the system model, discuss our assumptions and formally present the problem. In Section IV, we present a knapsack-based approach to find the worst-case demand of AVR tasks. We provide some necessary conditions to reduce the search space in Section V and describe the dominant sequence set in Section VI. Experimental results are presented in Section VII and the paper concludes in Section VIII.

II. Related Work

Usage of multiple worst-case execution times and periods for engine-controlled tasks was first studied by Kim et al. [7], where the authors proposed the rhythmic task model and obtained schedulability results assuming the dependency of a task's attributes on external physical events. However, the analysis is limited to a single AVR task scheduled along with periodic tasks using rate monotonic scheduling algorithm in which the AVR task has the highest priority.

Biondi et al. [8] presented the calculation of the worst-case demand as a search problem in the speed domain and the infinite number of paths in the search tree was narrowed down by identifying certain paths that met a given criteria. A similar method was applied using rate monotonic [3] and EDF scheduling [4]. However, these works assume a constant acceleration between two jobs releases, which does not always result in the worst-case demand, as shown by Mohaqeqi et al. [5]. In one of the first works on EDF scheduling of AVR tasks, Guo and Baruah [9] developed a sufficient schedulability test based on a speed-up factor analysis.

Identifying the fact that the exact speed of rotation of the crankshaft may not be known, Biondi et al. [10] proposed two methods to estimate the angular speed of the crankshaft. In a recent paper [11], Biondi et al. proposed a task model for expressing some practical features of engine control tasks and presented schedulability tests for engine control applications under EDF scheduling.

A complementary direction of research on AVR tasks was undertaken by Biondi et al. [12], and focused on finding the boundary speeds of the modes to maximize the performance of the engine using an optimization based approach.

Mohaqeqi et al. [5] partitioned the speed domain and constructed the corresponding digraph real-time (DRT) task graph to determine the worst-case demand of the AVR task by searching from each of the nodes of the DRT graph. Though such an approach gives the exact value of the worst-case demand of the AVR task, it considers many unnecessary paths, resulting in long computation times. In this paper, we propose an algorithm to obtain the speed partitioning and to select a smaller subset of the paths considered by Mohaqeqi et al. [5] to significantly reduce the computation time.

III. Preliminaries

In this section, we provide some background materials on the engine and its properties and introduce our task model. We also formally define the problem.

A. Task Model

Adaptive variable rate (AVR) tasks are triggered at certain angles with respect to the top dead center position of the crankshaft, unlike periodic tasks which release jobs at regular time intervals. For example, consider the different stages of fuel ignition in a vehicle as shown in Fig. 2. For optimal performance of the engine, fuel injection should occur at a precise angle. As the rate of arrival of the crankshaft at a given angle varies with its angular speed, AVR tasks do not have a fixed period. Rather, at a higher speed, a larger number of instances (i.e., jobs) of each task occur, potentially increasing the resource requirement.

While it is possible to determine the schedulability of a task set assuming that this increased resource requirement of an AVR task is its steady-state demand, doing so would lead to pessimistic analyses and hence resource underutilization. Moreover, the engine is more stable at higher speeds [1]. This allows jobs to have shorter execution times at higher crankshaft speeds. Hence, the execution time of AVR tasks is modeled as a function of the speed at which the jobs are released, as shown in Fig. 1.

The range of speeds in which the execution time is constant is referred to as a mode and the edge speeds of the mode are referred to as the *boundary speeds* of the mode. The set of speeds where the mode changes are defined as the set of right boundary speeds, i.e., $\mathbf{\Omega_{rb}} = \{\omega_{rb_1}, \ldots, \omega_{rb_m}\}$, $\forall \omega \in (\omega_{rb_{i-1}}, \omega_{rb_i}]$, speed ω is in the i^{th} mode. We further define $\mathbf{\Omega_{rb}}(\omega)$ to be all the right boundary speeds larger than ω that are reachable (defined later in Section III). The minimum and

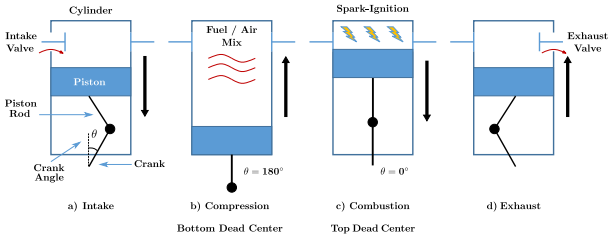
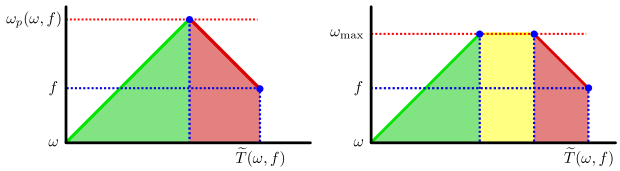


Fig. 2. Different stages of fuel ignition in a vehicle.



(a) $\widetilde{T}(\omega, f)|\omega_p(\omega, f) \leq \omega_{\max}$ (b) $\widetilde{T}(\omega, f)|\omega_p(\omega, f) > \omega_{\max}$

Fig. 3. Minimum interarrival time $\widetilde{T}(\omega, f)$ between speeds ω and f where (a) $\omega_p(\omega, f) \leq \omega_{\max}$ and (b) $\omega_p(\omega, f) > \omega_{\max}$. Ascending, flat, and descending lines represent periods of maximum ($\alpha_{\max}$), zero, and minimum ($\alpha_{\min}$) acceleration, respectively.

maximum allowable speeds of rotation of the crankshaft are represented by ω_{rb_0} and ω_{rb_m} respectively. For convenience, we refer to them as $\omega_{\min}$ and $\omega_{\max}$, respectively and assume $\omega \geq 0, \forall \omega \in [\omega_{\min}, \omega_{\max}]$, where ω is assumed to be in rev/min. In addition, the instantaneous speed at a given time t is denoted as $\omega(t)$. A table of notations can be found in Appendix B.

The maximum allowable acceleration and deceleration are denoted by $\alpha_{\max}$ and $\alpha_{\min}$, respectively, both assumed to be in rev/min^2. In this paper, we assume $\alpha_{max} = |\alpha_{min}|$. The execution time of the i^{th} mode is represented by c_i. Additionally, the execution time corresponding to a speed $\omega(t)$, is denoted by $c(\omega(t))$. We assume that a job is released at the top dead center position, which we consider as the beginning of the rotation. Thus, a job's execution time is determined by the speed of the crankshaft at the beginning of its the rotation.

Property 1 (Speed After n Rotations): Given an initial speed of $\omega(t)$ at time t and a constant acceleration α, the speed after an angular displacement of $\Delta\theta$ is [8], [13]

$$\Omega(\omega(t), \alpha, \Delta\theta) = \sqrt{\omega(t)^2 + 2\alpha\Delta\theta}. \tag{1}$$

Similar to the work by Mohaqeqi et al. [5] we assume that $\Delta\theta$ specifies the crankshaft revolution in terms of the number of rotations, i.e., $\Delta\theta = 1$ indicates a complete rotation. Hence, according to Equation 1, assuming an initial speed of $\omega(t)$ at time t, and a constant acceleration of α, the speed after one complete rotation is $\Omega_1(\omega(t), \alpha) = \sqrt{\omega(t)^2 + 2\alpha}$. In general, the speed after n complete rotations is [5],

$$\Omega_n(\omega(t), \alpha) = \sqrt{\omega(t)^2 + 2n\alpha}. \tag{2}$$

Biondi et al. [3] showed that multiple AVR tasks activated by the same source and, which have the same angular phase and period can be modeled as a single AVR task, called the representative AVR task. Hence, the analysis in this paper can also be extended to multiple AVR tasks.

B. Minimum Job Inter-arrival Times

The minimum inter-arrival time is the minimum time duration from the release of a job at t_1 when the speed is $\omega(t_1)$ to the next job release at t_2 and speed $\omega(t_2)$. We denote minimum inter-arrival time by $\widetilde{T}(\omega(t_1), \omega(t_2))$. In order to overcome the drawback of using constant acceleration between job releases as was assumed in most existing work [3], [4], [8], we consider the possibility of acceleration variations between two job releases similar to the work by Mohaqeqi et al. [5].

The minimum inter-arrival time equation by Mohaqeqi et al. [5] is briefly presented here using simplified notation for readability. To get the minimum inter-arrival time from any speed ω, the crankshaft has to be maximally accelerated from ω to reach ω_p, the peak speed, and then maximally decelerated to reach a target speed f in a single rotation,

$$\omega_p(\omega, f) = \frac{\sqrt{2\omega^2 + 2f^2 + 2\alpha_{\max}}}{2}. \tag{3}$$

However, if $\omega_p > \omega_{max}$, the crankshaft has to maximally accelerate from ω to ω_{max}, stay at that speed for some time and then maximally decelerate to f. The two cases are defined below and presented in Figure 3:

$$\widetilde{T}(\omega, f) = \tag{4}$$
$$\begin{cases} \frac{\sqrt{2\omega^2+2f^2+4\alpha_{\max}}-\omega-f}{\alpha_{\max}} & \omega_p(w, f) \leq \omega_{\max} \\ \frac{\omega_{max}-f-\omega}{\alpha_{\max}} + \frac{\omega^2+f^2}{2\omega_{\max}\alpha_{\max}} + \frac{1}{\omega_{\max}} & \omega_p(w, f) > \omega_{\max} \end{cases}$$

Property 2 (Reversability of Inter-Arrival Times): $\widetilde{T}(\omega(t_1), \omega(t_2)) = \widetilde{T}(\omega(t_2), \omega(t_1))$. In other words, the minimum inter-arrival time from a job released at $\omega(t_1)$ to a job released at $\omega(t_2)$ is equal to the inter-arrival time when the speeds are in the reverse order.

C. AVR Task Demand

Let $\mathbb{W}$ be the set of all possible speed functions $\omega(t)$ that are feasible (i.e., $\omega(t)$ is any continuous function with acceleration between $\alpha_{\min}$ and $\alpha_{\max}$ and speeds between $\omega_{\min}$ and $\omega_{\max}$). For any such $\omega(t)$, considering that a computational job of an AVR task is released at time t', we assume that the processor must successfully complete execution of this job by time $t' + \widetilde{T}(\omega(t'), \min(\Omega_1(\omega(t'), \alpha_{\max}), \omega_{max}))$; that is, the absolute deadline of this job coincides with the minimum time to complete a single rotation from a given speed $\omega(t')$. We refer to an AVR task that sets deadlines in this way as a **minimum angular deadline AVR task** [11] and refer to the relative deadline of a job released at speed ω as $\tilde{d}(\omega) = \widetilde{T}(\omega(t'), \min(\Omega_1(\omega(t'), \alpha_{\max}), \omega_{max}))$. We may denote the *demand* of $\omega(t)$ over any δ-length interval $[t_a, t_b]$ as $D_{\omega(t)}(t_a, t_b)$. $D_{\omega(t)}(t_a, t_b)$ represents the total execution

requirement of jobs released by the speed function with both arrival times and absolute deadline in the interval $[t_a, t_b]$. More formally, if $\{t_1, t_2, \ldots\}$ is the set of times (in order) where the crankshaft triggers job releases following the speed function $\omega(t)$, then:

$$D_{\omega(t)}(t_a, t_b) = \sum_{i:(t_i \geq t_a) \wedge (t_i + \tilde{d}(\omega(t_i)) \leq t_b)} c(\omega(t_i)). \quad (5)$$

We can compute the upper envelope on the demand over any interval of length $\delta > 0$ for $\omega(t)$ as:

$$\mathsf{dbf}_{\omega(t)}(\delta) = \max_{t' \geq 0} \{D_{\omega(t)}(t', t' + \delta)\}. \quad (6)$$

The *worst-case demand* of an AVR task over any δ-length interval is the speed schedule that maximizes the value of the upper envelope given in Equation 6:

$$\mathsf{dbf}(\delta) = \max_{\omega(t) \in \mathbb{W}} \{\mathsf{dbf}_{\omega(t)}(\delta)\}. \quad (7)$$

Considering any speed function $\omega(t)$ with job releases at speeds/times $\omega(t_1), \omega(t_2), \ldots$ in some interval $[t_a, t_b]$, it is clear that if we modify the function so that the crankshaft traverses one rotation in the minimum time (by Equation 4), then this will only lead to demand that exceeds or equals the original demand of $\omega(t)$. Therefore, without loss of generality, we can restrict $\mathbb{W}$ to speed functions that traverse the rotation between job releases in the minimum possible time. This is essentially the observation made by Mohaqeqi et al. [5] to reduce the problem of finding the worst-case $\omega(t) \in \mathbb{W}$ to the problem of identifying *sequences of job release speeds*; i.e., the speed of the job releases entirely characterizes the speed function. That is, we can completely describe any speed function $\omega(t)$ with job releases at times $t_1, t_2, \ldots$ instead by a sequence of speeds $(\omega_1 = \omega(t_1), \omega_2 = \omega(t_2), \ldots)$. Thus, after this point, we drop the speed-time function $\omega(t)$ and focus only on sequences of speeds.

The objective of this paper is to find an efficient way to calculate the worst-case AVR task demand defined in Equation 7.

Definition 1 (Reachable Speeds): Consider two speeds ω_1 and ω_2 such that $\omega_2 \geq \omega_1$. ω_2 is said to be *reachable* from ω_1 if $\Omega_1(\omega_1, \alpha_{max}) \geq \omega_2$.

Definition 2 (Valid Sequence): A set of jobs $j_1, j_2, j_3, \ldots, j_k$ released at speeds $\omega_1, \omega_2, \ldots, \omega_k$ is said to be a valid sequence of speeds if $\forall i \in \mathbb{N}_2^k$, ω_i is reachable from ω_{i-1} where $\mathbb{N}_j^k$ is the set of natural numbers from *j* to *k*, i.e., $\{j, j+1, \ldots, k\}$.

D. Problem Definition

Consider a minimum angular deadline AVR task, which is characterized by feasible speed $[\omega_{\min}, \omega_{\max}]$ and acceleration $[\alpha_{min}, \alpha_{max}]$ ranges, where $\alpha_{max} = |\alpha_{min}|$, and a set of modes $\mathbf{\Omega_{rb}} = \{\omega_{rb_1}, \ldots, \omega_{rb_m}\}$, each of which is associated with an execution time $c(\omega_{rb_i})$, $i = 1, \ldots, m$, as discussed earlier in this section. The objective is to find $\mathsf{dbf}(\delta)$ (Equation 7), the worst-case demand of the AVR task over any δ-length interval $[t_a, t_b]$ assuming that the acceleration of the crankshaft may change within a single rotation.

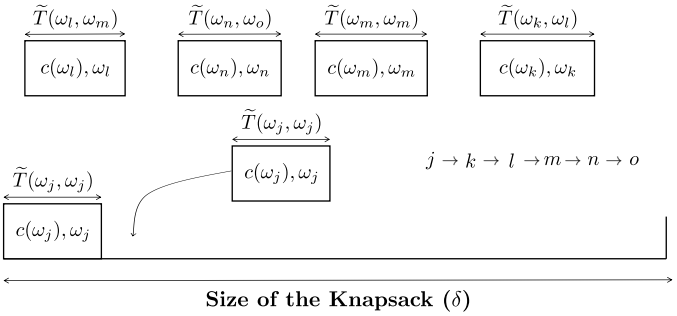


Fig. 4. Example items for our knapsack problem. A job that is higher in the precedence relation is preceded by a job lower in the relation.

IV. Knapsack-based approach for deriving the worst-case demand

We now describe the problem transformation from calculating $\mathsf{dbf}(\delta)$ to a variant of the knapsack problem. This section both provides context for and relies upon several properties and lemmas defined in Sections V and VI. Briefly, *dominant speed sequences* are sequences of job release speeds whose demand coincides with the value of Equation 7. These dominant speed sequences are non-decreasing (see Section V-B, Lemma 1) and start at right boundary speeds (see Section V-C, Lemma 2).

In the traditional knapsack problem, the aim is to maximize the total profit from a given set of items where each item is associated with a profit and weight, while ensuring that the aggregate weight is less than or equal to the maximum allowable weight of the knapsack. Our goal is to transform the problem of finding $\mathsf{dbf}(\delta)$ into a variant of the knapsack problem. In our case, a job is equivalent to an item. As such, the "weight" of a job (i.e., item) is the minimum inter-arrival time, and the "profit" is its execution time. The goal, then, is to maximize demand (i.e., profit) over a time interval (δ).

Since a dominant speed sequence is non-decreasing (Lemma 1), a job's execution time may contribute to the demand bound function $\mathsf{dbf}(\delta)$ more than once. As such, our knapsack problem is, in fact, a bounded knapsack problem. In addition, since adjacent speeds in a dominant speed sequence must be reachable from one another (Definition 1), once an item, i.e., job, has been included in the knapsack, there is a finite number of jobs that can follow, making our problem a bounded precedence constraint knapsack problem (BPCKP) [14], [15]. Fig. 4 provides a visual example representation of our problem as a BPCKP. For clarity, the subscripts of the speeds are used to denote the precedence constraints.

An effective way to model the precedence constraints among jobs is by using out-trees. The sequences beginning at each of the source nodes are independent of each other (i.e., sequences beginning at each of the right boundary speeds according to Lemma 2). An example out-tree is shown in Fig. 5. To represent the trees in a knapsack problem, we define a dependency graph $G_I = (V_I, A_I)$, where V_I denotes the set of vertices (i.e., items) in the out-tree G_I [16]. An edge between any two vertices denotes that the two speeds are reachable

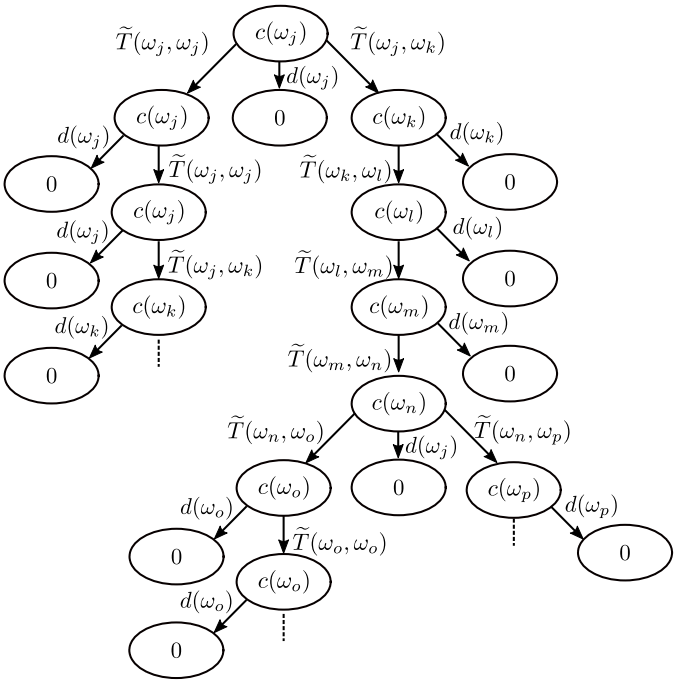


Fig. 5. Precedence constraints among jobs are expressed using out-trees. Nodes represent WCETs for the initial speeds and arrows represent minimum inter-arrival times. The tree demonstrates the various precedence relations and possible paths. Leaves represent completion of the parent job without adding subsequent jobs (i.e. items). Furthermore, each of the leaves has zero execution since its parent is the final job included in the knapsack.

from one another and is represented by $(j,k) \in A_I$. Formally, our BPCKP can be expressed as follows:

$$\underset{x}{\text{maximize}} \quad \sum_{j} \sum_{r=1}^{M_\delta} c(\omega_j) \cdot x_j^r \tag{8}$$

$$\text{subject to} \quad \sum_{j,k|(j,k)\in A_I} \sum_{r=1}^{M_\delta} \tilde{T}(w_j, w_k) \cdot x_k^r \leq \delta \tag{9}$$

$$\sum_{k|(j,k)\in A_I} x_k^1 \leq 1, \, \forall j \tag{10}$$

$$x_j^1 \geq x_k^1, \, \forall (j,k) \in A_I \tag{11}$$

$$x_j^r \geq x_j^{r+1}, \; \forall j, r \in \{1,2,3,...,M_\delta - 1\} \tag{12}$$

$$x_j^r \in \{0,1\}, \, \forall j, r \in \{1,2,3,...,M_\delta\} \tag{13}$$

where M_δ represents an upper bound on the number of job releases in a δ-length interval and x_j^r is a binary variable: x_j^r is one if there are at least r jobs released at speed ω_j in the knapsack; otherwise, x_j^r is zero. Equation 8 maximizes total demand. Equation 9 requires all deadlines of selected jobs fall within the δ-length interval. Equation 10 permits at most one child node of each parent node to be added to the knapsack. Equation 11 enforces the precedence constraint such that no child node may be added without its parent. Equation 12 ensures repeated jobs of a particular speed ω_j are added incrementally to (and do not skip indices of) x_j^r (i.e, if $x_j^\ell = 0$, then for all $s > \ell : x_j^s = 0$).

Algorithm 1 shows our pseudocode for the dynamic programming approach to solve the bounded precedence constraint knapsack problem. The *CalculateDemand* function is initially called with the parameters $\omega \in \mathbf{\Omega_{rb}}$ and δ, the total time length. The *maxDemand* parameter keeps track of the highest demand computed until the current instance of the recursive loop. In each recursive instance, the next speed is chosen from the list of possible next job release speeds according to Theorem 1 (Section VI). The variable D_w (Equation 5) tracks the accumulated demand of the current sequence.

Algorithm 1 DP for Calculating $\mathsf{dbf}(\delta)$

```
1:  function CalculateDemand(ω,δ):
2:      maxDemand ← 0
3:
4:      if StoredDemand(ω, t) ≠ φ then
5:          return StoredDemand(ω,δ)
6:
7:      if δ < d̃(ω) then
8:          return 0
9:
10:     for ω′ in nextPossibleSpeed(ω) do ▷ See Theorem 1
11:         δ ← δ − T̃(ω, ω′)
12:         D_w ← c(ω)+ CalculateDemand(ω′,δ)
13:
14:         if D_w > maxDemand then
15:             maxDemand ← D_w
16:
17:     StoredDemand(ω,δ) ← maxDemand
18:     return maxDemand
```

V. Dominant Speed Sequences

The previous section outlined how dominant speed sequences facilitate the problem transformation from calculating $\mathsf{dbf}(\delta)$ to a variant of the knapsack problem. This section formally defines Lemma 1 and Lemma 2 referenced in the above BPCKP approach.

The main challenge in determining the worst-case demand of an AVR task is the variation in the execution time, which varies as a function of the angular speed. Earlier work, e.g., Mohaqeqi et al. [5] showed that the speed sequences that maximize demand contain only speeds from some finite set. Mohaqeqi et al. used this set to design a DRT task which considers all possible feasible sequences of speeds. However, this can still lead to a large number of speed permutations to check when searching for the worst-case demand. As mentioned in the related work section, we show that we can greatly limit the sequences that need to be considered in the *dominant speed sequence set* when we consider minimum angular deadline AVR tasks with $\alpha_{max} = |\alpha_{min}|$. A *dominant speed sequence* is a speed sequence whose demand is equivalent to the maximum demand of a task over a given interval length (i.e., its demand coincides with the value of Equation 7). A *dominant speed sequence set* is a set of speed

sequences that must contain the dominant speed sequence. In this section, we derive the necessary characteristics of a dominant speed sequence.

A. Properties of Minimum Inter-arrival Times and Deadlines

We begin by establishing some useful properties regarding the minimum inter-arrival time function $\widetilde{T}(\omega, f)$ (Equation 4) that will be used to prove some characteristics of dominant speed sequences. The first property is that $\widetilde{T}(\omega, f)$ is always positive, which was already proved in previous work on AVR tasks [5]. We abuse terminology and refer to some lemmas as properties to make supporting concepts easier to read.

Property 3 (Positive Minimum Inter-arrival Times): For all $\omega, f \in [\omega_{\min}, \omega_{\max}]$,

$$\widetilde{T}(\omega, f) > 0. \tag{14}$$

We next show that $\widetilde{T}(\omega, f)$ is always non-increasing as we increase either the starting speed ω or the ending speed f.

Property 4 (Minimum Inter-arrival Time Decreases with Starting/Ending Speeds): For all $\omega, f \in [\omega_{\min}, \omega_{\max}]$,

$$\frac{\partial \widetilde{T}(\omega, f)}{\partial \omega} \leq 0 \quad \text{and} \quad \frac{\partial \widetilde{T}(\omega, f)}{\partial f} \leq 0. \tag{15}$$

Proof: Observe that the partial derivative of $\widetilde{T}$ with respect to ω and f, respectively are:

$$\frac{\partial \widetilde{T}(\omega, f)}{\partial \omega} = \begin{cases} \frac{\frac{2\omega}{\sqrt{4\alpha_{\max}+2f^2+2\omega^2}}-1}{\alpha_{\max}} & \text{if } \omega_p(w, f) \leq \omega_{\max} \\ \frac{\omega}{\omega_{\max}\alpha_{\max}} - \frac{1}{\alpha_{\max}} & \text{if } \omega_p(w, f) > \omega_{\max} \end{cases} \tag{16}$$

$$\frac{\partial \widetilde{T}(\omega, f)}{\partial f} = \begin{cases} \frac{\frac{2f}{\sqrt{4\alpha_{\max}+2f^2+2\omega^2}}-1}{\alpha_{\max}} & \omega_p(w, f) \leq \omega_{\max} \\ \frac{f}{\omega_{\max}\alpha_{\max}} - \frac{1}{\alpha_{\max}} & \omega_p(w, f) > \omega_{\max} \end{cases} \tag{17}$$

The above partial derivatives are clearly always non-positive for any $\omega, f \in [\omega_{\min}, \omega_{\max}]$. ∎

We now focus upon the relative deadline of a job released at speed ω (i.e., $\tilde{d}(\omega)$). We can show that the relative deadline decreases as we increase the speed the job is released at. Furthermore, the rate of decrease for the relative deadline of a job is faster than the rate of decrease in the minimum inter-arrival time of the next job.

Property 5 (Rate of Change of Relative Deadline): For all $\omega, f \in [\omega_{\min}, \omega_{\max}]$ where f is reachable from ω:

$$\frac{\partial \tilde{d}(\omega)}{\partial \omega} < 0 \quad \text{and} \quad \frac{\partial \tilde{d}(\omega)}{\partial \omega} \leq \frac{\partial \widetilde{T}(\omega, f)}{\partial \omega}. \tag{18}$$

Proof: First, consider the partial derivative of $\tilde{d}(\omega)$, given below. It is clear from the expression that it is always negative for all $\omega \in [\omega_{\min}, \omega_{\max}]$.

$$\tilde{d}(\omega) = \begin{cases} \frac{\sqrt{\omega^2+2\alpha_{\max}}-\omega}{\alpha_{\max}} & \tilde{d}_p(w) \leq \omega_{\max} \\ -\frac{\omega}{\alpha_{\max}} + \frac{\omega^2+\omega_{\max}^2}{2\omega_{\max}\alpha_{\max}} + \frac{1}{\omega_{\max}} & \tilde{d}_p(w) > \omega_{\max} \end{cases} \tag{19}$$

$$\tilde{d}_p(\omega) = \sqrt{\omega^2 + 2\alpha_{\max}\,\theta} \tag{20}$$

where $\tilde{d}(\omega)$ is derived from Equation 1 such that $\tilde{d}(\omega) = \frac{\Omega_1(\omega, \alpha_{\max})-\omega}{\alpha_{\max}}$ if $\tilde{d}_p(w) \leq \omega_{\max}$ and $\tilde{d}(\omega) = \widetilde{T}(\omega, \omega_{\max})$ if $\tilde{d}_p(w) > \omega_{\max}$.

$$\frac{\partial \tilde{d}(\omega)}{\partial \omega} = \begin{cases} \frac{\frac{\omega}{\sqrt{\omega^2+2\alpha_{\max}}}-1}{\alpha_{\max}} & \tilde{d}_p(w, f) \leq \omega_{\max} \\ \frac{\omega}{\omega_{\max}\alpha_{\max}} - \frac{1}{\alpha_{\max}} & \tilde{d}_p(w, f) > \omega_{\max} \end{cases} \tag{21}$$

where $\tilde{d}$ is derived by replacing f with Equation 1 in the definition of minimum angular deadline AVR task when $\tilde{d}_p(w) < \omega_{\max}$. By supposition, f is reachable from ω; thus by definition of reachable,

$$\begin{aligned} & f \leq \Omega_1(\omega, \alpha_{\max}) \\ \Leftrightarrow \quad & f \leq \sqrt{\omega^2 + 2\alpha_{\max}} \\ \Leftrightarrow \quad & 2f^2 \leq 2\omega^2 + 4\alpha_{\max} \\ \Leftrightarrow \quad & 4\alpha_{\max} + 2f^2 + 2\omega^2 \leq 4\omega^2 + 8\alpha_{\max} \\ \Leftrightarrow \quad & \omega\sqrt{4\alpha_{\max} + 2f^2 + 2\omega^2} \leq 2\omega\sqrt{\omega^2 + 2\alpha_{\max}} \\ \Leftrightarrow \quad & \frac{\frac{\omega}{\sqrt{\omega^2+2\alpha_{\max}}}-1}{\alpha_{\max}} \leq \frac{\frac{2\omega}{\sqrt{4\alpha_{\max}+2f^2+2\omega^2}}-1}{\alpha_{\max}} \\ \Leftrightarrow \quad & \frac{\partial \tilde{d}(\omega)}{\partial \omega} \leq \frac{\partial \widetilde{T}(\omega, f)}{\partial \omega} \end{aligned}$$

∎

B. Speed Sequence Order Transformations

In this subsection, we describe how given a valid speed sequence $S = (\omega_1, \omega_2, \ldots, \omega_n)$, we may transform it to one in non-decreasing order without reducing the total demand of the sequence. We begin with some notation that will be employed in describing the transformations.

Definition 3 (Non-Decreasing Speed Sequence): Given any valid, finite speed sequence $S = \{\omega_1, \omega_2, \ldots, \omega_n\}$, we define $S_A = (s_1, s_2, \ldots, s_n)$ to be the sequence obtained from reordering the speeds of S in a non-decreasing order (i.e., s_1 is the smallest ω_i in S and s_n is the largest). We call S_A a *non-decreasing speed sequence* of S.

We can show that for any valid sequence S (in arbitrary speed order) the corresponding non-decreasing sequence S_A must also be valid.

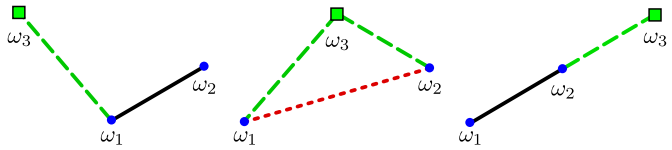
Property 6 (Validity of Non-Decreasing Sequences): If $S = (\omega_1, \omega_2, \ldots, \omega_n)$ is a valid sequence, then the non-decreasing sequence S_A is also valid.

Proof: For the sake of contradiction, assume that S is valid, but S_A is not. That means there exists some $s_i \in S_A$ such that $s_{i+1} > \Omega_1(s_i, \alpha_{\max})$. Furthermore, this also implies that the following must be true $\forall\, \ell,\, k \,|\, 1 \leq \ell \leq i < k \leq n$:

$$s_k > \Omega_1(s_\ell, \alpha_{\max}). \tag{22}$$

This is to say that the ascending sequence, S_A, is split between indices s_i and s_{i+1} which are not reachable from one another such that it is impossible to reach speed s_{i+1} or higher from any speed s_i or lower. However, this contradicts the validity of S. Since S_A has non-decreasing order, no rearrangement of S_A will make the speeds any closer to one another and therefore will not make previously unreachable speeds reachable. Thus, S is also invalid. ∎

We now provide some definitions that will be used to compare S and S_A.



(a) Leading Injection (b) Internal Injection (c) Final Injection

Fig. 6. A two-speed sequence, ω_1, ω_2, shown as dots, receives the (a) leading, (b) internal, and (c) final, injection of ω_3 shown as a square. Dashed, dotted, and solid lines represent the added, removed, and unchanged minimum inter-arrival times, respectively.

Definition 4 (k-Subsequence of S): Given any valid, finite speed sequence $S = \{\omega_1, \omega_2, \ldots, \omega_n\}$, for any $k \in \mathbb{N}_1^n$, we define $S(k) = (\omega_1^{(k)}, \omega_2^{(k)}, \ldots, \omega_k^{(k)})$ to be the k-subsequence of S containing the k smallest elements of S in the same order that they originally appear in S. (For example, $S = (4, 3, 1)$ would have $S(2) = (3, 1)$). Note that a k-subsequence of a valid subsequence must also be valid itself. Similarly, $S_A(k)$ has the k^{th} smallest elements of S in non-decreasing order.

Definition 5 (Injection into a Subsequence): Given a k-subsequence $S(k)$ of an original sequence S, we consider the addition of the $(k+1)^{th}$ smallest element of S into $S(k)$ to create $S(k+1)$. We categorize the three possible injection types as follows:

1) *Leading Injection*: If ω is the $(k+1)^{th}$ smallest item of S and it becomes the first element of $S(k+1)$. That is, $\omega_1^{(k+1)}$ equals ω and $\omega_{\ell+1}^{(k+1)}$ equals $\omega_\ell^{(k)}$ for all $\ell = 1, \ldots, k$. Figure 6(a) shows an example leading injection.
2) *Internal Injection*: If ω is the $(k+1)^{th}$ smallest item of S and it is neither the first nor last element of $S(k+1)$. That is, there exists some $j \in \mathbb{N}_2^k$ such that $\omega_j^{(k+1)}$ equals ω and $\omega_\ell^{(k+1)}$ equals $\omega_\ell^{(k)}$ for all $\ell = 1, \ldots, j-1$ and $\omega_{\ell+1}^{(k+1)}$ equals $\omega_\ell^{(k)}$ for all $\ell = j, \ldots, k$. Figure 6(b) shows an example internal injection.
3) *Final Injection*: If ω is the $(k+1)^{th}$ smallest item of S and it becomes the last item of $S(k+1)$. That is, $\omega_{k+1}^{(k+1)}$ equals ω and $\omega_\ell^{(k+1)}$ equals $\omega_\ell^{(k)}$ for all $\ell = 1, \ldots, k$. Figure 6(c) shows an example final injection.

Note that by definition, final injections of s_{k+1} into $S_A(k)$ will maintain the non-decreasing property of $S_A(k)$. We will refer to this later when constructing dominant sequences.

To compare the demand produced by two different sequences (with the same elements, but in different order), we will look at the time of the absolute deadline of the last job in the sequence under the assumptions that jobs of the sequence arrive according to their minimum interarrival time (i.e., $\widetilde{T}$) and the first job is released at time instant zero. Let $d(S)$ be the last absolute deadline for sequence $S = (\omega_1, \ldots, \omega_n)$:

$$d(S) = \sum_{i=1}^{n-1} \widetilde{T}(\omega_i, \omega_{i+1}) + \tilde{d}(\omega_n). \tag{23}$$

We can quantify how the $d(S)$ function changes as we inject elements of non-decreasing speed. Let $\Delta(S, k, k+1)$ represent the amount that the d function increases when injecting the $(k+1)^{th}$ smallest element (s_{k+1}) into $S(k)$. That is,

$$\Delta(S, k, k+1) = d(S(k+1)) - d(S(k)). \tag{24}$$

We can compute the above difference based on the type of injection that adding the $(k+1)^{th}$ smallest element to $S(k)$ results in:

$$\Delta(S, k, k+1) = \begin{cases} \Delta_L(S, k, k+1) & \text{if leading,} \\ \Delta_F(S, k, k+1) & \text{if final,} \\ \Delta_I(S, k, k+1) & \text{if internal,} \end{cases} \tag{25}$$

where $\Delta_L(S, k, k+1) = \widetilde{T}(s_{k+1}, \omega_1^{(k)})$ since we are adding one segment to the front of $S(k)$ in the leading injection; $\Delta_F(S, k, k+1) = \widetilde{T}(\omega_k^{(k)}, s_{k+1}) + \tilde{d}(s_{k+1}) - \tilde{d}(\omega_k^{(k)})$ since we are adding one segment to the end of $S(k)$ and adjusting the deadline of the last job for the new speed s_{k+1} for a final injection; and $\Delta_I(S, k, k+1) = \widetilde{T}(\omega_{j-1}^{(k)}, s_{k+1}) + \widetilde{T}(s_{k+1}, \omega_j^{(k)}) - \widetilde{T}(\omega_{j-1}^{(k)}, \omega_j^{(k)})$ if we inject s_{k+1} into the j^{th} position of $S(k)$, $j \in \mathbb{N}_2^k$.

We are now prepared to compare the amount of time added to the absolute deadline of the last job of the sequence, considering k-subsequences S and S_A and injecting the $(k+1)^{th}$ smallest element (s_{k+1}) into both of these sequences for each of the three types of injections.

Property 7 (Leading Injection Suboptimality): For any valid, finite sequence $S = (\omega_1, \ldots, \omega_n)$, for all $k = 1, \ldots, n-1$:

$$\Delta_L(S, k, k+1) \geq \Delta_F(S_A, k, k+1). \tag{26}$$

Proof: In this property, we have a leading injection into $S(k)$. However, observe that $\tilde{d}(s_{k+1}) \leq \tilde{d}(s_k)$ by Property 5 and since s_{k+1} is larger than any element of $S(k)$. Furthermore, $\widetilde{T}(\omega_k^{(k)}, s_{k+1}) \geq \widetilde{T}(s_k, s_{k+1})$ since, by Property 4, $\widetilde{T}$ is a decreasing function in its parameters and s_k is at least as large as any item in $S(k)$. Also, by Property 2, $\widetilde{T}(\omega_k^{(k)}, s_{k+1})$ equals $\widetilde{T}(s_{k+1}, \omega_k^{(k)})$. By the above observations, we get:

$$\begin{aligned} & \widetilde{T}(s_{k+1}, \omega_k^{(k)}) + \tilde{d}(s_k) \geq \widetilde{T}(s_k, s_{k+1}) + \tilde{d}(s_{k+1}) \\ \Leftrightarrow \quad & \widetilde{T}(s_{k+1}, \omega_k^{(k)}) \geq \widetilde{T}(s_k, s_{k+1}) + \tilde{d}(s_{k+1}) - \tilde{d}(s_k) \end{aligned}$$

The LHS and RHS of the last inequality matches Equation 26 and the lemma is proved. ∎

Property 8 (Internal Injection Suboptimality): For any valid, finite sequence $S = (\omega_1, \ldots, \omega_n)$, for all $k = 1, \ldots, n-1$:

$$\Delta_I(S, k, k+1) \geq \Delta_F(S_A, k, k+1). \tag{27}$$

Proof: Observe that by Properties 2 and 5, $\widetilde{T}(f, \omega+\epsilon) \geq \tilde{d}(\omega+\epsilon)$ for all f, ω and $\epsilon > 0$. Also, $\widetilde{T}(s_k+\epsilon, \omega) = \widetilde{T}(\omega, s_k+\epsilon)$ for all ω and $\epsilon > 0$ by Property 2. These properties imply that:

$$\begin{aligned} & \widetilde{T}(s_{k-1}, s_k) + \widetilde{T}(s_k, s_k) + \tilde{d}(s_k) \\ & \quad = \widetilde{T}(s_{k-1}, s_k) + \widetilde{T}(s_k, s_k) + \tilde{d}(s_k) \\ & \widetilde{T}(s_{k-1}, s_k + \epsilon) + \widetilde{T}(s_k + \epsilon, s_k) + \tilde{d}(s_k) \\ \Rightarrow \quad & \quad \geq \widetilde{T}(s_{k-1}, s_k) + \widetilde{T}(s_k, s_k + \epsilon) + \tilde{d}(s_k + \epsilon) \end{aligned} \tag{28}$$

Setting $\epsilon = s_{k+1} - s_k$ and substituting into the above inequality of Equation 28, we get the following:

$$\begin{aligned} & \widetilde{T}(s_{k-1}, s_{k+1}) + \widetilde{T}(s_{k+1}, s_k) + \tilde{d}(s_k) \\ & \quad \geq \widetilde{T}(s_{k-1}, s_k) + \widetilde{T}(s_k, s_{k+1}) + \tilde{d}(s_{k+1}) \\ \Rightarrow \quad & \widetilde{T}(s_{k-1}, s_{k+1}) + \widetilde{T}(s_{k+1}, s_k) - \widetilde{T}(s_{k-1}, s_k) \\ & \quad \geq \widetilde{T}(s_k, s_{k+1}) + \tilde{d}(s_{k+1}) - \tilde{d}(s_k) \\ \Rightarrow \quad & \widetilde{T}(\omega_{j-1}^{(k)}, s_{k+1}) + \widetilde{T}(s_{k+1}, \omega_j^{(k)}) - \widetilde{T}(\omega_{j-1}^{(k)}, \omega_j^{(k)}) \\ & \quad \geq \widetilde{T}(s_k, s_{k+1}) + \tilde{d}(s_{k+1}) - \tilde{d}(s_k) \end{aligned} \tag{29}$$

The last inequality (which implies Equation 27 of the property) above follows from observing that according to Property 4, the following is true for all ω and ω':

$$\begin{aligned} & \frac{\partial \widetilde{T}(\omega, s_{k+1})}{\partial \omega} \leq \frac{\partial \widetilde{T}(\omega, \omega')}{\partial \omega} \\ \Leftrightarrow \quad & \frac{\partial \widetilde{T}(\omega, s_{k+1})}{\partial \omega} + \frac{\partial \widetilde{T}(s_{k+1}, \omega')}{\partial \omega} - \frac{\partial \widetilde{T}(\omega, \omega')}{\partial \omega} \leq 0 \end{aligned}$$

Thus, the LHS of the second inequality of Equation 29 will not decrease if we substitute $\omega_{j-1}^{(k)}$ for s_{k-1} in the LHS. For symmetric reasons, we can also substitute $\omega_j^{(k)}$ for s_k. ■

In this next property, we show that any sequence may decrease the time of its last deadline by moving the highest speed job to the end (if it is valid). The proof of this property is in the appendix.

Property 9 (Highest-Speed Relative-Deadline Dominance): For any valid, finite sequence $S = (\omega_1, \ldots, \omega_n)$, if ω_ℓ ($\ell \in \mathbb{N}_1^n - 1$) is the highest speed s_n and not the last speed of sequence S and $s_n \leq \Omega_1(\omega_n, \alpha_{\max})$ then new sequence S' with the highest element moved to the last element (i.e., $S' = (\omega_1, \ldots, \omega_{\ell-1}, \omega_{\ell+1}, \ldots, \omega_n, s_n)$) is valid and has the following property:

$$d(S) \geq d(S'). \tag{30}$$

We are now ready to state the main lemma of this subsection. That is, for any dominant speed sequence, we can find another dominant speed sequence with equivalent demand that is in ascending order.

Lemma 1 (Dominant Non-Decreasing Speed Sequences): Given a valid, dominant speed sequence $S = (\omega_1, \omega_2, \ldots, \omega_n)$ over interval $[t_a, t_b]$, the sequence S_A is valid and has equivalent demand.

Proof: The proof is by induction on k. For each $k \in \mathbb{N}_1^n$, we show that for any k-subsequence $S'(k)$ containing the same elements of $S(k)$ such that $d(S(k)) \geq d(S'(k))$, the following is true:

$$d(S(k)) \geq d(S'(k)) \geq d(S_A(k)). \tag{31}$$

This implies the lemma, since if the deadline of the last job of S is before t_b, then the last job of S_A must also be before t_b; therefore, the demand over the interval does not decrease when reordering S to non-decreasing order.

Base Case: Consider when $k = 1$, clearly $S(1) = S_A(1) = (s_1)$. Thus, Equation 31 is vacuously true.

Induction Hypothesis: Assume that Equation 31 holds up to some $k < n$ for all $S'(k)$ containing the same elements of $S(k)$ such that $d(S(k)) \geq d(S'(k))$.

Inductive Step: We need to show that Equation 31 is true for $k+1$. Let $S'(k)$ be any k-subsequence. We now consider injecting s_{k+1} into $S'(k)$ and $S_A(k)$. (Note that S_A and S'_A are identical subsequences). There are three cases depending on the type of injection into $S'(k)$.

Case 1 (Leading Injection): By Property 7, $\Delta_L(S', k, k+1) \geq \Delta_F(S_A, k, k+1)$. Therefore, by the Induction Hypothesis, $d(S'(k+1)) = d(S'(k)) + \Delta_L(S', k, k+1) \geq d(S_A(k)) + \Delta_F(S_A, k, k+1) = d(S_A(k+1))$.

Case 2 (Internal Injection): By Property 8, $\Delta_I(S', k, k+1) \geq \Delta_F(S_A, k, k+1)$. Therefore, by the Induction Hypothesis, $d(S'(k+1)) = d(S'(k)) + \Delta_I(S', k, k+1) \geq d(S_A(k)) + \Delta_F(S_A, k, k+1) = d(S_A(k+1))$.

Case 3 (Final Injection): By Property 9, there exist a valid sequence $S''(k)$ obtained from $S'(k)$ by moving the highest term (s_k) to the end of $S''(k)$ where $d(S'(k)) \geq d(S''(k))$. (If $S'(k)$ already had s_k as its last term, then $S''(k) = S'(k)$). By Induction Hypothesis, $d(S''(k)) \geq d(S_A(k))$. Since the last term on $S'(k)$ and S_A is identical, $\Delta_F(S', k, k+1)$ equals $\Delta_F(S_A, k, k+1)$. Observe that $\Delta_F(S', k, k+1) \geq \Delta_F(S'', k, k+1)$ since the last element in S'' is s_k and Property 5 implies S'' will have less time added to its deadline. Therefore, we have $d(S'(k)) + \Delta_F(S', k, k+1) \geq d(S''(k)) + \Delta_F(S'', k, k+1) \geq d(S_A(k)) + \Delta_F(S_A, k, k+1)$. Hence, $d(S'(k+1)) \geq d(S_A(k+1))$.

In all the cases, we show that for all S' such that $d(S(k+1)) \geq d(S'(k)) \Rightarrow d(S'(k)) \geq d(S_A(k))$ which proves the lemma. ■

C. Starting Speed of a Dominant Sequence

Lemma 2 (Starting Speeds of a Dominant Sequence): For any non-decreasing dominant sequence S_{orig} over the interval $[t_a, t_b]$ where the first k jobs (denoted $S_{orig} = (\omega_1, \omega_2, \ldots \omega_k)$) are released in the i^{th} mode, the sequence obtained by replacing this first k jobs with jobs released at the right boundary speed of mode i, i.e., ω_{rb_i}, does not decrease the demand of the sequence in $[t_a, t_b]$.

Proof: Recall that the original sequence is $S_{orig} = (\omega_1, \omega_2, \ldots \omega_k)$. The new sequence, i.e., the one obtained by replacing the first k speeds with jobs released at the right boundary speed of mode i is simply $S_{new} = (\omega_{rb_i}, \omega_{rb_i}, \ldots, \omega_{rb_i})$. Since $c(\omega_1) = c(\omega_2) = \ldots = c(\omega_k) = c(\omega_{rb_i})$, the demand of the first k jobs of $S_{orig} = k \cdot c(\omega_{rb_i})$, which is the identical to the demand of the first k jobs of S_{new}.

Let us assume that the entire (original) sequence has n jobs, where $k \leq n$. If $k = n$, the lemma is proved. For $k < n$, we need to prove that when we replace the first k speeds with jobs released at ω_{rb_i}, the demand of the entire sequence does not decrease. Since $\widetilde{T}(\omega_{rb_i}, \omega_{rb_i}) \leq \widetilde{T}(\omega_{t_i}, \omega_{t_{i+1}})$ by Property 4, the release of the k'^{th} job, $1 \leq k' \leq k$, occurs earlier in the new sequence and the relative deadline is decreased (Property 5). As such, the deadline of the jobs released at speed ω_{rb_i} will have its deadline in the interval $[t_a, t_b]$ if the jobs released at $\omega(t_i)$, $i = 1, \ldots, k$ did. Therefore, the demand

of the entire new sequence is no less than the demand of the entire old sequence and the lemma is proved. ■

D. Speeds in Subsequent Modes of a Dominant Sequence

Lemma 1 eliminated all the decreasing speed sequences from consideration for the dominant sequence. Furthermore, Lemma 2 showed that the initial speed(s) of a dominant sequence must correspond to right boundary speeds. We now show the speed sequence pattern in the dominant sequence for subsequent modes.

Lemma 3 (Speeds Between Right Boundary Speeds in a Dominant Sequence): Consider a non-decreasing dominant speed sequence, S, for an interval $[t_a, t_b]$ where $k+1$ jobs of mode $i > 1$ are released at non-decreasing speeds $\omega_\ell, \omega_{\ell+1}, \ldots, \omega_{\ell+k}$. The previous job release is from a lower mode $h(< i)$; i.e., $\omega_{\ell-1} \leq \omega_{rb_{i-1}}$, and the subsequent job release $\omega_{\ell+k+1}$ is from some higher mode $r > i$ or does not exist (i.e., the sequence ends at $\omega_{\ell+k}$).

The sequence obtained by replacing the ℓ^{th} through $(\ell+k)^{\text{th}}$ jobs of the sequence as follows is valid, non-decreasing, and has demand no less than the original sequence: $\forall\, j \in \{\ell, \ldots, \ell+k\}$, replace the speed for ω_j with

$$\omega'_j = \min(\omega_{rb_i}, \Omega_1(\omega_{j-1}, \alpha_{\max})). \tag{32}$$

Proof: The proof is by induction on j.

Base Case: Consider $j = \ell$. If we replace ω_j with ω'_j according to Equation 32, then the speed is reachable from $\omega_{\ell-1}$ (by the second term in the $\min$ of Equation 32). Thus, the sequence remains valid up to ω'_ℓ with this replacement. Clearly, $\omega'_\ell \geq \omega_{\ell-1}$; so, the sequence remains non-decreasing up to ω'_ℓ.

In addition, the sequence $\omega_1, \ldots, \omega_{\ell-1}, \omega'_\ell$ has equivalent demand to the sequence $\omega_1, \ldots, \omega_{\ell-1}, \omega_\ell$ since the minimum time between $\omega_{\ell-1}$, ω'_ℓ is reduced (Property 4) and the execution of this subsequence is equivalent since $c(\omega'_\ell) = c(\omega_\ell)$. Furthermore, since the release of the ℓ^{th} job occurs earlier in the new sequence and the relative deadline is decreased (Property 5), the deadline of job released at speed ω'_ℓ will have its deadline in the interval $[t_a, t_b]$ if the job released at ω_ℓ in the original sequence did. Therefore, the new subsequence demand is no less than the original sequence demand.

Induction Hypothesis: Consider using Equation 32 to replace $\omega_\ell, \ldots, \omega_{\ell+k'}$ with $\omega'_\ell, \ldots, \omega'_{\ell+k'}$ for some $k' : 1 < k' < k$. Assume these replacements result in a valid, non-decreasing sequence up to $\omega'_{\ell+k'}$; furthermore, the resulting job releases up to $\omega'_{\ell+k'}$ has demand no less than the original sequence.

Inductive Step When $k'+1 < k$: Consider replacing $\omega_{\ell+k'+1}$ with $\omega'_{\ell+k'+1}$. By construction of Equation 32, $\omega'_{\ell+k'+1}$ is reachable and non-decreasing from $\omega'_{\ell+k'}$. Thus, the new subsequence remains valid up to $\omega'_{\ell+k'+1}$. Using an identical argument to the base case, it is clear that the demand of the sequence is no less when compared to the original subsequence up to the $(\ell+k'+1)^{\text{th}}$ job release.

Inductive Step When $k'+1 = k$: In this special (terminating) case of the inductive step, we also show that the entire sequence is valid, non-decreasing, and has unchanged demand. The same steps of the case for $k'+1 < k$ can be used to show that the subsequence up until (and including) $\omega'_{\ell+k}$ is valid, non-decreasing, and equivalent in demand.

We first show that the entire sequence is non-decreasing. All that is required is to show that $\omega'_{\ell+k} \leq \omega_{\ell+k+1}$. (The induction hypothesis shows the previous portion is non-decreasing and speeds after $\omega_{\ell+k+1}$ are already non-decreasing by supposition of the lemma). If $\omega_{\ell+k+1}$ does not exist, we are finished. Otherwise, observe that since $\omega_{\ell+k+1}$ is in a mode higher than mode i, it must have speed exceeding ω_{rb_i}. By the first term in the $\min$ in Equation 32, $\omega'_{\ell+k} \leq \omega_{\ell+k+1}$.

To prove validity of the entire sequence, observe that each of the replaced speeds in the sequence exceed or equal the original speed. Thus, $\omega'_{\ell+k} \geq \omega_{\ell+k}$. This implies that $\Omega_1(\omega'_{\ell+k}, \alpha_{\max}) \geq \Omega_1(\omega_{\ell+k}, \alpha_{\max}) \geq \omega_{\ell+k+1}$. The last inequality is due to $\omega_{\ell+k+1}$ being reachable from $\omega_{\ell+k}$ in a single rotation. Therefore, it follows that $\omega_{\ell+k+1}$ is still reachable for $\omega'_{\ell+k}$.

By the same reasoning for $k'+1 = k$ case, the demand for jobs $1, 2, \ldots, \ell+k$ is unchanged since the deadline of each job occurs earlier, as jobs are released earlier and the relative deadline of each job is shorter due to the replaced job occurring at a higher speed. Similarly, the jobs after $\ell+k$ have the same execution time (their speed is unchanged), and are released earlier due to the shortened inter-arrival times of jobs $\ell, \ldots, \ell+k$. Thus, if any of the jobs after $\ell+k$ had their absolute deadline in $[t_a, t_b]$, they continue to have their deadline in the interval; the demand of the entire sequence does not decrease after replacing the jobs. ■

VI. The Dominant Sequence Set

The previous section derived the necessary properties of a dominant speed sequence but the lemmas do not explicitly tell us what the actual dominant speed sequence is. In this section, we define the *dominant sequence set* which was used in Section IV to define the precedence constraint knapsack problem as a set of speed sequences that must contain the dominant speed sequence. Theorem 1 below will formally characterize this set.

First, let us give some notation. Let Ψ be an ordered set of speeds (non-decreasing order of speed) called the *dominant speed set* formally defined as follows:

$$\Psi = \{\Omega_n(\omega_{rb_i}, \alpha_{\max}) | (n \in \mathbb{N}_0) \wedge (\omega_{rb_i} \in \mathbf{\Omega_{rb}})\}. \tag{33}$$

Let ω_k be a speed in some non-decreasing speed sequence S obtained from using speeds in Ψ, $\mathsf{nextPossibleSpeed}(\omega_k)$ be a function that returns a set of valid subsequent speeds ω_{k+1} from the set Ψ that we need to consider given that we released a job in a speed sequence at speed $\omega \in \Psi$. We define ω_0 to be a sentinel speed to indicate that we are choosing the first speed of the sequence next. Intuitively, Lemma 2 implies that we must start with a right boundary speed; thus, $\mathsf{nextPossibleSpeed}(\omega_0)$ should be the set of right boundary speeds. For any other $k > 0$, if ω_k is a right boundary speed, Lemma 2 implies (as we will show in Theorem 1) that we may either remain at that right boundary speed, transition to a (reachable) right boundary speed of a higher mode, or

accelerate maximally to the next reachable speed. For an ω_k that is not a right boundary speed, we may only transition to a (reachable) right boundary speed of a higher mode, or accelerate maximally to the next reachable speed. Formally,

$$\mathsf{nextPossibleSpeed}(\omega_k) = \begin{cases} \mathbf{\Omega_{rb}} & \text{if } k = 0 \\ \{\Omega_1(\omega, \alpha_{\max})\} \cup \{\omega_{rb_i} \in \mathbf{\Omega_{rb}}(\omega_k)\} & \text{if } k > 0 \end{cases} \tag{34}$$

where, $\mathbf{\Omega}_{rb}(\omega_k)$ as defined earlier denotes the set of reachable right boundary speeds from ω_k. Please note that $\mathbf{\Omega}_{rb}(\omega_k)$ can return ω_k as a member of the set if ω_k is a right boundary speed; i.e., a right boundary speed is reachable from itself.

Let $\mathbb{S}(\delta)$ be a set of speed sequences defined as follows:

$$\left\{ \begin{array}{l} (\omega_1, \ldots, \omega_{|S|}) \in 2^{\Psi} \mid \\ \quad \left(\forall k \in \mathbb{N}_0^{|S|-1}, \omega_{k+1} \in \mathsf{nextPossibleSpeed}(\omega_k)\right) \\ \quad \wedge \left(\sum_{\ell=1}^{|S|-1} \widetilde{T}(\omega_\ell, \omega_{\ell+1}) + \Omega_1(\omega_{|S|}, \alpha_{\max}) \leq \delta\right) \end{array} \right\} \tag{35}$$

Theorem 1: The set $\mathbb{S}(\delta)$ must contain a dominant speed sequence for any interval of length $\delta > 0$.

Proof: The correctness of the theorem lies in proving that $\mathsf{nextPossibleSpeed}(\omega_k)$ always returns a dominant sequence. By Lemma 1 only non-decreasing sequences are considered.

In Equation 34, $k \geq 0$. According to Lemma 2, the first speed of a dominant sequence must be a right boundary speed, this proves Equation 34 when $k = 0$. We need to prove it when $k > 0$.

When $k > 0$, there are several possibilities for the k^{th} speed:

Case 1 (k^{th} job is the first job in mode $i > 1$): In this case, $(k-1)^{\text{th}}$ speed may or may not be a right boundary speed. Applying Equation 32, k^{th} speed will be replaced by $\min(\Omega_1(\omega_{k-1}, \alpha_{\max}), \omega_{rb_i})$, proving Equation 34 for Case 1.

Case 2 (k^{th} speed is the right boundary speed of mode i): Equation 32, when applied for ω_k will guide us to replace ω_k by ω_k itself because we assumed $\omega_k = \omega_{rb_i}$, proving Case 2.

Case 3 (k^{th} speed is an intermediate speed in the middle of mode i): This case follows on the application of Equation 32.

In addition, only jobs that are released and whose deadlines fall within an interval of length δ can be part of the dominant sequence. This can be verified by examining the definition of the demand bound function $\mathsf{dbf}(\delta)$ in Equation 7.

Together, Lemmas 1, 2, and 3 allow for elimination of unnecessary sequences from the dynamic programming search while Theorem 1 ensures the sequences produced by Algorithm 1 are dominant speed sequences whose demand coincide with Equation 7. ■

VII. Evaluation

In this section, we compare our algorithm against the DRT algorithm [5], which is the state-of-the-art technique for solving the problem under consideration. Our approach explores a subset of speeds considered by Mohaqeqi et al. [5], and so we inherit the same upper bound on number of speeds: $O\left(m \cdot \frac{\omega_{max}^2 - \omega_{min}^2}{2 \cdot \alpha_{max}}\right)$. Even though our algorithm shares similar complexity with the DRT algorithm with respect to the speeds, we eliminate several unnecessary traversals through these speeds, which significantly reduces the computational complexity of our algorithm. We compare the accuracy and runtime of the algorithms using two experiments. For all experiments, a maximum acceleration of 600,000 rev/min^2 and maximum deceleration of -600,000 rev/min^2 were assumed as in previous work [3]–[5], [17]. In each experiment, the worst-case demand is calculated for 100 time intervals in $[0, 1s]$ in steps of 10 ms. The experiments were performed using Python 3.6.5 on a 3.40 GHz, quad-core processor with 8 GB of RAM. While the results are platform dependent, the general trend showing the relative performance of the proposed approach against the DRT algorithm should be representative. Each experiment is run 10 times and the average values are reported. The code and the original data used for this publication can be found on the artifact evaluation page [18].

TABLE I
Task set used by existing work [5], [8]

i^{th} mode	1	2	3	4	5	6	ω_{rb_m}
ω_{rb_i}	500	1500	2500	3500	4500	5500	6500
$c(\omega_{rb_i})$	965	576	424	343	277	246	

TABLE II
A more general task set.

i^{th} mode	1	2	3	4	5	6	ω_{rb_m}
ω_{rb_i}	1200	2200	3200	4200	5200	6200	7200
$c(\omega_{rb_i})$	965	576	424	343	277	246	

In the first experiment, two task sets were used. The first task set appeared in existing publications [5], [8] (Table I). Note that this task set is not ideal, as some boundary speeds can be reached from the others in an integer number of rotations using maximum acceleration, which simplifies the search for the worst-case demand. The results are shown in Table III(a). Though both approaches were able to find the worst-case demand, our algorithm is 13.5 times faster.

Since the first task set is not ideal, as described earlier, we created another task set (Table II). This task set is more general in the sense that the right boundary speeds are not

TABLE III
Run time comparison of different algorithms

	Demand (in μs) over [0,1s]	Runtime
DRT Alg.	26,568	3 min. 31 sec.
Our Alg.	26,568	15.63 sec.

(a) An existing task set

	Demand (in μs) over [0,1s]	Runtime
DRT Alg.	35,892	17 min. 2 sec.
Our Alg.	35,892	19 sec.

(b) A more general task set

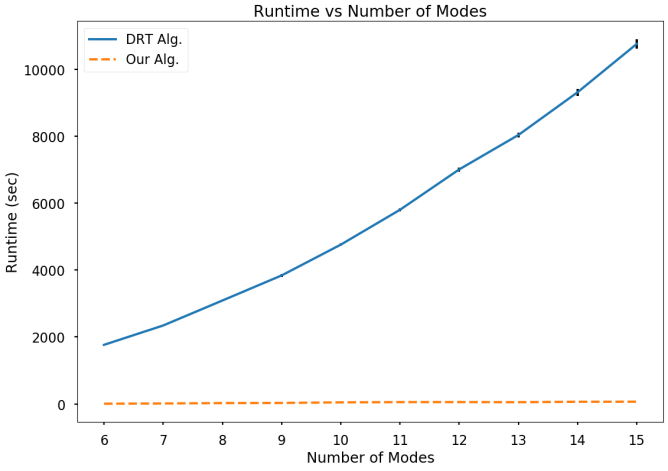


Fig. 7. Graph depicting the runtime of different algorithms as a function of the number of modes of randomly generated AVR task sets. For each mode, the 95% confidence interval is shown.

reachable from one another in integer number of rotations when using $\alpha_{\max}$. Hence, we expect that the algorithms will require longer running times to find the worst-case demand of this task set. The results are shown in Table III(b). As before both approaches were able to find the worst-case demand but our algorithm is 53.8 times faster.

For the second experiment, we generated multiple AVR tasks using an algorithm presented by Biondi et al. [3]. In this algorithm, multiple AVR tasks are modeled as a single task, called the representative AVR task, by combining the execution times and the boundaries speeds. The modes of the representative AVR task are generated by assuming a fixed value of 0.25 for the maximum utilization factor of the modes. Again, the same worst-case demands were found by both algorithm, but overall, our approach is 146 times faster on average and up to 250 times faster, as shown in Fig. 7.

VIII. Conclusions and Future Work

This paper presented an efficient method for calculating the exact worst-case demand bound function of AVR tasks. First, a knapsack-based dynamic programming approach was proposed to efficiently find the worst-case demand. Second, a collection of necessary conditions were presented, which reduce the search space of the knapsack-based approach to find the dominant sequence set. Experimental results confirm that the proposed approach is exact and is faster than the state-of-the-art technique. In the future, we plan on analyzing the worst-case demand of AVR tasks that have different phases and which are released by independent sources.

Acknowledgements.

This research was supported in part by the US National Science Foundation (CNS Grant Nos. 1618185 & 1618979) and a Thomas C. Rumble Graduate Fellowship from Wayne State University.

References

[1] D.Buttle, "Real-time in prime-time," keynote speech at the 24th Euromicro Conference on Real-Time Systems, Pisa, Italy, July 12, 2012.

[2] C. L. Liu and J. W. Layland, "Scheduling algorithms for multiprogramming in a hard-real-time environment," *J. ACM*, vol. 20, no. 1, pp. 46–61, Jan. 1973. [Online]. Available: http://doi.acm.org/10.1145/321738.321743

[3] A. Biondi, M. Di Natale, and G. Buttazzo, "Response-time analysis for real-time tasks in engine control applications," in *Proceedings of the ACM/IEEE Sixth International Conference on Cyber-Physical Systems*, ser. ICCPS '15. New York, NY, USA: ACM, 2015, pp. 120–129. [Online]. Available: http://doi.acm.org/10.1145/2735960.2735963

[4] A. Biondi, G. Buttazzo, and S. Simoncelli, "Feasibility analysis of engine control tasks under edf scheduling," in *2015 27th Euromicro Conference on Real-Time Systems*, July 2015, pp. 139–148.

[5] M. Mohaqeqi, J. Abdullah, P. Ekberg, and W. Yi, "Refinement of Workload Models for Engine Controllers by State Space Partitioning," in *29th Euromicro Conference on Real-Time Systems (ECRTS 2017)*, ser. Leibniz International Proceedings in Informatics (LIPIcs), M. Bertogna, Ed., vol. 76. Dagstuhl, Germany: Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2017, pp. 11:1–11:22. [Online]. Available: http://drops.dagstuhl.de/opus/volltexte/2017/7159

[6] M. Stigge, P. Ekberg, N. Guan, and W. Yi, "The digraph real-time task model," in *2011 17th IEEE Real-Time and Embedded Technology and Applications Symposium*, April 2011, pp. 71–80.

[7] J. Kim, K. Lakshmanan, and R. R. Rajkumar, "Rhythmic tasks: A new task model with continually varying periods for cyber-physical systems," in *Proceedings of the 2012 IEEE/ACM Third International Conference on Cyber-Physical Systems*, ser. ICCPS '12. Washington, DC, USA: IEEE Computer Society, 2012, pp. 55–64. [Online]. Available: http://dx.doi.org/10.1109/ICCPS.2012.14

[8] A. Biondi, A. Melani, M. Marinoni, M. D. Natale, and G. Buttazzo, "Exact interference of adaptive variable-rate tasks under fixed-priority scheduling," in *2014 26th Euromicro Conference on Real-Time Systems*, July 2014, pp. 165–174.

[9] Z. Guo and S. Baruah, *Uniprocessor EDF scheduling of AVR task systems*. Association for Computing Machinery, Inc, 4 2015, pp. 159–168.

[10] A. Biondi and G. Buttazzo, "Real-time analysis of engine control applications with speed estimation," in *2016 Design, Automation Test in Europe Conference Exhibition (DATE)*, March 2016, pp. 193–198.

[11] A. Biondi and G. Buttazzo, "Modeling and analysis of engine control tasks under dynamic priority scheduling," *IEEE Transactions on Industrial Informatics*, pp. 1–1, 2018.

[12] A. Biondi, M. Di Natale, and G. Buttazzo, "Performance-driven design of engine control tasks," in *Proceedings of the 7th International Conference on Cyber-Physical Systems*, ser. ICCPS '16. Piscataway, NJ, USA: IEEE Press, 2016, pp. 45:1–45:10. [Online]. Available: http://dl.acm.org/citation.cfm?id=2984464.2984509

[13] H. C. Verma, *Concepts of Physics - Vol. 1*. Bharati Bhawan Publishers and Distributors, 2010.

[14] G. Cho and D. X. Shaw, "A depth-first dynamic programming algorithm for the tree knapsack problem," *INFORMS Journal on Computing*, vol. 9, no. 4, pp. 431–438, 1997. [Online]. Available: https://doi.org/10.1287/ijoc.9.4.431

[15] D. S. Johnson and K. A. Niemi, "On knapsacks, partitions, and a new dynamic programming technique for trees," *Mathematics of Operations Research*, vol. 8, no. 1, pp. 1–14, 1983. [Online]. Available: https://doi.org/10.1287/moor.8.1.1

[16] H. Kellerer, U. Pferschy, and D. Pisinger, *Knapsack Problems*. Springer Berlin Heidelberg, 01 2004.

[17] R. I. Davis, T. Feld, V. Pollex, and F. Slomka, "Schedulability tests for tasks with variable rate-dependent behaviour under fixed priority scheduling," in *2014 IEEE 19th Real-Time and Embedded Technology and Applications Symposium (RTAS)*, April 2014, pp. 51–62.

[18] S. K. Bijinemula, A. Willcock, T. Chantem, and N. Fisher, "Code for the paper-efficient knapsack-based approach for calculating the worst-case demand of avr tasks," 2018. [Online]. Available: https://github.com/bsk1410/Efficient-Knapsack-for-AVR-tasks-RTSS2018

APPENDIX

A. Proof of Property 9

The property is restated here for readability.

Property 9 (Highest-Speed Relative-Deadline Dominance): For any valid, finite sequence $S = (\omega_1, \ldots, \omega_n)$, if ω_ℓ ($\ell \in \mathbb{N}_1^n - 1$) is the highest speed s_n and not the last speed of sequence S and $s_n \leq \Omega_1(\omega_n, \alpha_{\max})$ then new sequence S' with the highest element moved to the last element (i.e., $S' = (\omega_1, \ldots, \omega_{\ell-1}, \omega_{\ell+1}, \ldots, \omega_n, s_n)$) is valid and has the following property:

$$d(S) \geq d(S'). \tag{36}$$

Proof: Consider a valid, finite sequence $S = (\omega_1, \ldots, \omega_n)$, where ω_ℓ ($\ell \in \mathbb{N}_1^n - 1$) is the highest speed s_n and not the last speed of sequence S and $s_n \leq \Omega_1(\omega_n, \alpha_{\max})$.

Let S be constructed as described in the statement of the lemma: $S' = (\omega_1, \ldots, \omega_{\ell-1}, \omega_{\ell+1}, \ldots, \omega_n, s_n)$. Observe that S' is valid since s_n is reachable from ω_n; also, since $\Omega_1(\omega_{\ell-1}, \alpha_{\max}) \geq s_n \geq \omega_{\ell-1}$ and $\Omega_1(\omega_{\ell+1}, \alpha_{\max}) \geq s_n \geq \omega_{\ell+1}$, which implies that $\omega_{\ell-1}$ and $\omega_{\ell+1}$ are reachable from each other.

We define $\Delta_H(S, S')$ to be the difference $d(S) - d(S')$:

$$\begin{aligned}\Delta_H(S, S') = \quad & \widetilde{T}(\omega_{\ell-1}, s_n) + \widetilde{T}(s_n, \omega_{\ell+1}) + \tilde{d}(\omega_n) \\ & -\widetilde{T}(\omega_{\ell-1}, \omega_{\ell+1}) - \widetilde{T}(\omega_n, s_n) - \tilde{d}(s_n).\end{aligned} \tag{37}$$

We now prove that $\Delta_H(S, S') \geq 0$ which proves Equation 36 of the property. Let s_{n-1} and s_{n-2} be the second and third largest speed of S, respectively.

The rest of the proof is nearly identical to Property 8. Observe that by Properties 2 and 5, $\widetilde{T}(f, \omega + \epsilon) \geq \tilde{d}(\omega + \epsilon)$ for all f, ω and $\epsilon > 0$. Also, $\widetilde{T}(s_k + \epsilon, \omega) = \widetilde{T}(\omega, s_k + \epsilon)$ for all ω and $\epsilon > 0$ by Property 2. These properties imply that:

$$\begin{aligned}& \widetilde{T}(s_{n-2}, s_{n-1}) + \widetilde{T}(s_{n-1}, s_{n-1}) + \tilde{d}(s_{n-1}) \\ & \quad = \widetilde{T}(s_{n-2}, s_{n-1}) + \widetilde{T}(s_{n-1}, s_{n-1}) + \tilde{d}(s_{n-1}) \\ \Rightarrow \quad & \widetilde{T}(s_{n-2}, s_{n-1} + \epsilon) + \widetilde{T}(s_{n-1} + \epsilon, s_{n-1}) + \tilde{d}(s_{n-1}) \\ & \quad \geq \widetilde{T}(s_{n-2}, s_{n-1}) + \widetilde{T}(s_{n-1}, s_{n-1} + \epsilon) + \tilde{d}(s_{n-1} + \epsilon)\end{aligned} \tag{38}$$

Setting $\epsilon = s_n - s_{n-1}$ and substituting into the above inequality of Equation 38, we get the following:

$$\begin{aligned}& \widetilde{T}(s_{n-2}, s_n) + \widetilde{T}(s_n, s_{n-1}) + \tilde{d}(s_{n-1}) \\ & \quad \geq \widetilde{T}(s_{n-2}, s_{n-1}) + \widetilde{T}(s_{n-1}, s_n) + \tilde{d}(s_n) \\ \Rightarrow \quad & \widetilde{T}(s_{n-2}, s_n) + \widetilde{T}(s_n, s_{n-1}) - \widetilde{T}(s_{n-2}, s_{n-1}) \\ & \quad \geq \widetilde{T}(s_{n-1}, s_n) + \tilde{d}(s_n) - \tilde{d}(s_{n-1})\end{aligned} \tag{39}$$

Seeing that both $\omega_{\ell-1}$ and $\omega_{\ell+1}$ are at most s_{n-1} and either one of $\omega_{\ell-1}$ and $\omega_{\ell+1}$ must be less than s_{n-2}, we get:

$$\begin{aligned}& \widetilde{T}(\omega_{\ell-1}, s_n) + \widetilde{T}(s_n, \omega_{\ell+1}) + \tilde{d}(\omega_n) \\ & \quad -\widetilde{T}(\omega_{\ell-1}, \omega_{\ell+1}) - \widetilde{T}(\omega_n, s_n) - \tilde{d}(s_n) \geq 0\end{aligned} \tag{40}$$

The last inequality (which implies Equation 37 of the property) above follows from observing that according to Property 4, the following is true for all ω and ω':

$$\begin{aligned}& \frac{\partial \widetilde{T}(\omega, s_{k+1})}{\partial \omega} \leq \frac{\partial \widetilde{T}(\omega, \omega')}{\partial \omega} \\ \Leftrightarrow \quad & \frac{\partial \widetilde{T}(\omega, s_{k+1})}{\partial \omega} + \frac{\partial \widetilde{T}(s_{k+1}, \omega')}{\partial \omega} - \frac{\partial \widetilde{T}(\omega, \omega')}{\partial \omega} \leq 0\end{aligned}$$

■

B. Table of Notation and Units

Symbol	Term	Unit
ω	Angular speed	rev/min
ω_{rb}	Right boundary speed	rev/min
$\mathbf{\Omega_{rb}}$	Set of right boundary speeds	N/A
$\omega_{\max}$	Maximum angular velocity	rev/min
α	Angular acceleration	rev/min^2
$\alpha_{\max}$	Maximum angular acceleration	rev/min^2
$\alpha_{\min}$	Minimum angular acceleration	rev/min^2
t	Time	sec.
$\omega(t)$	Instantaneous angular velocity	rev/min
$c(\omega(t))$	Worst-case execution time	sec.
θ	Angular position	rev
$\Delta\theta$	Change in angular distance	rev
Ω_n	Angular velocity at the end of 'n' rotations	rev/min
$\tilde{T}$	Minimum interarrival time	sec.
ω_p	Peak angular velocity	rev/min
f	Angular velocity at the end of a rotation	rev/min
$d(\omega)$	Relative deadline	sec.
D_ω	Demand	sec.
dbf	Demand Bound Function	sec.
δ	Time interval length	sec.
$\mathbb{N}_j^k$	Number set $j, j+1, ..., k$	N/A
G_I	Dependency graph	N/A
V_I	Vertices	N/A
A_I	Edges	N/A
η	Number of job releases	N/A
$\mathbb{Z}^+$	Positive integers	N/A
S	Speed sequence	N/A
s_i	Job release speed	rev/min
Δ_L	Leading injection	N/A
Δ_F	Final injection	N/A
Δ_I	Internal injection	N/A