Demand Characterization and Codesign of Dynamics-Adaptive Real-Time Systems

by

AARON WILLCOCK

PROSPECTUS

Submitted to the Graduate School,

of Wayne State University,

Detroit, Michigan

in partial fulfillment of the requirements

for the degree of

DOCTOR OF PHILOSOPHY

2021

MAJOR: COMPUTER SCIENCE

Approved By:

_____

Advisor Date

DEDICATION

For Ellen,

Dad,

Mom,

Carina,

and Connor.

ACKNOWLEDGEMENTS

No words will sufficiently describe the gratitude I have for Professor Nathan Fisher's guidance in these years of study and self-development. Without his mentorship, patience, and support, this work would not be possible and I would not be the person I am today.

I'm deeply indebted to Professor Tam Chantem for her persistent support and feedback - especially from so far away. I'm very grateful for the many hours of discussion and guidance on writing, collaboration, and our work together.

Special thanks is also owed to the colleagues, mentors, and teachers outside of academia who taught me to teach, to speak publicly, and to never stop asking questions:

Mrs. Stafford, Kelly Kozlowski, and Mr. Arscheene.

I would also like to acknowledge all of my students, especially Blitz Creek and the ThunderChickens, for the opportunity to mentor and teach. It is your desire to learn, break things, and build them again that makes learning and teaching exciting - every day.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1 INTRODUCTION

1.1

THIS CITATION IS TO KEEP from breaking things until i add a real citation [34]

CHAPTER 2 RELATED WORK

I <3 my Wayne State Libraries! Do you? [34]

CHAPTER 3 SYSTEM MODELS AND TERMS

I <3 my Wayne State Libraries! Do you? [34]

CHAPTER 4 CODESIGN OF SOFTWARE

4.1 Introduction

Motivation and Applications From cell phones to solar panels, devices of all sizes require power semiconductors to control, direct, and manage the flow of electricity. Batteries, CPUs, and photovoltaic inverters all leverage this flow of electricity, known as current, to perform tasks from actuation to computation. In excess, however, current leads to thermal cycling and can be degrading or destructive to power circuitry. One cause of excessive current is short-circuiting. A short-circuit occurs when current travels through an alternate, unintended path in a circuit often with little or no resistance. This alternate path of travel with no resistance leads to high current, increased heat, and often circuit damage. This potential for damage creates a need for short-circuit protection.

The need for short protection can be seen in a variety of applications relying on power semiconductors including power converters and inverters [21]. In an industrial setting, this can include photovoltaic systems and hybrid fuel cells [40]. In a consumer setting, this can include cell phones and other portable electronics like the recent Samsung Galaxy Note 7 which was recalled due to fires caused by short-circuits in the device battery [19].

To mitigate the risk of short-circuits, devices using power-semiconductors can be constructed with short-circuit protection in the form of a fuse, thermal breaker, or other hardware designated to prevent the high current responsible for circuit damage. This short-circuit protection, however, is often fixed circuitry dedicated solely to short detection. In such systems, little flexibility is afforded to circuits which operate at varying voltages and currents over their lifetime as designers must protect against short-circuits at the highest

currents and voltages, even if they are not the most frequently used. Moreover, the rise in semiconductor power density continues to reduce the required latency for detecting and halting shorts [20].

Problem Statement In light of the motivation above, the problem of designing flexible short-circuit protection systems can be viewed through the lens of real-time software. Given the inherent possibility of catastrophic failure in short-circuit protection systems and the time-sensitive nature of current rise during short-circuits, we seek to frame the problem as one of hardware-software co-design via hard real-time systems. Specifically, we aim to address the problem of short-circuit protection in direct current (DC) resistor-inductor (RL) circuits - circuits containing resistors and inductors where current flows in only one direction.

To address the need for flexible, short-circuit protection, our problem statement is: Given a Direct-Current Resistor-Inductor circuit, devise a hardware-software co-design approach which relates hard-real time requirements to hardware size. More specifically, our objective is to construct a hardware-software relationship which allows designers to:

1. minimize hardware size while meeting maximum real-time utilization requirements, and

2. minimize real-time utilization while meeting maximum hardware size requirements.

Proposed Solution To address the need for flexible, real-time short-circuit protection, we propose a short-protection method via a real-time task as follows:

A DC RL circuit containing an air-core inductor placed in series between the system's

resistive load and ground has is connected (and controlled) by a microcontroller.The microcontroller executes a real-time task responsible for sampling voltage across the inductor (or a small resistor) to measure current via an Analog-to-Digital Converter (ADC) pin. Short protection is accomplished by identifying the maximum expected current and the maximum rate of current change as limited by the circuit's inductor. Using the inductor spatial parameters in conjunction with ADC sampling times, a minimum sampling period is derived for the real-time task. From this minimum period, a relationship between minimum real-time utilization under Earliest Deadline First (EDF) scheduling and inductor volume is provided. Real-time or physical system constraints may be applied to this relationship to facilitate the hardware-software co-design of a real-time short-circuit protection system.

This approach is intended to allow existing systems with microprocessors to migrate short-circuit protection from dedicated-circuitry-only to a software-based implementation and future systems to be designed with the proposed hardware-software tradeoff in mind.

Contributions The software-based protection methods depicted herein provide an alternative short-circuit protection technique to circuit designers. By relating utilization to the volume of (and board space consumed by) an inductor, system designers may trade short-protection circuitry for real-time task utilization on the microcontroller, leveraging either end of the relationship to meet fault-tolerance and space requirements. For example, applications with little available board space may opt for smaller inductors (minimizing board space) and greater utilization. Example applications include smaller IGBT modules as found in electric vehicles or applications where minimizing weight is import[23]. In con-

trast, larger applications with more available board space or a greater real-time task set may opt for a larger inductor and thus a smaller utilization for the short-protection task. Example applications include high power IGBT modules in photovoltaic and wind turbine inverters [40][9]. Perhaps most importantly, the established relationship between board space and processor utilization acts as a conduit through which advancements in electrical engineering may improve real-time system efficiency and vice versa. To summarize, our contributions include[1]:

1. a software-based short protection method for Direct Current Resistor-Inductor circuits,

2. a relationship between air-core inductor spatial parameters and real-time processor utilization under preemptive uniprocessor EDF scheduling for short-circuit protection,

3. a process for identifying optimal inductor orientation given a fixed volume, and

4. a process for minimizing utilization given a fixed volume for an air-core inductor and vice versa.

Outline Chapter 4.2 details the related work in both problem domains. Chapter 4.3 provides an electronics background and nomenclature overview. Chapter 4.4 provides the first paper contribution, a model for identifying circuit properties. Chapter 4.5 depicts short protection methods given the constraints provided in the circuit model. Chapter 4.6 formalizes the relation between real-time scheduling and short-circuit protection. Chapter 4.7 provides the model optimization for both fixed board constraints and fixed real-time

[1]This work was published in the 2017 IEEE 23rd International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA) under the same title [38] and is an extension upon the related senior thesis by Willcock [37].

utilization as contributions from this paper. Chapter 4.8 details experiments conducted to validate the proposed relationship and theoretical utilization. Chapter 4.9 discusses the results of experimentation. Chapter 4.10 identifies conclusions and future work.

4.2 Related Work

To the best of our knowledge, current approaches to short-circuit protection are rooted in dedicated circuitry and do not use adaptive real-time processing. Modern methodologies include Zone Selective Interlocking on systems with alternating current (AC) which, while successful, cannot be directly applied to DC [14]. Other methods use the Rogowski coil in conjunction with differential and integral signals from the coil to detect a short [36]. While feasible, the Rogowski coil implementation is large and does not slow current rise in the system. Some methods only require sampling of gate emitter voltage and a reference voltage [20]. Krone et al. use the gate emitter voltage, but also reference the DC link capacitors to assist in protection [27]. Each method, while different, relies solely on dedicated circuitry for short protection. Although variations exist in the components used and the design of the circuits, we sought an alternative where required circuitry was minimized.

Furthermore, protection methods relying on the change in current, as in Hain and Bakran [17], require an inductor. This approach also includes auxiliary MOSFETs, latch circuits, and comparators. Additionally, some current protection methods relying on gate charge require differential amplifiers attached to auxiliary MOSFETs as found in Horiguchi et al. which also require an inductor in test circuits [21]. Both models have more components than the single inductor required for protection in this approach, providing more motivation for software-based protection.

In the area of cyber-physical and real-time systems, the senior thesis [37] upon which this work and its conference-published variant [38] are based focused primarily on estab-

lishing a relationship between the inductance of an inductor and the utilization required for the task. This work extends the results of Willcock [37] by incorporating board space consumed into the utilization calculation and providing experimental validation of the extended relationship with low-cost hardware. Excluding the preceding variant of this work, we are unaware of other cyber-physical or real-time works specific to short protection. However, works identifying adaptive real-time tasks with multiple operating modes are present [22]. Examples include thermal-aware computing in Hettiarachchi et al [18] and rate-adaptive tasks as in Buttazzo et al. [10]. Biondi and Buttazzo furthered this model with thorough analysis of its implications on the executing processor [2]. These works address properties of environment and power-aware real-time tasks but are not specific to short-circuit protection.

4.3 Electronics Background

The following chapter covers background required for constructing the proposed circuit model. It includes an overview of nomenclature, DC RL circuits, inductors, and short-circuits. These electronics fundamentals can be found in a typical collegiate physics textbook [39].

Nomenclature For the purposes of describing our approach, we rely on the following nomenclature:

Term	Symbol	Unit	Description
Current	I	Ampere (A)	The rate of electric charge flow
Inductance	L	Henry (H)	The ability to induce electromotive force (voltage)
Voltage	V	Volt (V)	The difference in electric potential between two points

First-Order DC RL Circuits Direct current (DC) circuits are circuits in which the direction of current flow does not change [39]. A DC circuit in which current passes through a resistor and an inductor is deemed a DC RL circuit where "R" represents the resistor and "L" the inductor. The model presented in Chapter 4.4 relies on a first-order DC RL circuit with the resistor and inductor in series. An example first-order DC RL circuit is provided in Figure 1 with two resistors and an inductor in series.

Inductors An inductor is a passive electronic component typically illustrated as a coil or four- which resists change in current flow through itself. This property is useful as current

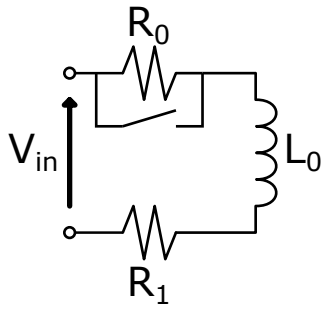


Figure 1: Model DC RL Circuit
A DC RL circuit with load R_0, inductor L_0, resistor R_1, and switch for inducing shorts.

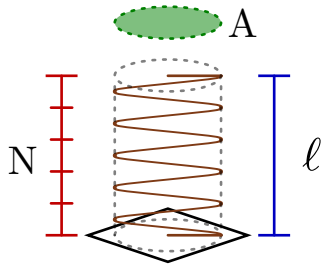


Figure 2: Spatial parameters of an air core inductor

through an inductor cannot change instantaneously. Equations describing these properties are provided in Chapter 4.5.

4.3.1 Spatial Parameters

The model proposed in Chapter 4.4 will rely on a single, air-core inductor. The spatial properties of an inductor may be modeled as seen in Figure 2. The figure describes a solenoid-style inductor where N represents the number of complete turns, A is the area of a coil, ℓ represents the length of the inductor. The relationship between these parameters and inductance can be constructed from fundamental electricity and magnetism equations found in [39]. This relationship is modeled as:

$$L = \frac{\mu N^2 A}{\ell} \tag{4.1}$$

where μ is the permeability of free space. For our purposes, we assume the spatial parameters are fixed – i.e., ℓ, A, and N are static[2]. Equation (4.1) will be referenced in Chapter 4.5 to relate current-flow to inductor size and in Chapter 4.7 to optimize hardware-software co-design solutions.

4.3.2 Assumption of Constant Turn Density

Equation (4.1) contains the term $\frac{N}{\ell}$, referred to as *turn density*. *Turn density* must remain constant with an increase in length for inductance to increase. Thus if an inductor is to be extended from length ℓ to $2{\cdot}\ell$, the number of turns N should be doubled accordingly (to $2 \cdot N$) to maintain constant *turn density*. Doubling both N and ℓ results in doubled inductance L. Whenever a change in inductor length ℓ is suggested, we assume the number of turns is doubled as well to maintain constant *turn density*. We rely on this assumption throughout Chapter 4.7.

4.3.3 Inductor Core Composition

The core of a solenoid-style inductor can be defined as the area within the coiled wire. In practice, many inductors are manufactured with ferromagnetic cores which increase the inductance for the same volume. In our model, we assume this core is empty and contains only air. Equation (4.1) reflects this assumption as it applies only to air-core inductors. We view this assumption as an upper bound on board space consumed since ferromagnetic core inductors provide a greater inductance by volume [15].

Definition of Short and Fault Types For the purposes of this model, a short is defined as the flow of current through an alternate, unintended path in a circuit with little or

[2]In Chapter 4.7, we consider variable sized inductors as part of our hardware-software co-design optimization.

no impedance. In the event of a short, this loss of impedance causes a large change in current which can be slowed and detected through careful reliance on an inductor's ability to resist instantaneous changes in current. However, short-circuits lead to joule heating, the process whereby current through a conductor releases heat modeled as:

$$\text{Heat} \propto RI^2t \tag{4.2}$$

Here, R is resistance, I is current, and t is time [39]. The thermal buildup from joule heating is responsible for the permanent damage that can result from a short and therefore serves as motivation for short-circuit protection. Protection from damage, however, requires catching two short-circuit fault types: the Fault under Load (FUL) and the Hard-Switching Fault (HSF).

4.3.4 Fault Under Load vs Hard-Switching Fault

In the context of short-circuit protection, a Fault Under Load is a short-circuit fault where a circuit has an active load at the time of the fault. In this type of fault, the current and impedance are both non-zero. Intuitively, this means the circuit is "on" at the instant the short-circuit occurs. The short forces current to rise, often above desired operating ranges. This fault type, commonly analyzed in IGBTs, has been studied in [32].

In contrast, a Hard-Switching Fault is a short-circuit fault where the circuit does not have an active load at the time of the fault [21]. In contrast to an FUL, an HSF does not have an initial operating current since there must be a short-circuit path before the circuit is "on".

Current as a Function of Time In the proposed system, a real-time task must be related to the change in current through an inductor. The following equation will provide such a relationship used Chapter 4.5:

$$I(t) = \frac{V}{R}(1 - e^{-t\frac{R}{L}}) \tag{4.3}$$

where $I(t)$ is current at time t, V is a constant voltage to the circuit, R is resistance, and L is inductance.

4.4 Electronic System Model

In the previous chapter, an electronics background was provided as context for our approach. In this chapter, we will define and describe the system model from an electronics perspective. Relying on the provided background, we propose an electronics model with four primary components:

1. a DC RL circuit to which software-based short-circuit protection is applicable and board space is consumed by air-core solenoid-style inductor,

2. an Operating Current Model (OCM) to characterize first-order DC RL circuits,

3. two short-circuit detection methods for first-order DC RL circuits, and

4. a real-time sporadic task for short protection.

The short-circuit detection methods will be discussed in Chapter 4.5 and the real-time model (and its utilization analysis under uniprocessor EDF scheduling) will be discussed in Chapter 4.6.

DC RL Short-Circuit Schematic Figure 1 is a model DC RL circuit with a switch for simulating a short-circuit. This schematic provides the requirements for the monitored, short-protected circuit: a voltage supply V_{in}, circuit load R_0, inductor L_0, and the optional, low-value resistor R_1 used for sensing current. If R_1 is not used, the resistance through inductor L_0 may be used to calculate current. Note that the short occurs via the switch around R_0.

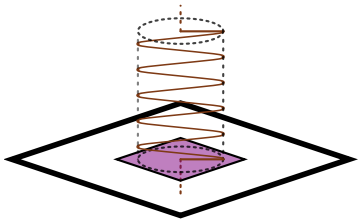


Figure 3: Board Space Consumed by an Inductor
The highlighted square circumscribing area A of the inductor represents the board space consumed by a solenoid-style air-core inductor.

As previously mentioned, in a manufactured circuit the spatial parameters and inductor value of L_0 in Figure 1 are expected to be fixed. During the design phase however, inductor parameters may be altered to increase space efficiency. These properties are afforded by Equation (4.1).

Board Space Consumed For ease of analysis, we assume the axis of the inductor is mounted perpendicular to the board as shown in Figure 3. Therefore, the board space consumed by the inductor is defined as the square that circumscribes area A of the inductor as seen in Figure 2:

$$A_{consumed} = \frac{4}{\pi} A \tag{4.4}$$

This definition of board space consumed will be referenced in Chapter 4.7 where the provided model is optimized for minimum board space consumption and volume.

Operating Current Model To simplify the analysis of DC RL circuits, we propose an Operating Current Model (OCM) for characterizing the circuit's maximum current, maximum voltage, and critical current. Hereafter, we model DC RL circuits as:

$$C = (\Gamma_I, I_{crit}) \tag{4.5}$$

where C is the circuit described using parameters Γ_I and I_{crit}. Γ_I is defined such that:

$$\Gamma_I = (\gamma_1, \gamma_2, ..., \gamma_n) \tag{4.6}$$

$$\gamma_i = (I_i, V_i) \qquad \text{for all } i = 1, ..., n \tag{4.7}$$

Here Γ_I is composed of n operating current sets γ_i where i is the index of an operating current set. Each operating current set is a 2-tuple of an operating current, I_i, and operating voltage, V_i. The resistance, R, of each operating current set can be solved using Ohm's Law and is not included.

I_{crit} is defined as the critical current of the system as determined by the user. The critical current represents the current value at which the system physically degrades or, in practice, the current value the user wishes to avoid reaching.

Two final derivable parameters I_{max} and V_{max}, are extracted from the OCM as follows:

$$I_{max} = \max_{i \in \Gamma_I}\{\gamma_i\} \quad \text{and} \quad V_{max} = \max_{i \in \Gamma_I}\{\gamma_i\} \tag{4.8}$$

Note that if $I_{crit} = I_{max}$, any current flow over I_{max} is assumed to be damaging and may not be prevented through this short protection model. Thus, we assume $I_{max} < I_{crit}$.

4.5 Methods of Protection

The intersection of electronics modeling and real-time systems begins with strategies for detecting and halting short-circuits. To detect a short in the generalized first-order DC RL circuit, as seen in Figure 1, we present two methods for detection:

1. a simple comparison of current, I, against maximum current, I_{max} and

2. a comparison of change in current, ΔI, against the maximum change in current, $\Delta I_{\max}$, allowed.

Both methods are applicable to any system which samples current of a monitored circuit. To prevent damage from shorts and fully utilize the following protection methods, power to the monitored circuit(s) must be removed immediately upon detection of a short. Without removal of power, these methods merely support detection and not protection.

Method 1: Maximum Operating Current The first method for detecting short circuits in a DC RL circuit is to identify when current rises above I_{max}. A current value above I_{max} indicates some malfunction caused current to rise above the maximum current in the OCM. When $I > I_{max}$, power to the monitored circuit should be disabled to protect the hardware.

Method 2: Maximum Change in Current The second method of detection requires observing the rate of change of current. From Equation (4.3) it is known that, given a constant voltage and resistance, the current will converge to $\frac{V}{R}$. During convergence, the

slope of ΔI approaches zero. The derivative of Equation (4.3) provides the value of $\Delta_{\max} I$:

$$\Delta I = \frac{\partial}{\partial t} I(t) = \frac{\partial}{\partial t}(\frac{V}{R}(1 - e^{-t\frac{R}{L}})) = \frac{V}{L} e^{-t\frac{R}{L}} \tag{4.9}$$

Assuming constant inductance, resistance, and voltage, the largest values of ΔI occur at $t = 0$ and $R = 0$ while excluding infinite inductance ($L \neq +\infty$). This leads to the following conclusions:

1. If $R \neq 0$, $\frac{dI(t)}{dt} = \frac{V}{L}$ instantaneously at time $t = 0$ only. Assuming no short has occurred, the change in current between any two consecutive current samples should be less than $\frac{V}{L}$. Formally, $\forall \delta > 0, t \geq 0, \frac{|I(t+\delta) - I(t)|}{\delta} < \frac{V}{L}$

2. If the change in current between two consecutive current samples is equivalent to $\frac{V}{L}$, then $R = 0$ and a short is occurring.

We can safely state the maximum ΔI through an inductor at any time t, including $t = 0$, is:

$$\Delta_{\max} I = \frac{V}{L}$$

Since all non-superconducting materials will provide some impedance, the $\Delta_{\max} I$ should have a threshold, ϵ, which serves as an implementation-specific offset. When consecutive current samples are compared, $\Delta_{\max} I = \frac{V}{L} - \epsilon$ should be used as the point of comparison. If a $\Delta I \approx \Delta I_{\max}$, it is likely that $R \approx 0$ and a short is occurring. A benefit of using $\Delta_{\max} I$ is the potential for short detection before I_{max} has been reached. In HSFs, there is no initial current ($I(0) = 0$) which could allow ΔI to approach $\Delta_{\max} I$ before I exceeds I_{max}.

Time-to-Detection The previous section discussed methods of detecting short-circuits which cover the logical requirements of the real-time short-protection task but did not provide any explicit temporal constraints. As previously mentioned, short-protection systems are time-sensitive and a valuable short-circuit detection occurs before critical current levels are reached. To do so requires determining the time taken for current to rise from its present value, I, to the critical current, I_{crit}. For the remainder of this paper, we deem this the *time-to-detection*.

4.5.1 Arbitrary Operating Voltage

If the OCM for the circuit in question contains a single operating voltage, it must hold that:

$$\forall V \in \Gamma_I, V = V_{max} \tag{4.10}$$

If Equation (4.10) holds, the *time-to-detection* is:

$$\delta(I, V) = \frac{I_{crit} - I}{V} \cdot L \tag{4.11}$$

where $\delta(I, V)$ is *time-to-detection*, I is current, I_{crit} is the critical current, V is voltage, and L is inductance. The function provides the time required for current to rise from I, to the critical current, I_{crit}, in a DC RL circuit with voltage V and inductance L. This function demonstrates that lower currents and voltages provide a lower *time-to-detection*.

However, to use this function for identifying the smallest time-to-detection in an OCM with multiple voltages would be optimistic. Suppose, for example, a DC RL circuit allows for two operating voltages, V_ℓ and V_h such that $V_\ell < V_h$. Suppose now that at the instant

a short-circuit occurs the operating voltage increases from V_ℓ to V_h. The value of $\delta(I, V_\ell)$, calculated before the short-circuit ocurred, will be an overestimate of the time required for I to exceed I_{crit}. To address this, we define the

4.5.2 Maximum Operating Voltage

Since using Equation (4.11) becomes optimistic when multiple operating voltages are involved, we can remove optimism by replacing V from Equation (4.11) with V_{max}:

$$\delta(I, V_{max}) = \frac{I_{crit} - I}{V_{max}} \cdot L \tag{4.12}$$

Equation (4.12) ensures a short combined with an instantaneous voltage change to V_{max} is still detected before reaching I_{crit}.

4.5.3 Minimum Time-to-Detection

To provide the worst-case time-to-detection, we use the maximum current, I_{max} and voltage, V_{max}, from the OCM. This *minimum time-to-detection* is defined as:

$$\delta_{min} = \delta(I_{max}, V_{max}) = \frac{I_{crit} - I_{max}}{V_{max}} \cdot L \tag{4.13}$$

This time frame represents time taken for current to rise from the maximum operating current, I_{max}, to the critical current, I_{crit}, with the highest voltage, V_{max}. This *minimum time-to-detection* will be used as a temporal constraint for the proposed real-time task.

4.6 Real-time System Model

Thus far, the spatial properties of an inductor have been related to its inductance. Thereafter, inductance is found to determine the maximum possible current rise at any given time, $\frac{V}{L}$ Amperes per second. This leads to the shortest time span over which a short would need to be detected, the *minimum time-to-detection* δ_{min}. Using δ_{min}, we provide a real-time system short-circuit protection task and its timing requirements. Before doing so, we present a real-time background on the sporadic task model and uniprocessor EDF scheduling.

Sporadic Task Model In real-time systems, *sporadic tasks* are defined by a *worst-case execution time* (WCET) e_i, relative deadline d_i, and minimum period p_i. The relative deadline is the time between each job arrival and its deadline. The minimum period is the smallest time between successive job arrivals. We use a sporadic task to model the short-circuit protection task [31]. The proposed sporadic task will have an *implicit deadline* where a new job of the short-circuit task T_{scd} may arrive at the absolute deadline of the previous job.

For our short-circuit task T_{scd}, the execution time depends on the short-circuit protection algorithm used, processor speed, and ADC conversion time. Thus, the execution time is not assessed here but modeled as e_{scd}.

The minimum period, however, is derived from Equation (4.12) which provides a scaling *time-to-detection*. The minimum period of the sporadic task must be half the value of the *minimum time-to-detection*. This is required to ensure one full job of T_{scd} is completed strictly after the short-circuit begins. This requirement is demonstrated in Figures

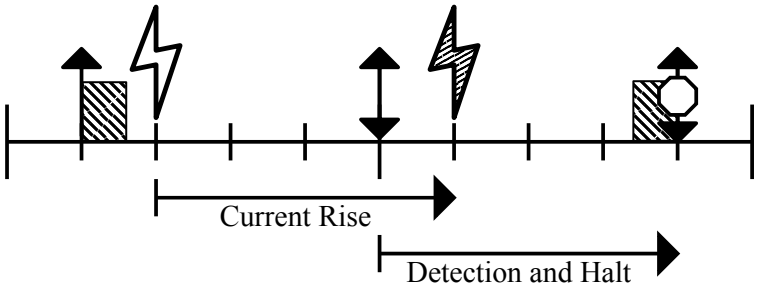


Figure 4: Failed Short Protection Schedule
T_{scd} executing with $p_{scd} = \delta(I_{max}, V_{max}) = 4$.
The short begins at $t = 2$ and I_{crit} is reached at $t = 6$ before T_{scd} can halt the short.

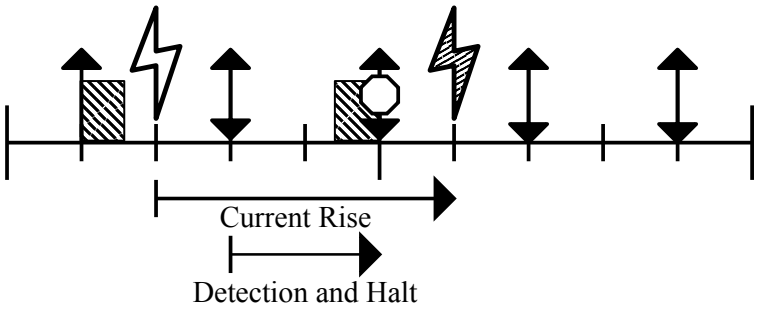


Figure 5: Successful Short Protection Schedule
T_{scd} executing with $p_{scd} = \frac{\delta(I_{max}, V_{max})}{2} = 2$.
The short begins at $t = 2$ and is halted by T_{scd} at $t = 5$ before I_{crit} is reached.

4 and 5 where T_{scd} executes with a period equal to δ_{min} and $\frac{\delta_{min}}{2}$. Recall that δ_{min} time units after a short, the current level has reached a I_{crit}. Each sporadic job is triggered $\delta(I, V_{max})/2$ seconds after the release of the preceding job, allowing release times to scale with current. The temporal requirements may be converted into a real-time sporadic task $\mathrm{T}_{scd} = (e_{scd}, p_{scd})$ with the following parameters:

$$e_{scd} = \textit{worst case execution time} \text{ of short-circuit algorithm} \tag{4.14}$$

$$p_{scd} = \frac{\delta(I_{max}, V_{max})}{2} \tag{4.15}$$

Preemptive Uniprocessor EDF Scheduling With T_{scd} defined, we derive its utilization under preemptive uniprocessor EDF scheduling. According to [28], a set of implicit-deadline sporadic tasks is schedulable with EDF if and only if:

$$\sum_{i=1}^{n} \frac{e_i}{p_i} \leq 1 \tag{4.16}$$

where i is the index of a task in the set, e_i is the execution time, and p_i is the period. Since we use a single, sporadic task with implicit deadlines, the utilization for T_{scd} is:

$$U(\mathrm{T}_{scd}) = \frac{2 \cdot e_{scd}}{\delta(I, V_{max})} \tag{4.17}$$

Using the *minimum time-to-detection*, Equation (4.17) becomes:

$$U(\mathrm{T}_{scd}) = \frac{2 \cdot e_{scd}}{\delta(I_{max}, V_{max})} \tag{4.18}$$

Substituting in Equations (4.11) and (4.1) gives:

$$U(\mathrm{T}_{scd}) = (\frac{2 \cdot e_{scd}}{(I_{crit} - I_{max})} \cdot V_{max} \cdot \frac{\ell}{\mu N^2 A}) \tag{4.19}$$

Equation 4.19 relates the inductor spatial parameters to T_{scd} utilization.

4.7 Model Optimization

Having related the board space consumed by an inductor and the real-time utilization under EDF schedulability, we propose an optimal solution for fixed values on either end of the relationship. Given a fixed utilization we propose a minimized board space consumption. Given a fixed allowable board space, we propose a minimized real-time utilization. Before optimization analysis of the model, we clarify the optimal inductor orientation for a given prism.

Optimal Inductor Orientation In practice, hardware-software co-design is constrained by real-world factors such as size, weight, power, and cost. In this paper, we focus on space as a constraint on our proposed model. Given a fixed volume of space to implement the proposed short-protection system, we must consider the optimal orientation of our solenoid-style air-core inductor inside the fixed volume which maximizes inductance. To find the inductor orientation providing the highest inductance for a given space, we consider two constraints:

1. The area A from Equation (4.1) requires a square area with regard to board space.

2. The board space must be defined in three dimensions: a length ℓ, width w, and height h.

Allowable board space is therefore defined as a set:

$$P = \{\ell, w, h\} \tag{4.20}$$

This set provides the prism dimensions in which the inductor resides. Independent of the prism's orientation, $A_{consumed}$ must be the smallest square to circumscribe the area A of the inductor. The largest square face on which the inductor's area, A, can be placed is limited by the median dimension in P. Thus, the square of the median is used to fit the largest possible square area:

$$A_{consumed} = median(P)^2 \tag{4.21}$$

The only remaining dimension is deemed the length of the inductor:

$$\ell = min(P) \tag{4.22}$$

A visualization of possible orientations can be found in Appendix 13 along with an explicit example. As previously mentioned and validated in Section 4.3.2, the *Assumption of Constant Turn Density* applies to ℓ and is required for the remaining model optimization.

Fixed Board Constraints Having defined equations for consumed area in terms of the volume allotted for the inductor, we address the first optimization problem where the board space is constrained. This approach is useful in situations where the embedded application, or the region of space allotted for short-circuit protection hardware, is restricted in size. Suppose allotted board space is restricted to the prism:

$$P = \{\ell, w, h\}$$

where each element of the set is defined in meters. Combining Equations (4.4) and (4.21) for the area of the inductor A gives:

$$A = \frac{\pi}{4} A_{consumed} = \frac{\pi}{4} \cdot median(P)^2$$

By substitution of Equation 4.23 into Equation (4.19), the utilization requirement becomes:

$$U(\mathrm{T}_{scd}) = \frac{2 \cdot e_{scd} \cdot V_{max} \cdot min(P)}{(I_{crit} - I_{max}) \cdot \mu \cdot N^2 \cdot \frac{\pi}{4} \cdot median(P)^2} \tag{4.23}$$

The equation above gives us a minimum utilization requirement for meeting short-circuit protection requirements given the constrained board space and OCM - which provides I_{crit} and I_{max}.

Fixed Utilization Having addressed the fixed-volume constraint, we not address the second approach by fixing the maximum utilization allowed short-circuit protection process. This approach is useful in situations where the microprocessor executing T_{scd} is responsible for other tasks which inherently limit $U(\mathrm{T}_{\mathrm{scd}})$. Suppose the allotted utilization is u. Relying on Equation (4.18) we find the minimum time-to-detection $\delta(I_{max}, V_{max})$ is solved as:

$$\delta_{min} = \delta(I_{max}, V_{max}) = \frac{2 \cdot e_{scd}}{u}$$

Applying Equation (4.13) and isolating L we find:

$$L = \frac{2 \cdot e_{scd} \cdot V_{max}}{u \cdot (I_{crit} - I_{max})}$$

Substituting Equation (4.1) in for L and isolating inductor spatial parameters results in:

$$\frac{A}{\ell} = \frac{2 \cdot e_{scd} \cdot V_{max}}{u \cdot (I_{crit} - I_{max}) \cdot \mu \cdot N^2}$$

Finally, substituting the prism area consumed (Equation 4.21) and prism length (Equation 4.22) into Equation 4.24 above gives:

$$\frac{median(P)^2}{min(P)} = \frac{4}{\pi} \cdot \frac{2 \cdot e_{scd} \cdot V_{max}}{u \cdot (I_{crit} - I_{max}) \cdot \mu \cdot N^2} \tag{4.24}$$

This result indicates the air core inductor used in the DC RL circuit must have a minimum allotted board space defined by P which satisfies the above equation.

The model optimizations above highlight how fixed board space can be used to prescribe a minimum utilization and vice versa. We now seek to validate our short-circuit protection model the next chapter where we discuss our experiments and subsequently our results.

4.8 Experiments

To implement and validate the application of the software-based approach, we conducted four separate experiments. The following section presents the experiment setup and outlines the conducted experiments. Additionally, the projected utilization for the experiments is presented.

Experiment Outline The experiments conducted measured both FUL and HSF short-circuits focusing primarily on FUL short-circuits as they have a smaller time-to-detection. Table 1 highlights the variations between experiments which can be summarized as follows: Experiment 1 examines the performance of our approach under FUL short-circuits in terms of detection latency and maximum current reached at the time of detection. Experiment 2 examines the performance under HSF short-circuits also in terms of latency and maximum current reached at the time of detection. Experiment 3 demonstrates how fixing utilization and varying inductance under FUL short-circuits relates to current at the time of detection. Experiment 4 demonstrates the same relationship but with varying utilization and fixed inductance.

Experiment	Fault Type	Demonstrates
1	FUL	Baseline
2	HSF	Lower initial I
3	FUL	Scaling L
4	FUL	Scaling U

Table 1: Experiment Descriptions

Setup Aside from the variations described above, all experiments were conducted as described here. Short-circuit protection for each experiment was performed on Microchip's

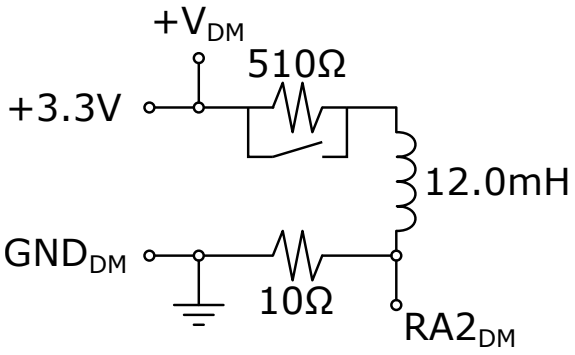


Figure 6: Experiment Setup Schematic
Experiment Setup Schematic

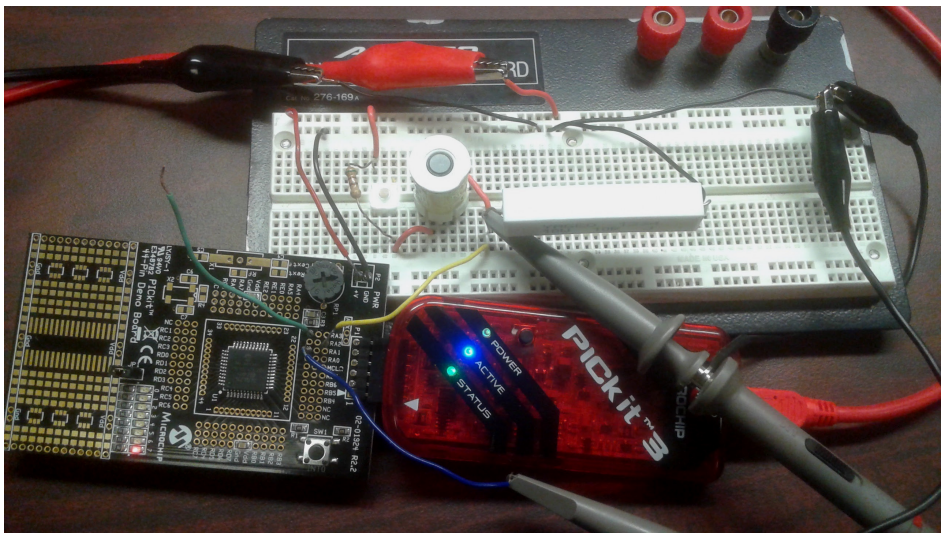


Figure 7: Experiment Setup Image

DM164103-4 demo board with a PIC18F45K20 CPU. The demo board and programmer were selected for their relatively low cost to encourage reproducibility. Figure 6 depicts the experimental circuit schematic. In the schematic, subscripts *DM* refer to connections made to the Microchip demo board. The setup also requires attaching oscilloscope probes to ports RA2 in Figure 6 and RA1 on the demo board. Figure 7 depicts the proper connection of all materials in Experiment 1. All probe connections route off-camera to the oscilloscope. An extended list of non-trivial materials used in the experiments can be found in Appendix 13.

The circuit was run in advance of experiments to identify experimental operating currents and voltages and their respective maximums. This process could be avoided by analyzing the tolerance values for all components used but was explicitly measured here to provide exact values. The OCM for the setup using solid, 22 American Wire Gauge (AWG) copper wire was:

$$C = (\Gamma_I, 150mA) \quad \Gamma_I = (\gamma_0) \quad \gamma_0 = (7.74mA, 3.3V)$$

Although the resistor R_1 pictured in 7 is a 10W power resistor capable of handling higher current, we defer to the power rating of a conventional though-hole power resistor which we assume to be 0.5 Watts. Since 150 mA through a 0.5 Watt resistor exceeds the power rating at an operating voltage of 3.3 V, 150 mA is considered the critical current I_{crit} for the circuit.

After deriving the OCM from the circuit, Equation (4.8) provides the maximum operating current and voltage:

$$I_{max} = 7.74mA \quad V_{max} = 3.3V$$

Note that the critical current is 150mA meaning current above 7.74mA is considered unexpected behavior while current at or above 150mA is damaging. Having constructed the mathematical model of our system, we may now project the utilization requirement in the following section.

Projected Utilization Using the OCM derived from the experiment setup, we apply Equations 4.17 and 4.18 as the scaling utilization and minimum utilization. For this projected

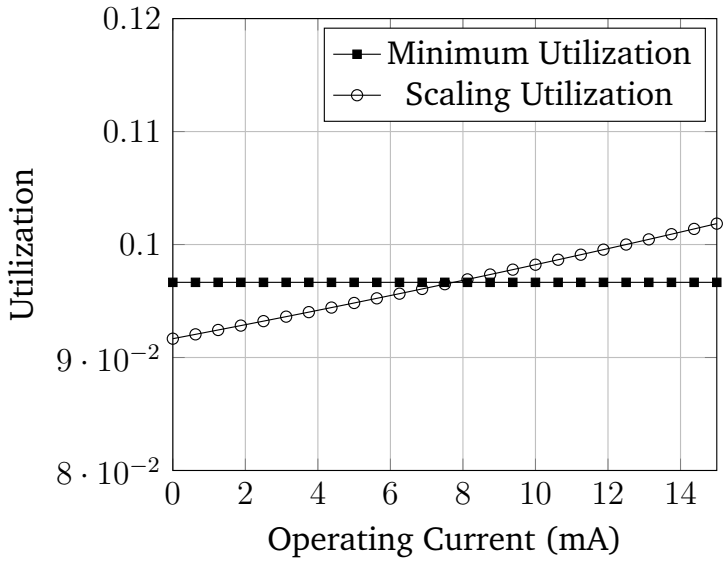


Figure 8: Simulated Utilization vs. Operating Current

utilization, our protection method required an execution time of 25 microseconds. Figure 8 depicts the projection of fixed minimum utilization and the scaling utilization. The minimum utilization indicates the expected minimum utilization required to detect a short-circuit in the experiment. The scaling utilization represents the minimum utilization required at *any* given value of I as derived from Equation (4.17). These projected utilizations indicate the provided short-protection approach could safely detect and halt a short-circuit before critical current levels are reached at a uniprocessor utilization of under 0.1. The scaling utilization curve indicates the lower minimum utilization requirement for systems with lower operating currents.

Fault	R (Ohm)	I_c (mA)	Δt (us)	U(T_{scd})
FUL	510	42.400	159.79	0.1316
FUL	510	38.800	142.59	0.1316
FUL	510	20.000	50.39	0.1316
HSF	0	35.200	149.60	0.1316
HSF	0	35.200	72.01	0.1316
HSF	0	35.200	175.20	0.1316

Table 2: FUL and HSF Comparison

4.9 Results

Experiments 1 and 2: FUL and HSF Detection To validate the hardware-software co-design approach, two short-circuit fault types were induced in separate experiments. As shown in Table 1, Experiment 1 induced an FUL short while Experiment 2 inducted an HSF short. Each experiment was run three times with all data provided in Table 2. In the table, I_c is the current at the time of detection and Δt is the latency between short-circuit and detection. Three runs per fault type was selected due to time required to manually induce the short and reset each experiment.

Both experiments demonstrate successful detection of FUL and HSF short-circuits before current rises above I_{crit}. Both experiments also demonstrate latency one-tenth of the minimum time to detection. For experiment 1, using the FUL, the short-circuit was manually induced while the circuit was powered. In contrast, experiment 2 required removal of load resistor R_0 from Figure 6. This ensures the short-circuit begins immediately upon powering the circuit. As a result of this change, there is no initial current in the HSF as there is with the FUL short.

In addition to the comparison table, an example FUL waveform is shown in Figure 9 as

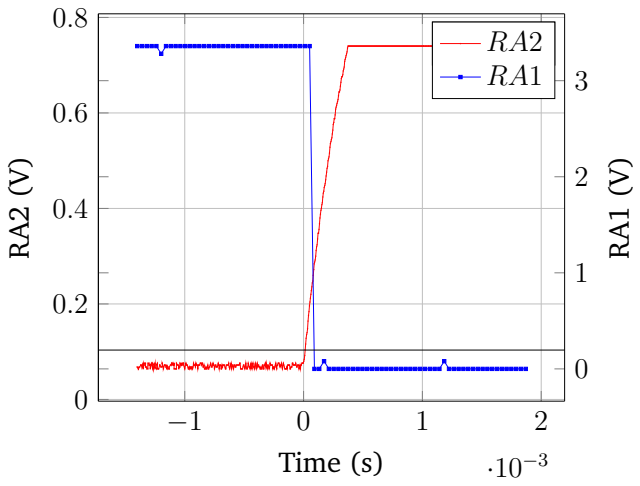


Figure 9: FUL Short Waveform

it was the most used short-circuit fault test. The plot shows the circuit current, sampled on port RA2, rise immediately after the short-circuit. The short-circuit detection signal, output on port RA1, shows the output signal voltage drop to zero indicating the detection of a fault and the cutting of power by the microprocessor. Due to its similar nature, the waveform for the HSF experiment is excluded. The current at the time of detection for both fault types in all runs did not exceed I_{crit}, 150mA; thus, the proposed short-circuit protection succeeded in safely mitigating damage from both FUL and HSF short-circuits.

Experiments 3 and 4: Inductance and Utilization Scaling Experiments 3 and 4 focused on demonstrating the potential for scaling inductance and utilization as co-design parameters are changed. In Experiment 3, the schematic depicted in Figure 6 was used with varying sized inductors as opposed to experiment 1 and 2 which used fixed values. For each inductance value, L, a FUL short was induced. Real-time utilization was maximized at 100%, $U = 1.0$, to demonstrate the broadest range of inductors. As seen in Figure 10, a decrease in inductance given a constant utilization results in a higher current at the

time of detection. This trend validates the relationship presented in Equation 4.19. In conjunction with the optimal inductor orientation analysis, the results show short-circuit detection at lower current levels can be traded for a smaller inductor consuming less board space.

A similar procedure was used in Experiment 4 but with fixed inductor sizes and varying real-time utilizations. In Experiment 4, the schematic in Figure 6 is used but with an inductance of $L = 12$mh. This large value of inductance was selected to demonstrate the broadest range of real-time utilizations. The results, presented in Figure 11, show a decrease in utilization given a constant inductance leads to a higher current at the time of detection. This again validates the relationship provided in Equation (4.19).

The results of Experiment 3 and Experiment 4 indicate that $U(\mathrm{T}_{scd})$ can be traded with inductance L in our model to maintain a stable current at the time of detection. It follows that inductor size and board space consumed by the proposed short-circuit protection model may be traded with the utilization of the provided software-based protection approach.

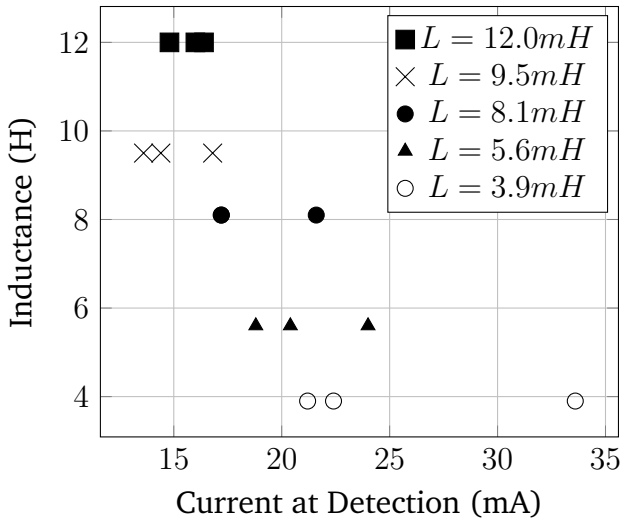


Figure 10: Experiment 3 Results: Scaling Inductance vs Current at Detection

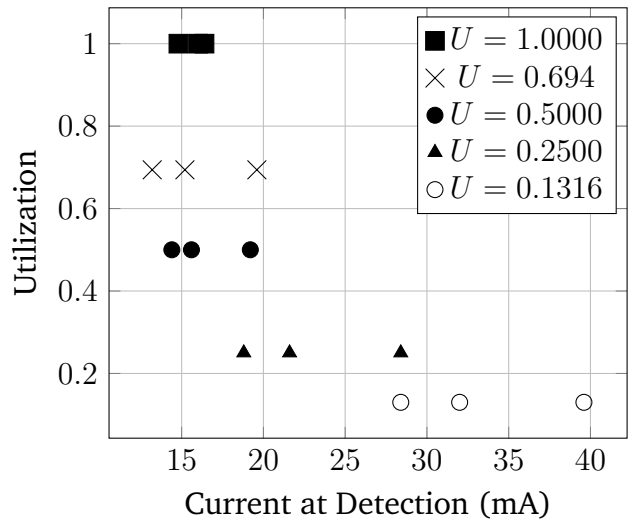


Figure 11: Experiment 4 Results: Scaling Utilization vs Current at Detection

4.10 Conclusion

This work provides a novel solution for hardware-software co-design of real-time software-based short-circuit protection systems – cyber-physical systems in which the inductive properties of a DC RL circuit are leveraged to construct a sporadic, real-time short-circuit protection task. We established a relationship between utilization and board space consumed by the air-core, solenoid-style inductor placed in-circuit which can be optimized for both fixed inductor volume and fixed uniprocessor utilization.

Like preceding works on engine control [2] and thermal-aware systems [18], this work demands further investigation into the real-time control of physical systems demonstrating dynamic behavior. Further study on the energy and performance trade-off between hardware, software, and physical system dynamics is reserved for future work.

CHAPTER 5 WORST-CASE DEMAND OF ENGINE CONTROL TASKS

CHAPTER 6 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 [13]. 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 [29] 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 [13]

and a constant execution time as shown in Fig. 12.

To determine whether an AVR task is schedulable using EDF, the demand bound function (dbf) is often used [6, 5] 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. [30] proposed an exact analysis, using the Digraph model [33], 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:

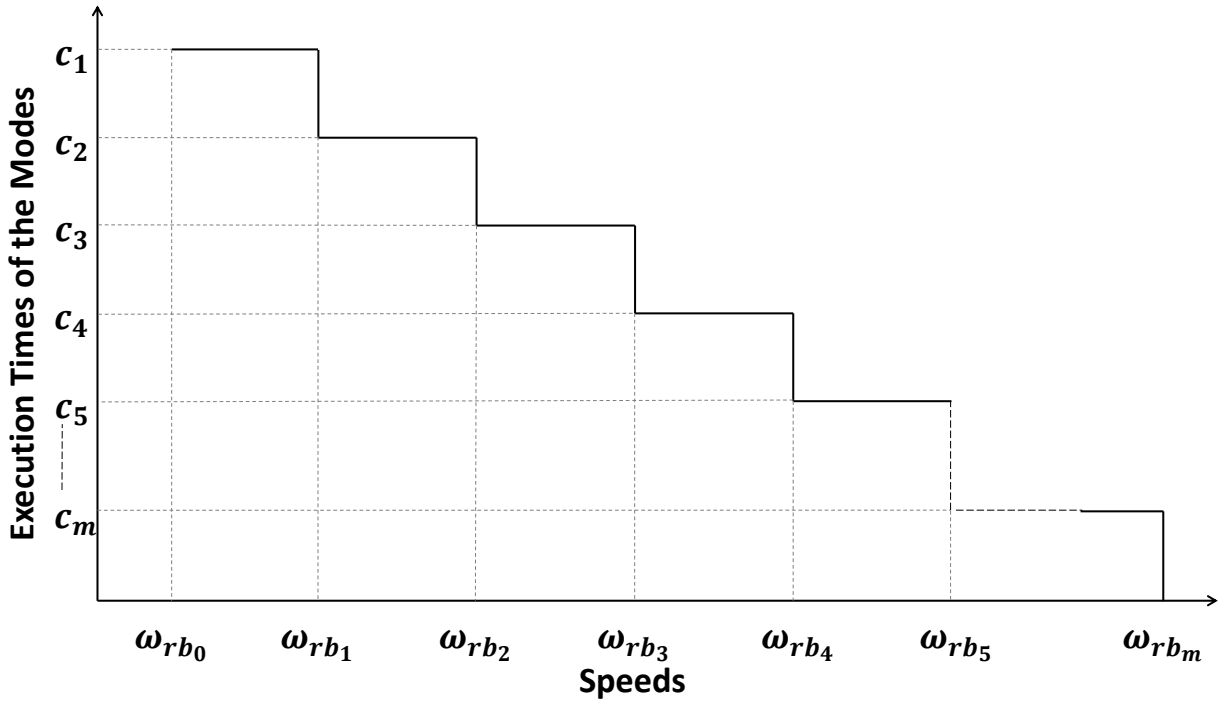


Figure 12: Different modes of an AVR task where c_i and ω_{rb_i} are the execution time and the right boundary speed of the i^{th} mode, respectively.

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 [30] in terms of computation time.

The rest of the paper is organized as follows. In Section 7, we review key existing work pertaining to AVR tasks. In Section 8, we introduce the system model, discuss our assumptions and formally present the problem. In Section 9, 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 10 and describe the dominant sequence set in Section 11. Experimental results are presented in Section 12 and the paper concludes in Section 13.

CHAPTER 7 RELATED WORK

Usage of multiple worst-case execution times and periods for engine-controlled tasks was first studied by Kim et al. [26], 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 [6] and EDF scheduling [5]. 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. [30]. In one of the first works on EDF scheduling of AVR tasks, Guo and Baruah [16] 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. [3] proposed two methods to estimate the angular speed of the crankshaft. In a recent paper [4], 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. [7], 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. [30] 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. [30] to significantly reduce the computation time.

CHAPTER 8 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.

8.1 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

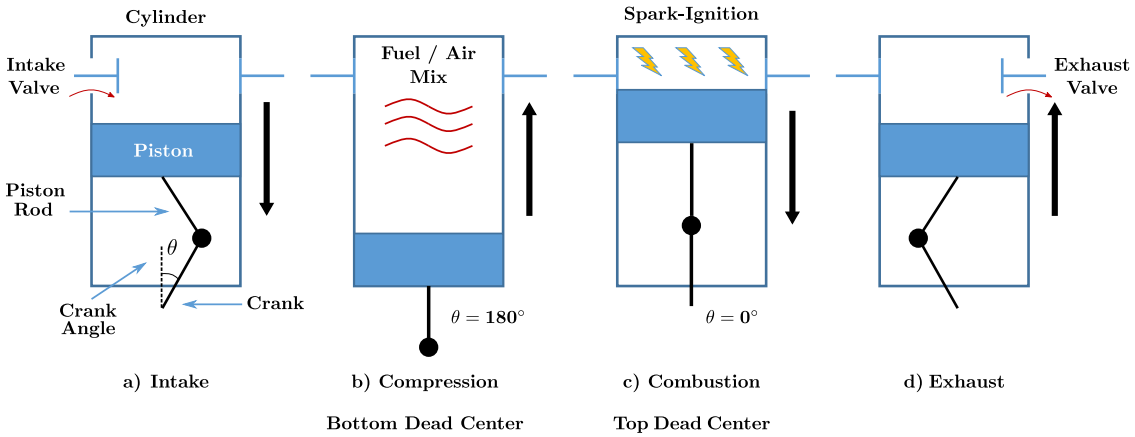


Figure 13: Different stages of fuel ignition in a vehicle.

shown in Fig. 13. 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 [13]. 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. 12.

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 8). The minimum and 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 .2.

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 [35, 8]*

$$\Omega(\omega(t), \alpha, \Delta\theta) = \sqrt{\omega(t)^2 + 2\alpha\Delta\theta}. \tag{8.1}$$

Similar to the work by Mohaqeqi et al. [30] 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 8.1, 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 [30],

$$\Omega_n(\omega(t), \alpha) = \sqrt{\omega(t)^2 + 2n\alpha}. \tag{8.2}$$

Biondi et al. [6] 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.

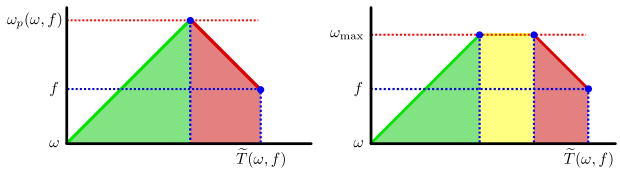
8.2 Minimum Job Inter-arrival Times

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 [6, 5, 8], we consider the possibility of acceleration variations between two job releases similar to the work by Mohaqeqi et al. [30].

The minimum inter-arrival time equation by Mohaqeqi et al. [30] 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,

$$\omega_p(\omega, f) = \frac{\sqrt{2\omega^2 + 2f^2 + 2\alpha_{\max}}}{2}. \tag{8.3}$$

However, if $\omega_p > \omega_{max}$, the crankshaft has to maximally accelerate from ω to ω_{max}, stay



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

Figure 14: 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.

at that speed for some time and then maximally decelerate to f. The two cases are defined below and presented in Figure 14:

$$\widetilde{T}(\omega, f) = \tag{8.4}$$

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

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.*

8.3 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** [4] 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:

$$D_{\omega(t)}(t_a, t_b) = \sum_{i:(t_i \geq t_a) \wedge (t_i + \tilde{d}(\omega(t_i)) \leq t_b)} c(\omega(t_i)). \tag{8.5}$$

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

$$\mathsf{dbf}_{\omega(t)}(\delta) = \max_{t' \geq 0} \{D_{\omega(t)}(t', t' + \delta)\}. \tag{8.6}$$

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 8.6:

$$\mathsf{dbf}(\delta) = \max_{\omega(t) \in \mathbb{W}} \{\mathsf{dbf}_{\omega(t)}(\delta)\}. \tag{8.7}$$

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 8.4), 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. [30] 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 8.7.

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., $\{j, j+1, \ldots, k\}$.*

8.4 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 8.7), 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.

CHAPTER 9 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 10 and 11. Briefly, *dominant speed sequences* are sequences of job release speeds whose demand coincides with the value of Equation 8.7. These dominant speed sequences are non-decreasing (see Section 10.2, Lemma 1) and start at right boundary speeds (see Section 10.3, 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) [11, 24]. Fig. 15 provides a visual example representation of our problem as a

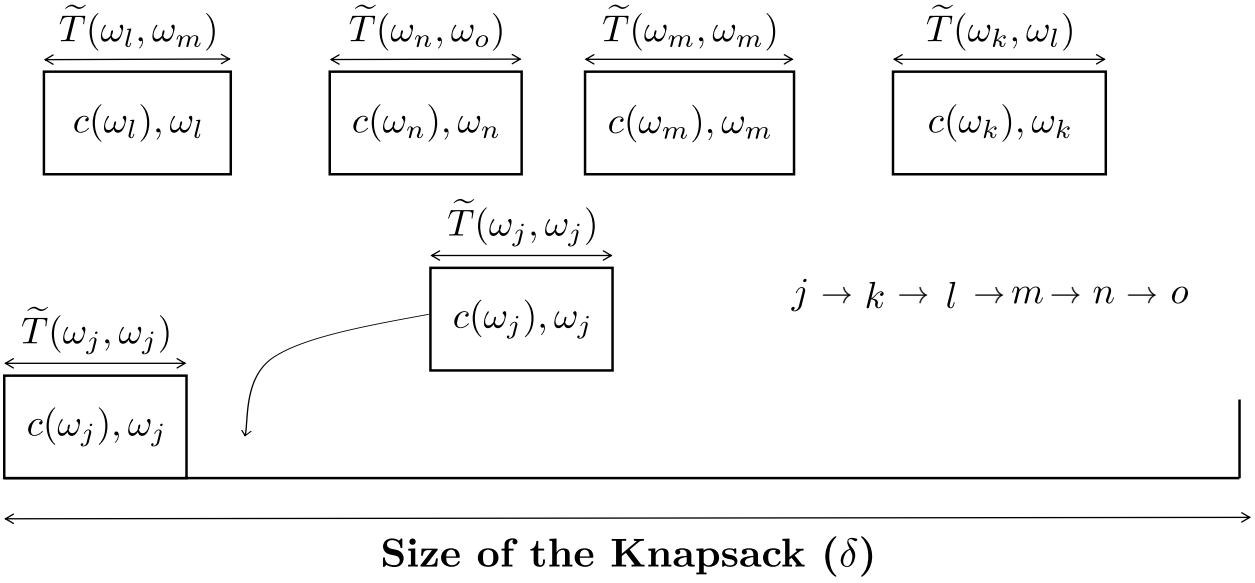


Figure 15: Example items for our knapsack problem. A job that is higher in the precedence relation is preceded by a job lower in the relation.

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. 16. 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 [25]. An edge between any two vertices denotes that the two speeds are reachable from one another and is represented by $(j, k) \in A_I$. Formally, our BPCKP can be

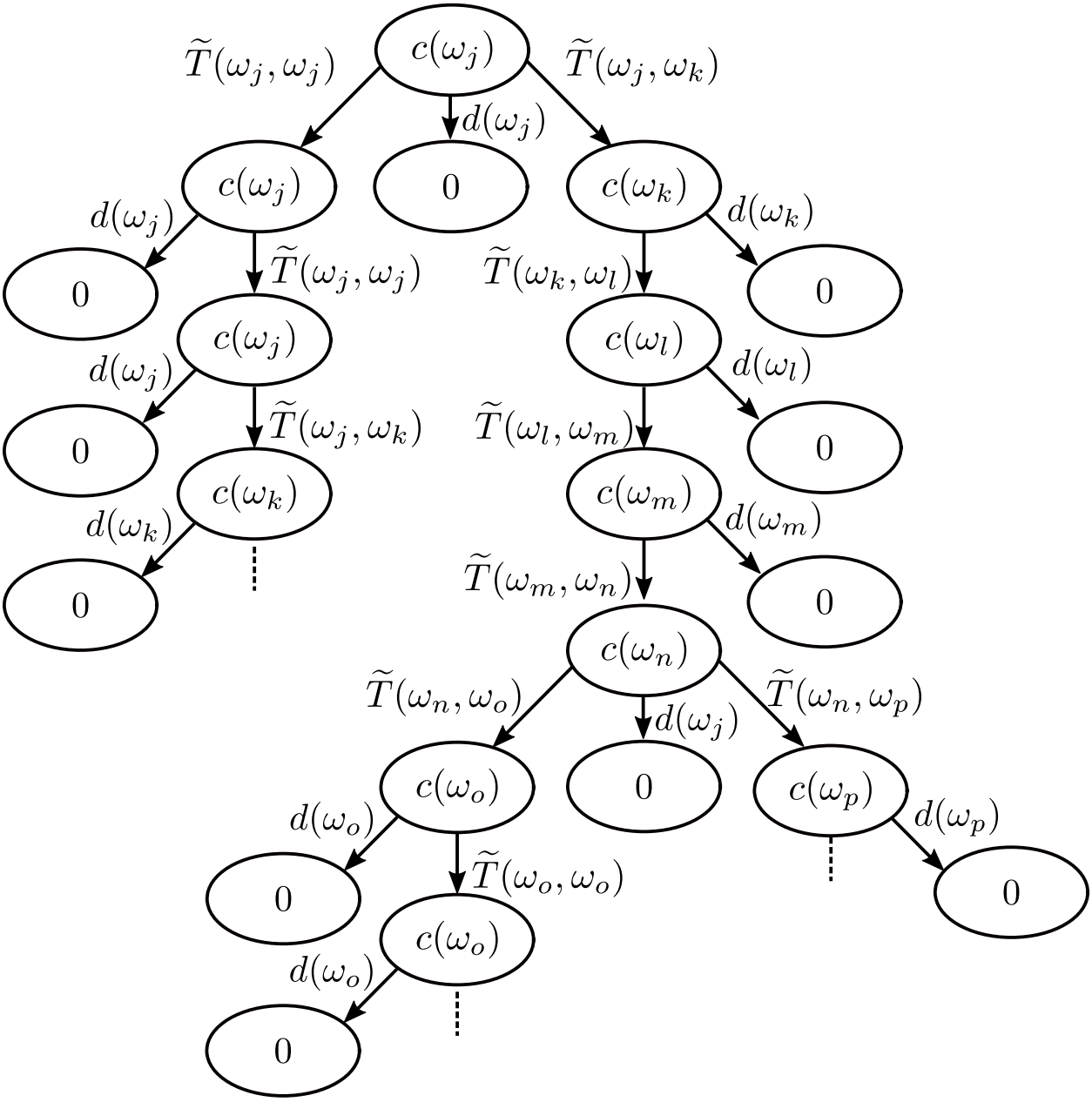


Figure 16: Precedence constraints among jobs are expressed using out-trees. Nodes represent WCETs for the initial speeds and arrows represent minimum inter-arrival times. The tree demonstrates the various precedence relations and possible paths. Leaves represent completion of the parent job without adding subsequent jobs (i.e. items). Furthermore, each of the leaves has zero execution since its parent is the final job included in the knapsack.

expressed as follows:

$$\underset{x}{\text{maximize}} \quad \sum_{j} \sum_{r=1}^{M_\delta} c(\omega_j) \cdot x_j^r \tag{9.1}$$

$$\text{subject to} \quad \sum_{j,k|(j,k)\in A_I} \sum_{r=1}^{M_\delta} \tilde{T}(w_j, w_k) \cdot x_k^r \leq \delta \tag{9.2}$$

$$\sum_{k|(j,k)\in A_I} x_k^1 \leq 1, \; \forall j \tag{9.3}$$

$$x_j^1 \geq x_k^1, \; \forall (j,k) \in A_I \tag{9.4}$$

$$x_j^r \geq x_j^{r+1}, \; \forall j, r \in \{1, 2, 3, ..., M_\delta - 1\} \tag{9.5}$$

$$x_j^r \in \{0, 1\}, \; \forall j, r \in \{1, 2, 3, ..., M_\delta\} \tag{9.6}$$

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 9.1 maximizes total demand. Equation 9.2 requires all deadlines of selected jobs fall within the δ-length interval. Equation 9.3 permits at most one child node of each parent node to be added to the knapsack. Equation 9.4 enforces the precedence constraint such that no child node may be added without its parent. Equation 9.5 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 11). The variable D_w (Equation 8.5) 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
```

CHAPTER 10 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. [30] 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 8.7). 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.

10.1 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 8.4) 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 [30]. 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* $\omega, f \in [\omega_{\min}, \omega_{\max}]$*,*

$$\widetilde{T}(\omega, f) > 0. \tag{10.1}$$

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* $\omega, f \in [\omega_{\min}, \omega_{\max}]$*,*

$$\frac{\partial \widetilde{T}(\omega, f)}{\partial \omega} \leq 0 \quad \textit{and} \quad \frac{\partial \widetilde{T}(\omega, f)}{\partial f} \leq 0. \tag{10.2}$$

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

$$\frac{\partial \widetilde{T}(\omega, f)}{\partial \omega} = \begin{cases} \frac{\frac{2\omega}{\sqrt{4\alpha_{\max}+2f^2+2\omega^2}}-1}{\alpha_{\max}} & \text{if } \omega_p(w,f) \leq \omega_{\max} \\ \frac{\omega}{\omega_{\max}\alpha_{\max}} - \frac{1}{\alpha_{\max}} & \text{if } \omega_p(w,f) > \omega_{\max} \end{cases} \tag{10.3}$$

$$\frac{\partial \widetilde{T}(\omega, f)}{\partial f} = \begin{cases} \frac{\frac{2f}{\sqrt{4\alpha_{\max}+2f^2+2\omega^2}}-1}{\alpha_{\max}} & \omega_p(w,f) \leq \omega_{\max} \\ \frac{f}{\omega_{\max}\alpha_{\max}} - \frac{1}{\alpha_{\max}} & \omega_p(w,f) > \omega_{\max} \end{cases} \tag{10.4}$$

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:*

$$\frac{\partial \tilde{d}(\omega)}{\partial \omega} < 0 \quad \textit{and} \quad \frac{\partial \tilde{d}(\omega)}{\partial \omega} \leq \frac{\partial \widetilde{T}(\omega, f)}{\partial \omega}. \tag{10.5}$$

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 $\omega \in [\omega_{\min}, \omega_{\max}]$.

$$\tilde{d}(\omega) = \begin{cases} \frac{\sqrt{\omega^2+2\alpha_{\max}}-\omega}{\alpha_{\max}} & \tilde{d}_p(w) \leq \omega_{\max} \\ -\frac{\omega}{\alpha_{\max}} + \frac{\omega^2+\omega_{\max}^2}{2\omega_{\max}\alpha_{\max}} + \frac{1}{\omega_{\max}} & \tilde{d}_p(w) > \omega_{\max} \end{cases} \tag{10.6}$$

$$\tilde{d}_p(\omega) = \sqrt{\omega^2 + 2\alpha_{\max}\,\theta} \tag{10.7}$$

where $\tilde{d}(\omega)$ is derived from Equation 8.1 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 $\tilde{d}_p(w) > \omega_{\max}$.

$$\frac{\partial \tilde{d}(\omega)}{\partial \omega} = \begin{cases} \frac{\frac{\omega}{\sqrt{\omega^2+2\alpha_{\max}}}-1}{\alpha_{\max}} & \tilde{d}_p(w,f) \leq \omega_{\max} \\ \\ \frac{\omega}{\omega_{\max}\alpha_{\max}} - \frac{1}{\alpha_{\max}} & \tilde{d}_p(w,f) > \omega_{\max} \end{cases} \tag{10.8}$$

where $\tilde{d}$ is derived by replacing f with Equation 8.1 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,

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

□

10.2 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 $\forall\, \ell,\, k \,|\, 1 \leq \ell \leq i < k \leq n$:

$$s_k > \Omega_1(s_\ell, \alpha_{\max}). \tag{10.9}$$

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.

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:*

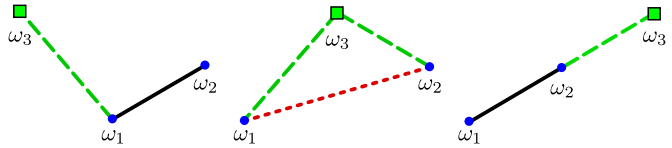
1. Leading Injection*: If ω is the $(k+1)^{th}$ smallest item of S and it becomes the first element of $S(k+1)$. That is, $\omega_1^{(k+1)}$ equals ω and $\omega_{\ell+1}^{(k+1)}$ equals $\omega_\ell^{(k)}$ for all $\ell = 1, \ldots, k$. Figure 17(a) shows an example leading injection.*

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 17(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 17(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



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

Figure 17: 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.

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 $S = (\omega_1, \ldots, \omega_n)$:

$$d(S) = \sum_{i=1}^{n-1} \widetilde{T}(\omega_i, \omega_{i+1}) + \tilde{d}(\omega_n). \tag{10.10}$$

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,

$$\Delta(S, k, k+1) = d(S(k+1)) - d(S(k)). \tag{10.11}$$

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:

$$\Delta(S, k, k+1) = \begin{cases} \Delta_L(S, k, k+1) & \text{if leading,} \\ \Delta_F(S, k, k+1) & \text{if final,} \\ \Delta_I(S, k, k+1) & \text{if internal,} \end{cases} \tag{10.12}$$

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* $k = 1, \ldots, n-1$*:*

$$\Delta_L(S,k,k+1) \geq \Delta_F(S_A,k,k+1). \tag{10.13}$$

Proof. In this property, we have a leading injection into $S(k)$. However, observe that $\widetilde{d}(s_{k+1}) \leq \widetilde{d}(s_k)$ 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 10.13 and the lemma is proved. □

Property 8 (Internal Injection Suboptimality)**.** *For any valid, finite sequence* $S = (\omega_1, \ldots, \omega_n)$*,*

for all $k = 1, \ldots, n-1$*:*

$$\Delta_I(S, k, k+1) \geq \Delta_F(S_A, k, k+1). \tag{10.14}$$

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:

$$\begin{aligned} & \widetilde{T}(s_{k-1}, s_k) + \widetilde{T}(s_k, s_k) + \tilde{d}(s_k) \\ & \quad = \widetilde{T}(s_{k-1}, s_k) + \widetilde{T}(s_k, s_k) + \tilde{d}(s_k) \\ \Rightarrow \quad & \widetilde{T}(s_{k-1}, s_k + \epsilon) + \widetilde{T}(s_k + \epsilon, s_k) + \tilde{d}(s_k) \\ & \quad \geq \widetilde{T}(s_{k-1}, s_k) + \widetilde{T}(s_k, s_k + \epsilon) + \tilde{d}(s_k + \epsilon) \end{aligned} \tag{10.15}$$

Setting $\epsilon = s_{k+1} - s_k$ and substituting into the above inequality of Equation 10.15, we get the following:

$$\begin{aligned} & \widetilde{T}(s_{k-1}, s_{k+1}) + \widetilde{T}(s_{k+1}, s_k) + \tilde{d}(s_k) \\ & \quad \geq \widetilde{T}(s_{k-1}, s_k) + \widetilde{T}(s_k, s_{k+1}) + \tilde{d}(s_{k+1}) \\ \Rightarrow \quad & \widetilde{T}(s_{k-1}, s_{k+1}) + \widetilde{T}(s_{k+1}, s_k) - \widetilde{T}(s_{k-1}, s_k) \\ & \quad \geq \widetilde{T}(s_k, s_{k+1}) + \tilde{d}(s_{k+1}) - \tilde{d}(s_k) \\ \Rightarrow \quad & \widetilde{T}(\omega_{j-1}^{(k)}, s_{k+1}) + \widetilde{T}(s_{k+1}, \omega_j^{(k)}) - \widetilde{T}(\omega_{j-1}^{(k)}, \omega_j^{(k)}) \\ & \quad \geq \widetilde{T}(s_k, s_{k+1}) + \tilde{d}(s_{k+1}) - \tilde{d}(s_k) \end{aligned} \tag{10.16}$$

The last inequality (which implies Equation 10.14 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 10.16 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:*

$$d(S) \geq d(S'). \tag{10.17}$$

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:

$$d(S(k)) \geq d(S'(k)) \geq d(S_A(k)). \tag{10.18}$$

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 10.18 is vacuously true.

Induction Hypothesis: Assume that Equation 10.18 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 10.18 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. □

10.3 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.

□

10.4 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

$$\omega'_j = \min(\omega_{rb_i}, \Omega_1(\omega_{j-1}, \alpha_{\max})). \tag{10.19}$$

Proof. The proof is by induction on j.

Base Case: Consider $j = \ell$. If we replace ω_j with ω'_j according to Equation 10.19, then the speed is reachable from $\omega_{\ell-1}$ (by the second term in the $\min$ of Equation 10.19). 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 10.19 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 10.19, $\omega'_{\ell+k'+1}$ 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 10.19, $\omega'_{\ell+k} \leq \omega_{\ell+k+1}$.

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. □

CHAPTER 11 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 9 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:

$$\Psi = \{\Omega_n(\omega_{rb_i}, \alpha_{\max}) | (n \in \mathbb{N}_0) \wedge (\omega_{rb_i} \in \mathbf{\Omega_{rb}})\}. \tag{11.1}$$

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,

$$\mathsf{nextPossibleSpeed}(\omega_k) = \begin{cases} \mathbf{\Omega_{rb}} & \text{if } k = 0 \\ \{\Omega_1(\omega, \alpha_{\max})\} \cup \{\omega_{rb_i} \in \mathbf{\Omega_{rb}}(\omega_k)\} & \text{if } k > 0 \end{cases} \tag{11.2}$$

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:

$$\left\{ \begin{array}{l} (\omega_1, \ldots, \omega_{|S|}) \in 2^{\Psi} \mid \\ \quad \left(\forall k \in \mathbb{N}_0^{|S|-1}, \omega_{k+1} \in \mathsf{nextPossibleSpeed}(\omega_k)\right) \\ \quad \wedge \left(\sum_{\ell=1}^{|S|-1} \widetilde{T}(\omega_\ell, \omega_{\ell+1}) + \Omega_1(\omega_{|S|}, \alpha_{\max}) \leq \delta\right) \end{array} \right\} \tag{11.3}$$

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 11.2, $k \geq 0$. According to Lemma 2, the first speed of a dominant sequence must be a right boundary speed, this proves Equation 11.2 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 10.19, k^{th} speed will be replaced by $\min(\Omega_1(\omega_{k-1}, \alpha_{\max}), \omega_{rb_i})$, proving Equation 11.2 for Case 1.

Case 2 (k^{th} speed is the right boundary speed of mode i): Equation 10.19, 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 10.19.

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 8.7.

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 8.7.

□

CHAPTER 12 EVALUATION

In this section, we compare our algorithm against the DRT algorithm [30], 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. [30], 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 [6, 5, 30, 12]. 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 [1].

Table 3: Task set used by existing work [8, 30]

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 4: 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 [8, 30] (Table 3). 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 5(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 5: 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

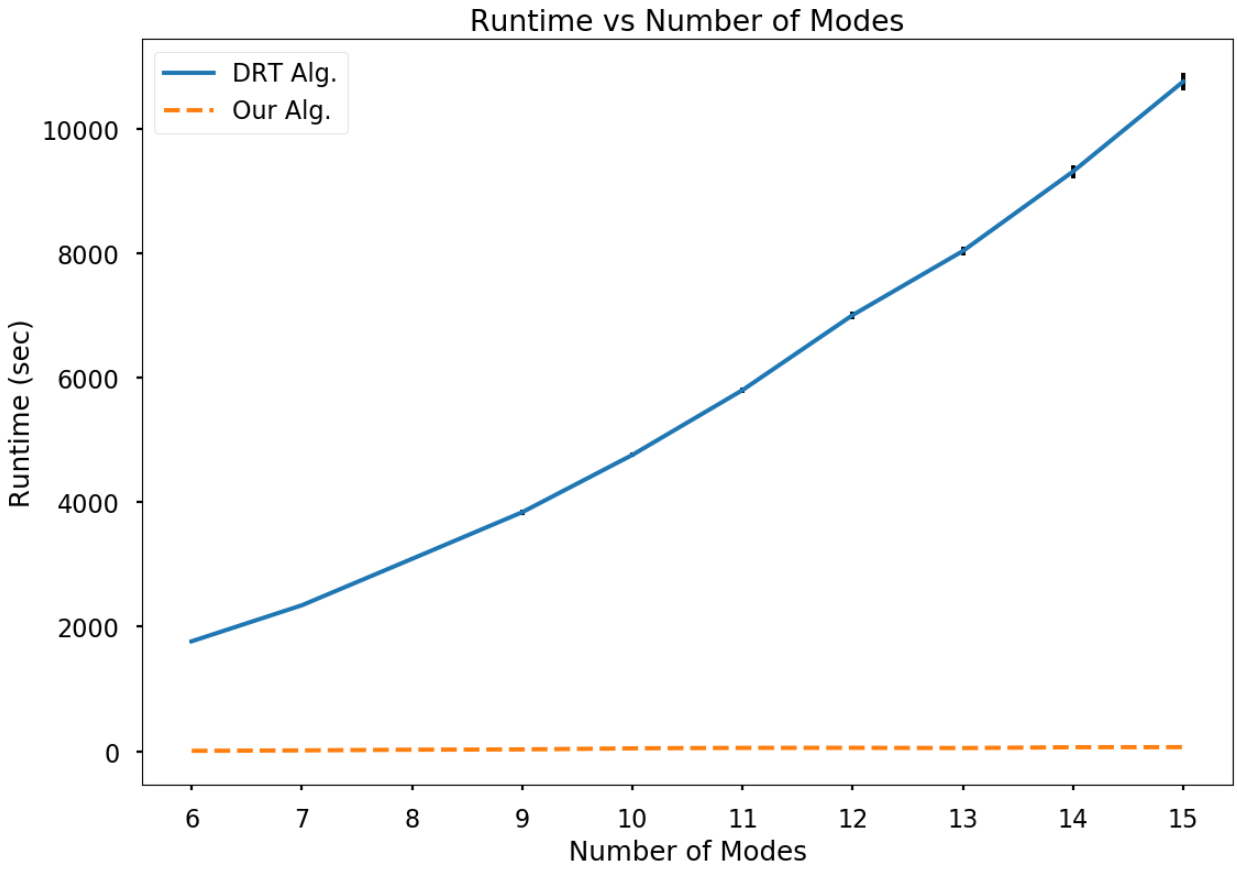


Figure 18: Graph depicting the runtime of different algorithms as a function of the number of modes of randomly generated AVR task sets. For each mode, the 95% confidence interval is shown.

(Table 4). This task set is more general in the sense that the right boundary speeds are not 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 5(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. [6]. In this algorithm, multiple AVR tasks are modeled as a single task, called the representative AVR task, by combining the execution times and the bound-

aries 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. 18.

CHAPTER 13 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.

APPENDIX A

Materials Non-trivial materials used in the experiments can be found in Figure 19. Excluded from the materials list are elements including cooper wire, breadboards, and a power supply. To minimize cost of replication, through-hole parts are used in place of smaller, surface-mount components.

Item	Manufacturer	Part Number(s)
Debugger	Microchip	PG164130
Inductor	Triad Magnetics	RC-1; RC-2; RC-3
Microprocessor	Microchip	DM164130-4
Oscilloscope	RIGOL	DS1102E
Oscilloscope Probe	HANTEK	PP-150
Power Resistor	YAGEO	1334 10W 10R J
Resistor	N/A	510Ω 5% resistor

Figure 19: Materials List
Materials list for Experiments 1-4

APPENDIX B

Inductor Orientation Visualization Given an allowable board space defined as a prism, $P = \{\ell, w, h\}$, there are only three unique orientations for an air-core, solenoid-style inductor in the space. The inductor may be oriented with A located on:

1. the plane formed by l and w extending along h.

2. the plane formed by l and h extending along w.

3. the plane formed by w and h extending along l.

Figure 20 depicts each of these orientations inside a prism with dimensions:

$$P = \{3, 2, 1\}$$

Notice the inductor consuming the most volume has its area A on the plane formed by the two largest dimensions.

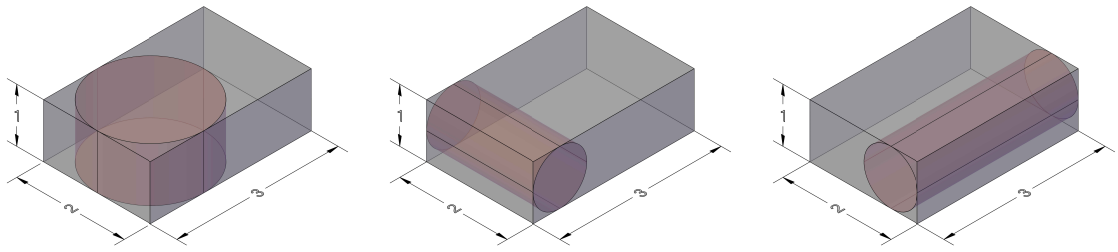


Figure 20: Inductor Orientation Visualization
Possible orientations of air-core, solenoid-style inductors in a constrained space

APPENDIX C

Engine Control Appendix

.1 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:

$$d(S) \geq d(S'). \tag{.1}$$

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 $d(S) - d(S')$:

$$\begin{aligned} \Delta_H(S, S') = \quad & \widetilde{T}(\omega_{\ell-1}, s_n) + \widetilde{T}(s_n, \omega_{\ell+1}) + \tilde{d}(\omega_n) \\ & -\widetilde{T}(\omega_{\ell-1}, \omega_{\ell+1}) - \widetilde{T}(\omega_n, s_n) - \tilde{d}(s_n). \end{aligned} \tag{.2}$$

We now prove that $\Delta_H(S, S') \geq 0$ which proves Equation .1 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:

$$\begin{aligned} & \widetilde{T}(s_{n-2}, s_{n-1}) + \widetilde{T}(s_{n-1}, s_{n-1}) + \tilde{d}(s_{n-1}) \\ & \quad = \widetilde{T}(s_{n-2}, s_{n-1}) + \widetilde{T}(s_{n-1}, s_{n-1}) + \tilde{d}(s_{n-1}) \\ \Rightarrow \quad & \widetilde{T}(s_{n-2}, s_{n-1} + \epsilon) + \widetilde{T}(s_{n-1} + \epsilon, s_{n-1}) + \tilde{d}(s_{n-1}) \\ & \quad \geq \widetilde{T}(s_{n-2}, s_{n-1}) + \widetilde{T}(s_{n-1}, s_{n-1} + \epsilon) + \tilde{d}(s_{n-1} + \epsilon) \end{aligned} \tag{.3}$$

Setting $\epsilon = s_n - s_{n-1}$ and substituting into the above inequality of Equation .3, we get the following:

$$\begin{aligned} & \widetilde{T}(s_{n-2}, s_n) + \widetilde{T}(s_n, s_{n-1}) + \tilde{d}(s_{n-1}) \\ & \quad \geq \widetilde{T}(s_{n-2}, s_{n-1}) + \widetilde{T}(s_{n-1}, s_n) + \tilde{d}(s_n) \\ \Rightarrow \quad & \widetilde{T}(s_{n-2}, s_n) + \widetilde{T}(s_n, s_{n-1}) - \widetilde{T}(s_{n-2}, s_{n-1}) \\ & \quad \geq \widetilde{T}(s_{n-1}, s_n) + \tilde{d}(s_n) - \tilde{d}(s_{n-1}) \end{aligned} \tag{.4}$$

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:

$$\begin{aligned} & \widetilde{T}(\omega_{\ell-1}, s_n) + \widetilde{T}(s_n, \omega_{\ell+1}) + \tilde{d}(\omega_n) \\ & \quad - \widetilde{T}(\omega_{\ell-1}, \omega_{\ell+1}) - \widetilde{T}(\omega_n, s_n) - \tilde{d}(s_n) \geq 0 \end{aligned} \tag{.5}$$

The last inequality (which implies Equation .2 of the property) above follows from observing that according to Property 4, the following is true for all ω and ω':

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

□

.2 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

APPENDIX Z

More Stuff and Things, Again My Appendix Z...

REFERENCES

[1] 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.

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

[3] 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)*, pages 193–198, March 2016.

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

[5] 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*, pages 139–148, July 2015.

[6] 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*, ICCPS '15, pages 120–129, New York, NY, USA, 2015. ACM.

[7] 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*, ICCPS '16, pages 45:1–45:10, Piscataway, NJ, USA, 2016. IEEE Press.

[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*, pages 165–174, July 2014.

[9] C. Busca, R. Teodorescu, F. Blaabjerg, S. Munk-Nielsen, L. Helle, T. Abeyasekera, and P. Rodriguez. An overview of the reliability prediction related aspects of high power {IGBTs} in wind power applications. *Microelectronics Reliability*, 51(9âĂŞ11):1903 – 1907, 2011. Proceedings of the 22th European Symposium on the RELIABILITY OF ELECTRON DEVICES, FAILURE PHYSICS AND ANALYSIS.

[10] G. C. Buttazzo, E. Bini, and D. Buttle. Rate-adaptive tasks: Model, analysis, and design issues. In *2014 Design, Automation Test in Europe Conference Exhibition (DATE)*, pages 1–6, March 2014.

[11] G. Cho and D. X. Shaw. A depth-first dynamic programming algorithm for the tree knapsack problem. *INFORMS Journal on Computing*, 9(4):431–438, 1997.

[12] 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)*, pages 51–62, April 2014.

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

[14] F. Du, W. Chen, Y. Zhuo, and M. Anheuser. A new method of early short circuit detection. *Journal of Power and Energy Engineering JPEE*, 02(04):432âĂŞ427, 2014.

[15] X.-Y. Gao, Y. Cao, Y. Zhou, W. Ding, C. Lei, and J.-A. Chen. Fabrication of solenoid-type inductor with electroplated nife magnetic core. *Journal of Magnetism and Mag-*

netic Materials, 305(1):207 – 211, 2006.

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

[17] S. Hain and M. M. Bakran. New ultra fast short circuit detection method without using the desaturation process of the power semiconductor. In *PCIM Europe 2016; International Exhibition and Conference for Power Electronics, Intelligent Motion, Renewable Energy and Energy Management*, pages 1–8, May 2016.

[18] P. M. Hettiarachchi, N. Fisher, M. Ahmed, L. Y. Wang, S. Wang, and W. Shi. A design and analysis framework for thermal-resilient hard real-time systems. *ACM Trans. Embed. Comput. Syst.*, 13(5s):146:1–146:25, July 2014.

[19] S. Hollister. Here's why samsung note 7 phones are catching fire. `https://www.cnet.com/news/why-is-samsung-galaxy-note-7-exploding-overheating/`, Oct 2016. CNET.

[20] T. Horiguchi, S. i. Kinouchi, Y. Nakayama, T. Oi, H. Urushibata, S. Okamoto, S. Tominaga, and H. Akagi. A short circuit protection method based on a gate charge characteristic. In *2014 International Power Electronics Conference (IPEC-Hiroshima 2014 - ECCE ASIA)*, pages 2290–2296, May 2014.

[21] T. Horiguchi, S. i. Kinouchi, Y. Nakayama, T. Oi, H. Urushibata, S. Okamoto, S. Tominaga, and H. Akagi. A high-speed protection circuit for igbts subjected to hard-switching faults. *IEEE Transactions on Industry Applications*, 51(2):1774–1781, March 2015.

[22] W. H. Huang and J. J. Chen. Techniques for schedulability analysis in mode change systems under fixed-priority scheduling. In *2015 IEEE 21st International Conference*

on Embedded and Real-Time Computing Systems and Applications, pages 176–186, Aug 2015.

[23] B. Ji, V. Pickert, W. Cao, and B. Zahawi. In situ diagnostics and prognostics of wire bonding faults in igbt modules for electric vehicle drives. *IEEE Transactions on Power Electronics*, 28(12):5568–5577, Dec 2013.

[24] D. S. Johnson and K. A. Niemi. On knapsacks, partitions, and a new dynamic programming technique for trees. *Mathematics of Operations Research*, 8(1):1–14, 1983.

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

[26] 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*, ICCPS '12, pages 55–64, Washington, DC, USA, 2012. IEEE Computer Society.

[27] T. Krone, C. Xu, and A. Mertens. Fast and easily implementable detection circuits for short-circuits of power semiconductors. In *2015 IEEE Energy Conversion Congress and Exposition (ECCE)*, pages 2715–2722, Sept 2015.

[28] C. L. Liu and J. W. Layland. Scheduling algorithms for multiprogramming in a hard-real-time environment. *J. ACM*, 20(1):46–61, Jan. 1973.

[29] C. L. Liu and J. W. Layland. Scheduling algorithms for multiprogramming in a hard-real-time environment. *J. ACM*, 20(1):46–61, Jan. 1973.

[30] M. Mohaqeqi, J. Abdullah, P. Ekberg, and W. Yi. Refinement of Workload Models for Engine Controllers by State Space Partitioning. In M. Bertogna, editor, *29th Euromicro Conference on Real-Time Systems (ECRTS 2017)*, volume 76 of *Leibniz In-*

ternational Proceedings in Informatics (LIPIcs), pages 11:1–11:22, Dagstuhl, Germany, 2017. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik.

[31] A. K. Mok. *Fundamental Design Problems Of Distributed Systems For The Hard-Real-Time Environment*. PhD thesis, Massachusetts Institute of Technology, Cambridge, MA, USA, 1983.

[32] R. Pagano, Y. Chen, K. Smedley, S. Musumeci, and A. Raciti. Short circuit analysis and protection of power module igbts. In *Twentieth Annual IEEE Applied Power Electronics Conference and Exposition, 2005. APEC 2005.*, volume 2, pages 777–783 Vol. 2, March 2005.

[33] 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*, pages 71–80, April 2011.

[34] W. S. University. Wayne state university library system. `https://library.wayne.edu/`, Mar 2019. WSU.

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

[36] Y. Wang, X. Zhai, Z. Song, and Y. Geng. A new method to detect the short circuit current in dc supply system based on the flexible rogowski coil. In *Electric Power Equipment - Switching Technology (ICEPE-ST), 2011 1st International Conference on*, pages 237–240, Oct 2011.

[37] A. Willcock. Short circuit detection utilization analysis under uniprocessor edf scheduling. Senior Thesis, 2016.

[38] A. Willcock and N. Fisher. Trading utilization for circuitry: Hardware-software co-design for real-time software-based short-circuit protection. In *2017 IEEE 23rd International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA)*, pages 1–8, Aug 2017.

[39] H. D. Young, R. A. Freedman, A. L. Ford, and H. D. Young. *Sears and Zemansky's University physics*. Pearson Learning Solutions, 2012.

[40] L. Zhang and A. Q. Huang. Model-based fault detection of hybrid fuel cell and photovoltaic direct current power sources. *Journal of Power Sources*, 196(11):5197 – 5204, 2011.

ABSTRACT

TITLE LINE 1
TITLE LINE 2 (if needed)
TITLE LINE 3 (if needed)
HOW LONG IS THIS TITLE GOING TO BE?

by

FULL NAME

(MONTH YOU WILL GRADUATE) 20XX

Advisor: DR. FISHER

Major: COMPUTER SCIENCE

Degree: DOCTOR OF PHILOSOPHY

In this work, we design a new flux capacitor that allows time travel at 87 mph, a 1 mph improvement over the state of the art.

AUTOBIOGRAPHICAL STATEMENT

Your bio goes here...