Literature Review:
AVR Task Scheduling
Rashawn Peters

Introduction
This paper is a literature review of a handful of research papers on the topic of AVR task scheduling. The research papers being examined are endeavouring to evaluate the schedulability of AVR tasks in different scenarios and with different stipulations. The papers are sourcing information from well renowned experts in the Real-Time Systems field. Each of the papers conducts its own analysis and facilitates a discussion on its own. This paper is evaluating each of those research articles and discussing their findings as a whole.

The sources chosen for this literature review are all currently available and up-to-date sources of Real-Time Systems knowledge. They all come from credible journals published between 2015 and 2018.

Adaptive-Variable Rate Tasks
The main task featured in all of these papers is the Adaptive-Variable Rate (or AVR) task. This is a set of tasks whose execution time is dependent on a variable period. This is formally a difficult thing to consider because the variable period causes issues in the standard schedulability tests. The known theorems and proofs for schedulability on different types of algorithms all tend to be made to work on static period task sets. Having the period being able to shift throughout the execution makes it more difficult to determine different forms of schedulability. In the case of these papers examined, the example of AVR tasks that were examined are engine-triggered tasks. They are tasks that release when the crankshaft of an engine makes a full rotation.

Knapsack-Based Approach
The paper dealing with this topic is titled “An Efficient Knapsack-Based Approach for Calculating the Worst-Case Demand of AVR Tasks.” The standard knapsack problem is a scenario dealing with organizing different combinations of products with the goal of betting the best optimized result. You are given a set of example items that each have their own weight and value to them. The solution to the problem is to find the number of each item to add to the knapsack in order to get as close as possible to the knapsack limit without giving over and overflowing the bag. This is a problem that has been around for over a century and taken different forms, but the basis of the problem is generally the same.

This paper brings out a great point about the need for fine-tuned balance when it comes to scheduling algorithms in real use. It mentions, “Pessimistic assumptions lead to overestimation of workload, which results in underutilization of the resources. On the other hand, workload underestimation can cause deadline misses.” (Sandeep et al. 1) In either sense, there is an issue with the schedule that should be corrected. Overestimating the workload will be a waste of the systems resources and lead to some things being unused. The simple thought process to counteract this would be to just have a lower estimate of the workload, so there won’t be a ton of excess resources. However, this leads to the potential problem of deadline misses.

In this paper, it uses the definition of AVR tasks to be “a set of modes, each of which is expressed by a range of speeds and a constant execution time.” These tasks are used to model the behavior of engine-trigger tasks, like mentioned before. Using standard models of tasks systems with constant periods will not work for engine-trigger tasks because their periods are variable and it would lead to an overestimation of workload resources. This applies to vehicles with powertrain control modules, which result in higher speeds and more jobs being released, causing deadlines very quickly if they are not handled properly. To
schedule an AVR task on an Earliest Deadline First algorithm, the paper mentions that the demand bound function is usually used because it allows for the resource requirement to be measured over a given amount of time.

The paper had three main goals that it set out to accomplish when it comes to these AVR tasks. The first, determine the worst-case demand of an AVR task. It provides a dynamic programming solution to this knapsack problem that solves it efficiently. Secondly, it determines that the number of job sequences needed for calculating the worst-case demand of an AVR task is much less if the kinematic properties of the engine are used. Also, it states that the approach detailed in the paper outperforms the known methods for dealing with this problem previously.

The task model for this paper includes the AVR tasks triggered by certain angles of the crankshaft. However, getting to this angle varies depending on the angular speed of the crankshaft. As the speed increases, more and more jobs of each task occur and that increases the amount of resources needed. This leads to the use of more pessimistic cases. To account for this increase in speed and variability, AVR task execution times are modeled as a function of the speed at which the jobs are released. The findings of this paper can be extended to multiple AVR tasks because it was proven before that multiple AVR tasks with the same variable factors can be represented by one AVR task. This increases the usability of the analysis shown in the paper.

After defining the variables needed for the analysis, the exact problem was defined. It was considering a minimum angular deadline AVR task that is characterised by feasible speed and acceleration ranges. The maximum acceleration is equal to the absolute value of the minimal acceleration. There are a set of modes that are associated with an execution time with a variable ranging from 1 to m. The main objective was to find the worst case demand of the AVR task over any time interval. All of this was also taking into consideration that the acceleration of the crankshaft may change during a single rotation.

In this equivalent of the knapsack problem, the problem is finding $dbf(\delta)$ when the job is the item and the weight is the minimum inter-arrival time, and the profit of the knapsack problem is the execution time. The end result is a maximized demand over $\delta$. This is a bounded knapsack problem due to the fact that a job’s execution time may contribute to the $dbf(\delta)$ more than once. There are precedence constraints put upon the jobs and that was modeled using the figure below from the paper.

![Diagram](image)

This model is an “out-tree” in which the nodes represent the WCETs and arrows are for the minimum inter-arrival times.

The set of speed sequences that must contain the dominant speed sequence was defined as the dominant sequence set in this paper and the proofs and theorems used to back up the definition were detailed throughout the middle section.

In final evaluation of the algorithm used in the study, it was compared to the DRT algorithm that was the current standard for this type of analysis. With comparison, it was shown that it was very similar to the DRT algorithm when it came to the speed benchmarks. However, the real upper hand that the paper’s algorithm had was that it got rid of several unnecessary traversals through the tested speeds. This resulted in a big decrease in computational complexity. Looking at the detailed lengths went through to make the testing fair and equal, it is safe to assume that the results are accurate. The two algorithms were compared on the same task sets and done multiple times with an average taken. On this non-ideal task set, it was shown that the algorithm described in the paper
was 13.5 times faster than the DRT algorithm. In a more general task set that they created, they were able to achieve a worst-case demand 53.8 times faster than the DRT. Furthermore, in a second experiment, they used an algorithm to generate multiple AVR tasks modeled as one by combining the execution times and boundary speeds. With this test, they were able to see that their algorithm was up to 250 times faster. It truly is a great breakthrough compared to the DRT algorithm.

Uniprocessor EDF Scheduling

The paper dealing with this topic was titled “Uniprocessor EDF Scheduling for AVR Task Systems.” It endeavoured to help define the role of cyber-physical systems in the real-time scheduling theory. Complex model-based designs are often used for cyber-physical systems where a model is specified, then one demonstrates the correctness of the model, and eventually implements that model down the line. That last step of implementation is where real-time systems scheduling theory comes into play. Elements of the abstract model are mapped to recurrent task models like periodic tasks or sporadic tasks used in real-time scheduling theories. These tasks are evaluated like real-time schedules and applied. This is not the best case all of the time because it tends to introduce a lot of additional pessimism into the system because of the gaps left.

A solution to this would be to develop more low level real-time system task models. Making them inspired by real physical systems could make them more useful when in use for bigger applications. This paper deals with “the modeling of recurrent processes in cyber-physical systems for which each activation of the recurrent process is triggered by the values of variables describing the state of the physical system.” (Guo and Baruah 1) The example explored here is the same one used in other papers before, the Engine Control Unit. This unit on a vehicle depends on the position of each engine piston, leading back to the crankshaft angular position.

The engine control unit executes are varied rates depending on the engine speed. Engine behavior is typically more stable at higher speeds, and therefore some lower speed functions don't need to be executed at higher speeds too. The challenge here is described as being dependent on engine rotation speed that varies between maximum acceleration and deceleration. Some early work shown of the AVR problem used a simplified model with one single task in order to analyze and come up with conclusions about the worst case execution times. There was not much other work before this paper studying uniprocessor dynamic-priority scheduling with AVR tasks in the task set for a system.

Throughout the paper, it goes into detail on the AVR tasks used in models and how it affects them. It also provides an efficient sufficient schedulability test for implicit mixed AVR task sets. Furthermore, it moves into considering AVR task systems with constrained deadlines, as they are different modelling than the implicit jobs. All of this is to compare with the current standards at the time of AVR scheduling.

For the system model, earliest deadline first scheduling on a set of tasks with a preemptive uniprocessor. AVR sets were obviously included on that as well. With the example of the car engines, the AVR tasks are the crankshaft rotations for a cumulative angle of $4\pi$, but the results should be unchanged with different activation angles used. The measurement used throughout the paper for the engine rotation is the pretty standard rpm. For the purposes of the paper, the formula was given as 1 rpm = 0.10472 rad/sec, which is the angular speed measurement used often in this field.

The WCET of a job was determined from the angular speed of the system at its release, but the period may vary due to dynamic changes in the system. Due to this variety, the paper defined the period of the AVR task as the equation below:

$$T_i(\omega) = \begin{cases} \sqrt{\frac{\omega(t)^2 + 8\pi\omega}{\omega(t) - \omega}} & \text{if } \omega(t) \leq \Omega(4\pi), \\ \frac{8\pi(\omega(t) - \omega(t))^2}{2\omega_{\max}} & \text{if } \omega(t) > \Omega(4\pi); \end{cases}$$

It is shown that the activation period is being determined by the minimum possible separation and therefore the worst case angular speed ($w$) changes during runtime when estimating the task period.
The consideration of implicit systems included the discussion of a utilization-based test and the speedup factor. Implicit tasks were defined as tasks with relative deadlines equal to the periods for both regular and AVR tasks. Their relative deadlines are the same as the activation periods. The utilization test shown in this paper for implicit systems is the common test. Since this is an EDF schedulability test, the total utilization needs to be less than or equal to one. Just like other utilizations, for AVR tasks, their utilization is equal to their execution time divided by their period.

The speedup factor helped to qualify how far the current schedulability test was from being optimal. This was the first analysis of its kind shown in the report. The speedup factor is used when comparing how a task set is scheduled on a unit-speed processor to how it performs on a processor that is “s” units faster. To create generalization for the testing done in this section of the report, AVR tasks were restricted to have the same mode and angular speed when their utilizations are maximized. The schedulability test discussed earlier with having the total utilization be a maximum of 1 has a speedup factor of at most \( 1/1 - \beta + \beta/\eta(\omega) \) according to the theorem stated in the paper. The proof for this theorem involved comparing the total utilization of two task sets divided by each other and showing that they were more than the speedup factor. That in turn showed that the speedup factor for that was the equation stated.

\[
\forall i, j, \alpha^* \leq \Delta j \omega i/(2\omega j i + \Delta j \omega i)/(16\pi)
\]

was upheld as a condition. In proving this, it was important to mention that this schedulability test was very similar to the previous work mentioned before. The main difference was in how the utilization of AVR tasks was calculated.

Moving away from implicit scenarios, now the paper considered constrained deadlines for tasks. These are deadlines that arise quite often in engine triggered tasks so it makes sense to consider them. Initially, there was a simple example used to show how the digraph model can be used to show the schedulability intuitively and be used for analysis. This can be used after to describe the transformation further. A digraph is sometimes called a directed graph and it is made up of sets of vertices and edges connected together with directions associated to each edge.

When choosing the number of vertices to be used in the digraph, the paper mentioned the different things that needed to be considered. Each one represented a pessimistically chosen range of angular speeds. The range still needed to be small enough and that could lead to infinite vertices. To increase the precision, it was shown that more vertices could be added to the system.

For the analysis of this paper, their proposed method was compared with existing tests for schedulability and evaluated to see their effectiveness. Their method was compared against RTA-SP and Exact-CON. The former uses the maximum worst-case execution time, minimum period, and minimum deadline for all the modes to model each AVR task as a
traditional sporadic task. The latter is done by computing the maximum interference with added constraints from the physical system instead of Integer Linear Programming. Exact-CON is currently the best method from analyzing AVR task systems under fixed priority scheduling.

The sets of tasks were generated using the same four steps in order to ensure fair comparisons and analysis throughout the study. In the testing of implicit systems, it was shown that the method detailed in the paper outperformed the existing ones. This case was exasperated when the systems were heavily loaded with tasks.

The schedulability tests listed before were compared with respect to changes in one of the parameters to see how the system would change. This is done by weighted schedulability. In a trio of figures in the report, the weighted schedulability was shown when the size of the task set increased. The weighted schedulability was also shown when the adjusting factor was increased. As the adjusting factor increased, the system became less affected by other modes. Since the proposed analysis from the paper was precise to one node, the schedulability ratio was increasing. The third figure showed how the proportion of AVR tasks to other periodic or sporadic tasks affected the weighted schedulability. There was a minor decrease of weighted schedulability when more AVR tasks are present in a set. This is due to the fact that there are not yet schedulability analysis methods for AVR tasks in this paper that are as accurate as the ones for sporadic tasks.

The conclusions drawn from this paper helped add to the scarce landscape of AVR analysis at the time. Most previous work on this subject was only on fixed-priority scheduling algorithms, not on an earlier deadline first like this paper considered. It was able to develop an efficient schedulability analysis for the AVR task model in the physical constraints of an automobile engine (a cyber-physical system). It was able to show that fixed priority scheduling is not always something that would work for AVR task sets. As the deadline for AVR tasks change, its priority should change as well. If not, then the system will suffer from priority inversion eventually. Priority inversion in scheduling is when a high priority task is indirectly blocked and preempted by a lower priority task, which inverts the relative priority of the two tasks. This will result in an under-utilization of resources with an AVR task set. In order to avoid this, the earliest deadline first scheduling was the natural solution that the paper decided to explore. It was able to show an efficient and sufficient schedulability test for implicit systems through the report.

Deriving the speedup factor as a function of engine rotation speed helped to add to the discussion of AVR task analysis. It was proved to be necessary and sufficient by the analysis in the paper. For systems with deadlines that were constrained in respect to the periods, demand based function analysis was done by transforming the function into a digraph based task model. All of the methods discussed in the two papers show there was a better way of schedulability that can outperform the current state of the art methods of dealing with AVR task sets.

Bibliography
