Omega ∎ (∎∎∎∎) ∎∎∎–∎∎∎

Contents lists available at ScienceDirect

Omega

journal homepage: www.elsevier.com/locate/omega

An analytical approach to prototype vehicle test scheduling ☆

Yuhui Shi [a,*], Daniel Reich [b], Marina Epelman [a], Erica Klampfl [b], Amy Cohn [a]

[a] Industrial & Operations Engineering, University of Michigan, Ann Arbor, MI 48109, United States
[b] Research & Advanced Engineering, Ford Motor Company, Dearborn, MI 48120, United States

ARTICLE INFO

Article history:
Received 26 November 2014
Accepted 10 May 2016

Keywords:
Automobile industry
Scheduling
Integer programming
Heuristics

ABSTRACT

The test planning group within Ford's Product Development division develops schedules for building prototype vehicles and assigning them to departments in charge of different vehicle components, systems and aspects (e.g., powertrain, electrical, safety). These departments conduct tests at pre-production phases of each vehicle program (e.g., 2015 Ford Fusion, 2016 Ford Escape) to ensure the vehicles meet all requirements by the time they reach the production phase. Each prototype can cost in excess of $200 K because many of the parts and the prototypes themselves are hand-made and highly customized. Parts needed often require months of lead time, which constrains when vehicle builds can start. That, combined with inflexible deadlines for completing tests on those prototypes introduces significant time pressure, an unavoidable and challenging reality. One way to alleviate time pressure is to build more prototype vehicles; however, this would greatly increase the cost of each program. A more efficient way is to develop test plans with tight schedules that combine multiple tests on vehicles to fully utilize all available time. There are many challenges that need to be overcome in implementing this approach, including complex compatibility relationships between the tests and destructive nature of, e.g., crash tests. We introduce analytical approaches for obtaining efficient schedules to replace the tedious manual scheduling process engineers undertake for each program. Our models and algorithms save test planners' and engineers' time, increases their ability to quickly react to program changes, and save resources by ensuring maximal vehicle utilization.

1. Introduction

Ford's Product Development division is responsible for designing and testing new vehicles and readying them for production. Each vehicle program (e.g., 2015 Ford Fusion, 2016 Ford Escape) progresses through several consecutive stages: concept, design, development and testing, etc., before a new vehicle is manufactured on the assembly line. After the concept and design phases are completed, prototype vehicles are built and subjected to tests to ensure the new vehicle model meets all the design criteria. Each required test needs to be completed by its deadline to ensure adherence to the overall program timing. Test planners and engineers are tasked with scheduling all the tests, placing orders for parts to build the required prototype vehicles, scheduling the order of the builds (e.g., prototype vehicle with automatic transmission on day 1, one with manual transmission on day 2) and assigning the vehicles to departments in charge of tests for different vehicle components, systems, and aspects (e.g., powertrain, electrical, safety).

Each prototype built during the development and testing phases of a vehicle program can cost in excess of $200 K because many of the parts and the prototypes themselves are hand-made and highly customized. Parts needed often require months of lead time, which constrains when prototype vehicle builds can start. That, combined with inflexible deadlines for completion of tests on those vehicles, introduces significant time pressure, an unavoidable and challenging reality associated with maintaining the overall program timing. One way to alleviate time pressure is to build more vehicles, essentially decreasing competition between tests for available vehicle time; however, this would greatly increase the cost of each program. A more efficient way is to develop test plans with tight schedules that combine multiple tests on vehicles to maximally utilize available time. There are many challenges that need to be overcome in implementing this approach. For example, many tests are destructive (e.g., crash tests performed by the safety department), preventing scheduling further tests on the vehicle. Another complicating factor is that different tests may have different vehicle specification requirements; for example, one test may require a hybrid engine whereas

☆This manuscript was processed by Associate Editor W. Shen.

* Corresponding author.
E-mail addresses: yuhuishi@umich.edu (Y. Shi), dreich8@ford.com (D. Reich), mepelman@umich.edu (M. Epelman), eklampfl@ford.com (E. Klampfl), amycohn@umich.edu (A. Cohn).

http://dx.doi.org/10.1016/j.omega.2016.05.003
0305-0483/

another may require a conventional 4-cylinder I4 engine, prohibiting combinations of these tests on the same vehicle.

Prior to our work, test plans were exclusively developed manually using pen and paper and Excel spreadsheets. However, this process is tedious and constructing a test plan may take days, if not weeks. The schedule achieved may not be optimal in terms of the number of vehicles needed; moreover, when changes occur to deadlines and individual tests, manually editing the plan requires significant additional time and effort, and may lead to decreasing vehicle utilization. In this paper, we formally define the problem of obtaining optimized schedules (i.e., ones that minimize the number of vehicles used subject to all pertinent constraints) and introduce computational heuristics that replace the tedious manual scheduling process engineers undertake for each program. Automation saves test planners' and engineers' time, increases their ability to quickly react to program changes, and saves resources by providing schedules with high vehicle utilization.

In this paper, we describe the development and piloting of our schedule optimization models. We introduce the details of the scheduling problem, then provide an exact mathematical formulation, which turns out to have limited tractability. This leads to our practical heuristic algorithm that provides good feasible schedules. We present results from our first pilot and discuss ongoing efforts and goals of this project.

2. Literature review

Although certain aspects of the optimization problem addressed in this paper are specifically motivated by prototype vehicle test scheduling at Ford, it has some features of bin packing on the one hand, and parallel machine scheduling on the other, and can be viewed as an extension of both problems.

In the classic bin packing problem, a set of items with different sizes needs to be packed into bins of limited capacities, and the minimum number of bins required is to be determined. There are extensive studies of this problem (see, e.g., [1]). In our setting, determining the minimum number of vehicles needed to perform all tests is akin to bin packing with non-identical bins (vehicles), whose capacity reflects the time interval during which the vehicle is available, and with additional restrictions on the compatibility of items (tests) to be assigned to the same bin. A paper in this area most closely related to our research is [2]. In it, the authors consider a variation of the bin packing problem with conflicts between items. The authors provide a set-partitioning formulation of the problem and propose a branch-and-price algorithm to solve it exactly, with the pricing problem solved as a knapsack problem with conflicts. Our problem, however, is more complex, since tests assigned to the same vehicle need to be scheduled as well.

In the parallel machine scheduling problem, a set of time-sensitive tasks with associated processing times need to be scheduled on a given set of machines, while minimizing a certain criterion, usually time-related, such as make-span or total tardiness. The literature on parallel machine scheduling has developed over several decades and contains a variety of models and algorithms; comprehensive surveys and comparisons between different solution strategies can be found in [3–5]. Associating machines with vehicles and jobs with tests, one can see many similarities between test scheduling and certain types of machine scheduling problems. Indeed, in machine scheduling jobs often have release and due dates, and test compatibility and sequencing restrictions can be represented by including setup times between jobs, setting them to very high values for tests that cannot be performed together or in a particular order. However, our test scheduling problem has several features that make it unique in the scheduling literature. In particular, machines are usually assumed to be available throughout the scheduling process, whereas prototype vehicles are released gradually during testing. Moreover, while specification of which machines are capable of executing which jobs is considered in the literature, whether a prototype vehicle has the features needed for a particular test is determined by the *other* tests assigned to this vehicle (see Section 3 for details), making a priori specification impossible. Finally, the objective of minimizing the number of vehicles used is fairly uncommon in the scheduling literature. In light of the above, in our review of machine scheduling literature we will focus on the papers that aim to minimize the number of machines used. We also discuss representative papers which emphasize sequencing aspects of scheduling in the presence of precedence constraints or setup times, especially those that utilize heuristic algorithms similar to the Fit-and-Swap heuristic we propose in Section 5, to emphasize relevant results as well as elucidate the distinct features of our problem.

A small subset of machine scheduling literature focuses on a problem most closely related to ours, where the goal is to optimize some machine-related metrics, such as the cost of holding and using machines, rather than the traditional job-related time metrics. In addition to bin-packing resources, in the context of scheduling, a particularly relevant paper [6] studies the so called "Scheduling with Release times and Deadlines on a minimum number of Machines (SRDM)" problem, where each task has a duration and a time window for execution, and the number of machines required to perform all tasks on time remains a decision. The authors propose a polynomial algorithm for the special case when time windows of jobs are tight (namely, exactly 1 plus job duration). In the more general case, they propose an approximation method called Greedy-Best-Fit, which is a list scheduling algorithm that assigns jobs to the machine with the left-most available time slot. They prove that this heuristic is a 9-approximation algorithm in the special case of equal processing times. Reference [7] improves the approximation bound from 9 to 6 for the same special case. For another special case of common release dates, the authors of the latter paper propose another list scheduling algorithm, called Greedy-Increasing-Slack, which sorts and assigns jobs in reverse order of flexibility of shifting within its time window. This method has a constant approximation bound 2.

For a related vehicle routing problem, Reference [8] studies minimizing the number of trucks to satisfy customer loads, where each load has a time window during which it should be delivered. The authors propose a two-phase approach which uses a time-indexed formulation to form sequences of loads and later heuristically assigns them to trucks. Such an approach can solve up to instances with 34 loads and 10 trucks—smaller than the test scheduling instances for which we provide computational results.

Several other papers consider machine scheduling while minimizing the number of machines used, including [9–11]. However, they each make restrictive assumptions to guarantee that their proposed algorithms solve the problem exactly or with a guaranteed bound. For example, equal processing times and/or common release or due dates of jobs are often assumed in such papers. Interestingly, in [11] the authors also consider precedence constraints among different jobs across machines, where the start of job A cannot precede the completion of another job B (they do assume equal processing times). They propose polynomial algorithms that can solve this problem for special cases of precedence graph structures, such as trees and chains. Although precedence relationships may seem similar to a feature of the test scheduling problem, in the latter case, the precedence relations between two test are only relevant if they are assigned to the same vehicle.

In the more traditional context of scheduling (one where the number of available machines is specified a priori), one class of problems that are particularly relevant to ours is the setting with

Please cite this article as: Shi Y, et al. An analytical approach to prototype vehicle test scheduling. Omega (2016), http://dx.doi.org/10.1016/j.omega.2016.05.003