Computers and Operations Research 100 (2018) 128–139

Contents lists available at ScienceDirect

Computers and Operations Research

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

Advanced constraint propagation for the combined car sequencing and level scheduling problem

Mesut Yavuz[a,*], Hüseyin Ergin[b]

[a] The University of Alabama, Department of Information Systems, Statistics, and Management Science, United States
[b] Ball State University, Department of Computer Science, United States

ARTICLE INFO

Article history:
Received 20 September 2017
Revised 11 June 2018
Accepted 19 July 2018
Available online 27 July 2018

Keywords:
Automotive industry
Branch-and-bound
Just-in-Time
Manufacturing
Optimization
Sequencing

ABSTRACT

We present an advanced constraint propagation algorithm for the combined car sequencing and level scheduling problem, used within a branch-and-bound framework. The new method solves the only unsolved instance from Drexl et al. (2006) and four of the eight unsolved instances from Yavuz's (2013) testbeds. The paper also introduces 18 new instances, 9 of which are solved by the algorithm. Optimal sequences of up to 250 cars are obtained.

1. Introduction

Mixed-model automotive assembly line sequencing is addressed in two main streams of research: *car sequencing* and *level scheduling*. Car sequencing is a pure constraint satisfaction approach. Car variants are expressed as combinations of some key product characteristics (called "*options*") that affect the processing times of variants in critical workstations. Without loss of generality, an option is assumed to increase the processing time of a variant requiring it. Car sequencing aims to find a feasible sequence of cars so that total requirement of each option in every subsequence is limited, where the maximum allowed requirement and subsequence length vary among options.

Level scheduling is also a discrete-time sequencing approach aiming to minimize aggregate deviations from an ideal (smooth) schedule. Level scheduling is widely used in just-in-time manufacturing systems. Only a few papers have addressed the Combined Car Sequencing and Level Scheduling Problem (CCSLSP). Drexl and Kimms (2001) wrote the seminal paper on the CCSLSP and proposed a column generation algorithm to solve it. Drexl et al. (2006) and then Yavuz (2013) incorporated constraint propagation into tree search algorithms of branch-and-bound and iterated beam search, respectively. The present paper solves the CCSLSP more efficiently by advancing constraint propagation and utilizing it within a branch-and-bound algorithm.

The remainder of the paper is organized as follows. Section 2 briefly reviews the related work in car sequencing and level scheduling literatures. Section 3 mathematically defines the problem. In Section 4, we first build a hierarchical structure of auxiliary variables that complements the mathematical formulation of the previous section and exploits the underlying mathematical structure of the problem. We then establish 38 rules regulating the relationships among those auxiliary variables, and embed them in our constraint propagation algorithm. Section 5 demonstrates the effectiveness of the constraint propagation algorithm when used within a branch-and-bound algorithm on existing and new, larger-size instances of the problem. Finally, Section 6 provides some concluding remarks and points at possible future research directions.

2. Related work

2.1. Car sequencing

The car sequencing problem (CSP) is a discrete-time scheduling problem consisting of assignment and option constraints, and it aims to find a feasible sequence of a given set of car variants.

Let V denote the number of distinct car variants sharing a mixed-model assembly line. Demand (d_v) for the planning horizon, which is a day or a shift, is given for each variant $v = 1, ., V$.

* Corresponding author.
E-mail address: myavuz@cba.ua.edu (M. Yavuz).

https://doi.org/10.1016/j.cor.2018.07.018
0305-0548/

The planning horizon consists of T equal-length slots, where $T = \sum_{v=1}^{V} d_v$ is the total demand for the planning horizon and each slot's length equals the cycle time of the assembly line. The slots, hereafter referred to as positions or periods, of this discrete sequence are indexed by t.

If the assembly line is perfectly synchronized, i.e., each variant takes the same amount of processing time in each workstation, then any sequence of T cars could flow through the assembly line without causing any line stoppages. However, when assembly line balancing does not result in such a perfect synchronization, some options may require longer (than cycle time) time in some workstations. Line stoppages may be preventable as long as the total time requirement in a workstation throughout the planning horizon does not exceed the length of the planning horizon, i.e., average processing time in a workstation does not exceed the cycle time. Stations can be made longer and also cars with longer and shorter processing times can be alternated in a sequence to allow the workstation to catch up with the workload. In the automotive industry, processing times are typically discrete such that a workstation may have a long processing time for each car requiring a certain option and a short (possibly zero) processing time for each car not requiring that option. Line stoppages can then be prevented by spacing cars requiring a certain option, giving rise to the option constraints in the CSP.

There are O distinct options, indexed by o. Variant v requires option o if $b_{v,o} = 1$, and does not require it if $b_{v,o} = 0$. The total requirement of option o is $R_o = \sum_{v=1}^{V} b_{v,o} d_v$. An option constraint is denoted by H_o: N_o, meaning that at most H_o cars requiring o are allowed to appear in any subsequence of N_o cars. Option constraints are also called "spacing" constraints due to their role in spacing subsequent occurrences of options. The subsequence of N_o cars ending with position t is called the constraint window for option o ending with position t, and is denoted by $w_{o,t} = \{\max\{1, t + 1 - N_o\}, ., t\}$. We also define $\Phi_{o,\,t}$ as the total requirement for option o in $w_{o,\,t}$ for future use.

Bolat and Yano (1992) provide the basic analytical approach to determine H_o and N_o in presence of binary processing times such that variants requiring an option take longer than cycle time and others take shorter than cycle time. Golle et al. (2010) analyze the differences between feasibility of sequences with respect to option spacing constraints (H_o: N_o) and the actual workloads in individual workstations along the line. The authors develop a new approach to determining H_o and N_o also applicable in presence of multiple processing times. We refer the interested reader to these works and to Lesert et al. (2011) for further reading on selecting critical options and quantifying their spacing constraints, and focus on sequencing cars with given option constraints.

The CSP is concerned with finding a feasible sequence of T cars satisfying the stated assignment and option constraints, and it is strongly $\mathcal{NP}$-hard (Kis, 2004). Assignment constraints assure that each car is assigned to exactly one position, and exactly one car is assigned to each position. Clearly, the assignment constraints are easily satisfied and the difficulty of the CSP lies in the option constraints. The problem is $\mathcal{NP}$-hard even when all options constraints are 1:2, i.e., $H_o = 1$ and $N_o = 2$ for all options (Estellon and Gardi, 2013). The CSP, first proposed by Parrello et al. (1986), has received significant interest from the constraint programming community (Dincbas et al., 1988; van Hentenryck et al., 1992; Tsang, 1993). A test-bed for the CSP is included in CSPLib: a library for constraints (Gent and Walsh, 1999).

Exact solution approaches for the CSP are limited due to its computational complexity. Kis (2004) developed a dynamic programming formulation that can practically solve problems up to 20 cars. Given the problem-size limitations of exact solution approaches, heuristics have been researched in more depth. Siala et al. (2015) build a unified heuristic algorithm that can be characterized based on four components. The authors state that 42 meaningful combinations exist and some of them correspond to existing constraint programming heuristics in the literature. Through a computational experiment, they analyze the impact of different selections available for the four components and find that branching on variants is superior to branching on options . Another finding from this study is that starting from the middle of the planning horizon and filling positions towards the ends in an alternating fashion works slightly better than starting from the first position and filling sequentially therefrom.

The operations research community has also shown some interest in the CSP, especially around 2005, when an extended version of the CSP was the topic of the French Society of Operations Research and Decision Aid's bi-annual challenge; see Solnon et al. (2008) and the references therein for papers developed for the stated challenge. Operations research approaches to the CSP mostly rely on modifications to the problem to move option constraints to the objective function, which aggregates "soft" option constraint violations. Basically, an option constraint violation is observed when $\Phi_{o,\,t}$ exceeds H_o, for an option o in a period t. It is thus straightforward to define the objective function coefficient, or the total option constraint violation penalty, as $\max\{\Phi_{o,t} - H_o, 0\}$ for each $(o,\, t)$ pair.

Fliedner and Boysen (2008) highlight a potential problem with simply counting the number of violations for $(o,\, t)$ pairs and propose a new objective function counting such violations only if the variant scheduled in a position requires the over-demanded option. The authors solve the arising optimization problem via branch-and-bound. Bautista et al. (2008a) define upper-over-assignment and upper-under-assignment that measure deviations from maximum option allowances, and then combine them in a weighted objective function. This approach is similar to the multi-level just-in-time level scheduling problem, except requirements from lower levels are counted only within option windows instead of from the beginning of sequence. The authors then solve this problem via a beam search heuristic. Thiruvady et al. (2014) develop a hybrid Lagrangian-ACO matheuristic for the same problem. Bautista et al. (2008b) extend this formulation by defining a minimum number of occurrences in each option window and adding lower-over-assignment and lower-under-assignment terms to the objective function and solve the arising problem via a GRASP heuristic. We refer the reader to Gagné et al. (2006) and Golle et al. (2014), and the references therein for further reading on *soft-constraint* approaches to the CSP.

2.2. Level scheduling

The level scheduling problem (LSP) traces back to Monden's (1983) seminal book on the Toyota Production System. It has been studied in several forms and under various names such as just-in-time scheduling, mixed-model assembly line scheduling, and production smoothing (Yavuz and Akçali, 2007). Level schedules help achieve the much desired one-piece-flow in unconstrained systems and approximate it in constrained systems.

Toyota's level scheduling approach focuses on reducing the variability in the demand for subassemblies used at the final stage created by the final assembly schedule. The LSP is naturally multi-level, is commonly known by the name output rate variation (ORV) and is $\mathcal{NP}$-hard (Kubiak, 1993). The first exact algorithm developed for the ORV is a dynamic programming procedure of Kubiak et al. (1997). Stronger mathematical properties of the problem are shown and incorporated into improved dynamic programming procedures by Fliedner et al. (2010); Miltenburg (2007). A recent Branch-and-Bound algorithm developed by Pereira and Vilà (2015) uses multiple improved lower and upper bounds and is the most capable solution developed for the ORV so far.