Flexible Services and Manufacturing Journal (2023) 35:320–342
https://doi.org/10.1007/s10696-021-09438-3

Flexible scheduling of diagnostic tests in automotive manufacturing

Simone König[1,2] · Maximilian Reihn[2] · Felipe Gelinski Abujamra[2] · Alexander Novy[2] · Birgit Vogel-Heuser[1]

Accepted: 6 December 2021 / Published online: 16 January 2022

Abstract

The car of the future will be driven by software and offer a variety of customisation options. Enabling these customisation options forces modern automotive manufacturers to update their standardised scheduling concepts for testing and commissioning cars. A flexible scheduling concept means that every chosen customer configuration code must have its own testing procedure. This concept is essential to provide individual testing workflows where the time and resources are optimised for every car. Manual scheduling is complicated due to constraints on time, predecessor-successor relationships, mutual exclusion criteria, resources and status conditions on the car engineering and assembly line. Applied methods to handle the mathematical formulation for the corresponding industrial optimisation problem and its implementation are not yet available. This paper presents a procedure for automated and non-preemptive scheduling in the testing and commissioning of cars, which is built on a Boolean satisfiability problem on parallel and identical machines with temporal and resource constraints. The presented method is successfully implemented and evaluated on a variant assembly line of an automotive Original Equipment Manufacturer. This paper is the starting point for an automated workflow planning and scheduling process in automotive manufacturing.

Keywords Non-preemptive scheduling · Multiple constraints · Boolean satisfiability problem · Automotive testing · Car manufacturing

✉ Simone König
simone.k.koenig@daimler.com

[1] Institute of Automation and Information Systems, Technical University of Munich, Munich, Germany

[2] Mercedes-Benz AG, Böblingen, Germany

1 Introduction and motivation

The automotive industry is affected by the increasing importance of automotive software (Ebert and Favaro 2017). Automotive software allows customers to use more assisted, software-linked car functions during their car journey. These software-linked customer car functions are related to components and electronic control units (ECUs), such as the head-up display. The functions support new concepts for automotive software engineering in manufacturing: the Original Equipment Manufacturers (OEM) offer their customers flexible customisation possibilities and equip the manufacturing processes accordingly.

Automotive manufacturing should therefore be able to produce, test, and commission software-linked, customised cars. As every chosen configuration code may need its own testing procedure to complete the car, such as the commissioning of a seat massage, it is essential to provide time and resource-optimised individual testing workflows for any given car. Those testing workflows consist of software and worker-guided process steps. Nowadays, all of these tests are written into a schedule by hand with a preferably short test time. To minimise factory costs, the schedule of the test procedures could be optimised with respect to the total test time within a factory by applying an automated scheduling procedure.

The main contribution of this paper is a method for the automated and flexible scheduling of diagnostic tests on cars subject to several constraints. The constraints relate to time, direct predecessors, set of all predecessors, mutual exclusion criteria, resource and status conditions for the car engineering and assembly line. The corresponding mathematical model can be formulated as a Boolean satisfiability problem on parallel identical machines. This is the first use case to describe flexible scheduling on parallel identical machines with complex constraints for testing and commissioning in car manufacturing.

The paper is structured as follows: Section 2 introduces the background to scheduling, business challenges and deduces resulting requirements of scheduling for testing and commissioning in automotive manufacturing systems. Section 2 hereby outlines the research questions of this work. The related work is discussed in Section 3. Section 4 describes the mathematical model for scheduling diagnostic tests. A novel automated and flexible scheduling procedure based on the mathematical model is presented in Section 5. The computational study is illustrated in Section 6. The results of the study are presented and discussed in the following Section 6. The final Section 7 concludes this paper.

2 Problem definition

Section 2 introduces the basic concept of production planning in manufacturing. We explain the technical requirements arising from expert discussions that need to be fulfilled for scheduling in testing and commissioning. We can then select an appropriate mathematical concept based on the defined technical requirements.

2.1 Background to scheduling and real-time operating systems

The process of translating customer orders into manufacturing jobs on specific dates is called production planning (Lawlor 1973). Production planning creates a plan that describes the necessary, ordered set of jobs required to accomplish a specific goal in terms of customer orders, given certain starting conditions. Scheduling is a subprocess of production planning, dealing with the allocation of resources to jobs over given time periods with respect to the optimisation of one or more objectives (Błażewicz 2007). A typical optimisation problem in industry is minimising the makespan of a job schedule, which can be interpreted as the time needed from the start to the end of the schedule (Hillier and Herrmann 2006). Scheduling is broadly applied in open, flow and job shop production problems (Brucker 2007).

Scheduling in manufacturing can be interpreted as non-preemptive scheduling in real-time operation systems (Quilliot et al. 2021). To narrow down related work, we continue with the industrial challenges and derive requirements from these.

2.2 Challenges and requirements in the complex scheduling of diagnostic tests

Modern automotive manufacturers offer their customers a wide range of car customisation options. A fixed production plan is therefore no option in flexible manufacturing systems, or in standardised car diagnostic testing schedules for electric and electronic car components (challenge C_1). As every choice of configuration set may need its own testing procedure, manufacturing systems must provide time and resource-optimised individual testing schedules for any given car (challenge C_2).

Fig. 1 shows three exemplary testing scenarios on an automotive assembly production line. In test location 1, the maximum number of tests is conducted on the fully equipped car with seven scheduled tests. A fully equipped car has more electrical and electronic components than others and thus, for cars with less

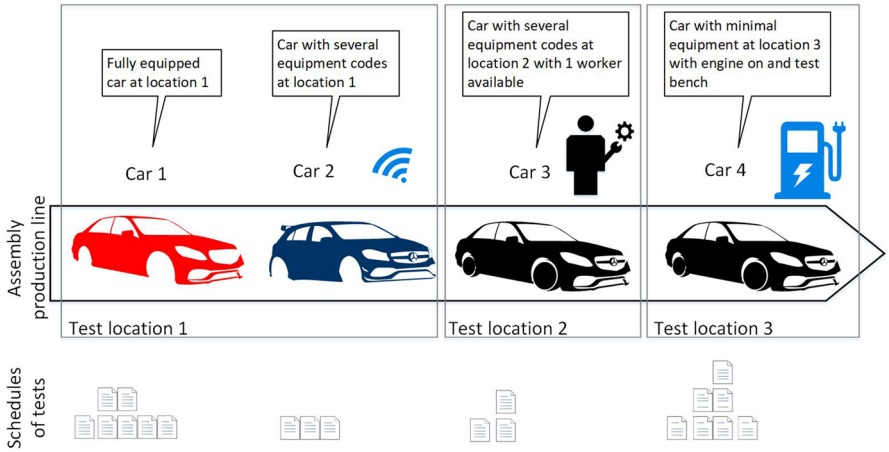


Fig. 1 Scheduling scenarios in automotive assembly lines - a document icon symbolises a diagnostic test

equipment codes, fewer tests need to be scheduled. In test location 2, there is a schedule with a worker-guided test. In test location 3, the smallest scope of tests is conducted on the cars with the least equipment. The tests are performed on a car with motor status *on* and in communication with a test bench.

The tests should be performed in parallel at all test locations to minimise testing run times if feasible. In the following, we will take a closer look at the constraints that decide on the parallel execution of tests.

Expert discussions were held with engineers and programmers to address the challenges C_1 and C_2 and resulted in six requirements for the creation of a procedure to schedule diagnostic tests in automotive manufacturing. We introduced a flexible manufacturing system (FMS) to handle challenge C_2: a FMS is an integrated group of processing computer, numerical control machines and material handling equipment controlled by computers for the automatic processing of parts (ElMaraghy and Caggiano 2016). In an FMS, testing schedules consist of equipment-dependent tests and can be computed event-driven.

Six technical requirements will be formulated with mathematical expressions in the following to handle challenge C_1. These explicit requirements arise from expert knowledge in automotive diagnostic test scheduling. The baseline scheduling model is built by an engineer who writes all tests into a software sequence schedule by hand, preferably with a short test time. It is based on experience.

Schedules were either created on demand or beforehand to plan machine usage better. Thus, the following explicit requirements form the technical basis of the mathematical optimisation model for flexible scheduling in this work. To formulate the constraints, we list all of the basic mathematical notation of this work in Table 1.

Table 1 Relevant notation in this work

$I = \{1, \dots, N\}$	Set of $N \in \mathbb{N}$ tests
$K = \{1, \dots, S\}$	Set of $S \in \mathbb{N}$ machines
$J = \{1, \dots, K\}$	Set of $K \in \mathbb{N}$ resources
$L = \{1, \dots, Z\}$	Set of $Z \in \mathbb{N}$ status conditions of objects
$\{t_i\}_{i \in I}$	$t_i \in \mathbb{R}_{>0}$ units of time that test $i \in I$ takes
$\{pre_i\}_{i \in I}$	$pre_i \in I$ direct predecessor of test $i \in I$
$\{P_i\}_{i \in I}$	$P_i \subset I$ set of predecessors that must end before test $i \in I$ can start
$\{M_i\}_{i \in I}$	$M_i \subset I$ set of tests that may not run parallel to test $i \in I$
$\{r_{i,j}\}_{i \in I, j \in J}$	$r_{i,j} \in [0, 100]$ resources temporarily taken by test $i \in I$ on resource $j \in J$
$\{c_{i,l}\}_{i \in I, l \in L}$	$c_{i,l} \in \{turn\_on, turn\_off, req\_on, req\_off\}$ condition status of test $i \in I$ on kind of condition $l \in L$
$\{s_i\}_{i \in I}$	$s_i \in \mathbb{R}$ start time of test $i \in I$
$\{e_i := s_i + t_i\}_{i \in I}$	$s_i \in \mathbb{R}$ end time of test $i \in I$
$\{T_i := [s_i, e_i]\}_{i \in I}$	$T_i \in \mathbb{R}^2$ time interval of test $i \in I$
$e_{\max} = \max_{i \in I} \quad e_i$	Makespan, the total length of a schedule
C	Set of constraints
$\Theta_C \in \mathbb{R}_{\geq 0}^{\lvert I \rvert}$	Feasible solution for a given set of constraints C

We define the finite set of N individual tests that are scheduled for a car at an automotive assembly line as

$$I = \{1, \ldots, N\}, \quad N \in \mathbb{N},$$

and the finite set of K different resources that may be used by a test $i \in I$ as

$$J = \{1, \ldots, K\}, \quad K \in \mathbb{N}.$$

For example, the specific test $i \in I$ may use 20% of the car engine's resource capacity. The finite set of Z different status conditions, which must be considered in the test schedule, is defined as

$$L = \{1, \ldots, Z\}, \quad Z \in \mathbb{N}.$$

For example, the car engine must be turned *on* before testing the air conditioning. The respective test would require the motor status to be *on* as the status condition.

Without loss of generality, we formulate the requirements R_{time}, R_{dir}, R_{pre}, R_{mutex}, R_{res} and R_{status} for the test $i \in I$.

- Requirement R_{time}: a test has a run time
  Test $i \in I$ has a predicted run time $\{t_i\}_{i \in I},\ t_i \in \mathbb{R}_{>0}$. The prediction can be based on expert knowledge and an analysis of historic tests. The run time t_i is a crucial requirement to minimise with respect to the makespan of the test schedule. Note that the more accurate the time predictions are, the more efficient an optimised test schedule will be in the production line.
- Requirement R_{dir}: a test can have a direct predecessor
  Test $i \in I$ may require a test as a direct predecessor $\{pre_i\}_{i \in I},\ pre_i \in I$. Therefore, the end time of pre_i must equal the start time of i. For example, the engine must be brought up to a certain temperature before it is tested under maximum load. There must be short time between the test of heating and the main test of the engine itself.
- Requirement R_{pre}: a test has a set of predecessors
  Test $i \in I$ may require a set of predecessors $\{P_i\}_{i \in I},\ P_i \subset I$, which must be completed before i starts. For example, an ECU should be flashed by software before it is coded to a specific configuration.
- Requirement R_{mutex}: a test can have mutual exclusive tests
  Test $i \in I$ may require a set of mutual exclusive tests $\{M_i\}_{i \in I},\ M_i \subset I$, which may not run parallel to i. For example, you cannot simultaneously test whether the air conditioning can precisely regulate to a predefined high or low temperature.
- Requirement R_{res}: a test has resource criteria
  Test $i \in I$ may temporarily consume a fixed amount $\{r_{i,j}\}_{i \in I, j \in J},\ r_{i,j} \in [0, 100]$ of automotive bus resource $j \in J$. At no point in the testing schedule may the consumed amount of resources be greater than 100% on any of the resources.
- Requirement R_{status}: a test has status criteria
  A test $i \in I$ may require or change a status of a reference object $l \in L$, i.e., $\{c_{i,l}\}_{i \in I, l \in L},\ c_{i,l} \in \{require\_on,\ require\_off,\ turn\_on,\ turn\_off,\ any\}$. For exam-

ple, if test $i \in I$ has the status $c_{i,l} = require\_on$ for $l = engine$, another test $j \in I$ with status $c_{j,l} = turn\_on$ must be run beforehand. No test $m \in I$ with status $c_{m,l} = turn\_off$ has to be performed in between. For example, before performing a worker-guided test, at least one worker must be present at the test location and thus be set to the state *turn_on*.

The scheduling of N tests is processed on $K = \{1, \ldots, S\}$, which is the set of $S \in \mathbb{N}$ identical and parallel machines. Each test $i \in I$ has a run time $\{t_i\}_{i \in I}$ and can only be processed by one processing unit of the FMS at a time. We specify that the machines are identical, so that processing $i \in I$ takes time $\{t_i\}_{i \in I}$ on any machine (Shmoys et al. 1991).

We define C as a fixed and finite set of requirements and will refer to s_i as the start time and $e_i := s_i + t_i$ as the end time for the test $i \in I$. Therefore, the time interval of test $i \in I$ is denoted as $\{T_i := [s_i, e_i]\}_{i \in I}$ with $T_i \in \mathbb{R}^2$. A feasible solution of the start time $\{s_i\}_{i \in I}$ with respect to constraints in C is called $\Theta_C \in \mathbb{R}^{|I|}_{\geq 0}$. We minimise the schedule with respect to the makespan

$$e_{\max} = \max_{i \in I} \quad e_i. \tag{1}$$

An optimal solution for a set of constraints given an objective is called $\hat{\Theta}_C$.

Every test has a predicted run time, as stated in requirement R_{time} and can inherit any combination of further constraints. Fulfilling those requirements will ensure a faultless testing procedure and exclude the risk of damage to any of the components.

We classify the scheduling problem in terms of the three-field notation of Brucker (2007). A scheduling problem is classified in $\alpha|\beta|\gamma$. α determines the machine environment. In this work, we assume $\alpha = \{\alpha_1 = K\}$ because of identical parallel machines. Job characteristics are determined by $\beta = \{\beta_2 = prec, \beta_3 = const\}$. In this work, directed acyclic graphs are of interest so that $\beta_2 = prec$. The specification of release dates is also of interest, in combination with the car's state of construction on the assembly line, i.e., $\beta_3 = const$. The optimality criterion is the minimal completion time of the test schedule, i.e., $\gamma = e_{\max}$. As proposed by Brucker (2007), we ignore the empty symbols in the three-field notation.

2.3 Research questions

We assume that the fulfilment of the six outlined requirements R_{time}, R_{dir}, R_{pre}, R_{mutex}, R_{res} and R_{status} forms the theoretical basis for a procedure to schedule diagnostic tests flexibly in automotive manufacturing. To investigate the complex scheduling problem, we formulate the research questions $RQ1$ and $RQ2$ as follows:

$RQ1$: Which mathematical model describes the scheduling problem, taking into account the requirements R_{time}, R_{dir}, R_{pre}, R_{mutex}, R_{res} and R_{status}, and how can a numerical solution be found?

$RQ2$: What are the key elements of the procedure for scheduling diagnostic tests during the assembly line production in automotive companies based on the model provided by $RQ1$?

3 Related work

Based on the requirements of Section 2.2, we identified the criteria relevant for the classification of related work (see Table 2).

Scheduling in automotive use cases has been extensively discussed in literature. Shi et al. (2017), for example, developed test plans with tight schedules that combine multiple diagnostic tests on cars to fully utilise all of the available time in prototype car test scheduling. Moreover, Spieckermann et al. (2004) dealt with the scheduling problem of operating paint shop systems with sequence-dependent set-up costs.

Schedules of diagnostic tests in automotive production lines contain safety and certification-relevant test steps. This is why we prefer exact deterministic mathematical methods in this paper. The constraints implicated by the direct predecessor R_{dir} and the set of predecessors R_{pre} are explained exhaustively in any context of linear programming (Vanderbei 2020). The mutual exclusion constraint R_{mut} can be formulated using binary helper variables.

We will develop a method to translate the requirements of Section 2.2 into a piecewise linear model (mixed-integer model) for every requirement except R_{res}, which will be formulated as a logical constraint.

To do so, the mathematical model of this paper will be formulated as a *Boolean satisfiability* (SAT) problem and will be solved accordingly, e.g., with the Generic seaRch Algorithm for the Satisfiability Problem (GRASP) of Marques-Silva and Sakallah (1999). An overview on the theory and applications of SAT problems is provided by Pulina and Seidl (2020). Huang and Zhou (2018), for example, provide an efficient SAT encoding method for complex job-shop scheduling.

The group of SAT problems is proven to be NP-complete by Cook (1971). This means that it is possible to verify any given solution in polynomial time, but no algorithm exists that can find one such solution in polynomial time. As the algorithm engineering of modern SAT solvers has improved over recent years, the real world running time in the present use case of static scheduling is of negligible importance (Alouneh et al. 2019).

Weckenborg et al. (2020) discussed the automotive capacity scheduling of prototype vehicle production at the Volkswagen Pre-Production Center. Resource allocation, the selection and scheduling of orders for prototype vehicle production are of interest here. Moreover, Weckenborg et al. (2020) proposed a spreadsheet-based decision support system for daily capacity scheduling. Nevertheless, the setting is different to our work in terms of the application and the criteria of Table 2.

Buergin et al. (2018) introduced an optimisation model for the local order scheduling of mixed-model aircraft assembly lines covering both assignment to lines as well as sequencing with respect to cost. Requirements R_{time}, R_{pre} and R_{res} are the focus of the work of Buergin et al. (2018).

Dörmer et al. (2015) covered the problem of master production scheduling for high-variant, mixed-model automotive assembly lines. Individual, customer-defined models of a basic product type are assigned to short-term production

Table 2 Classification of related work - symbol "X" means applicable, symbol "-" means not applicable or not specified, P.I.M. = Parallel identical machines for batch scheduling

Reference	Application	Solving method	P.I.M.	R_{time}	R_{dir}	R_{pre}	R_{mutex}	R_{res}	R_{status}
This paper	Car diagnostics testing	GRASP	X	X	X	X	X	X	X
Weckenborg et al. (2020)	Automotive capacity scheduling	Binary integer programming	-	X	-	-	-	X	X
Buergin et al. (2018)	Local order assignment in aircraft manufacturing	Mixed-model sequencing	-	X	-	X	-	X	-
Dörmer et al. (2015)	Automotive master production	Heuristics	-	X	-	-	-	X	-
Wang and Liu (2015)	Production scheduling and preventive maintenance	Genetic	X	X	-	X	-	X	-
Bartels and Zimmermann (2009)	Automotive prototypes	Priority heuristic	-	X	-	X	-	X	-

periods while anticipating the negative impacts of an unbalanced model sequence at the lower planning level. The focus is on workload and processing times at the stations on the assembly lines with respect to cost, thus the requirements R_{time} and R_{res} are fulfilled.

Wang and Liu (2015) described a multi-objective, parallel machine scheduling problem with resources on machines and moulds, and with flexible preventive maintenance activities on resources. The work deals with parallel identical machines and addresses the requirements R_{time}, R_{pre} and R_{res}.

Bartels and Zimmermann (2009) covered an approach to experimental cars based on multi-mode, resource-constrained project scheduling with minimum and maximum time lags as well as renewable and cumulative resources, i.e., requirements R_{time}, R_{pre} as well as R_{res} are studied. However, the objective is different as scheduling is used to minimise the number of automotive prototypes used in Bartels and Zimmermann (2009).

To the best of the authors' knowledge, no literature exists on combining the constraint criteria R_{time}, R_{dir}, R_{pre}, R_{mutex}, R_{res} and R_{status} into a deterministic comprehensible method which optimises the makespan of the test schedule. There is no suitable approach to fulfill all of the criteria in Table 2 for automotive batch scheduling on assembly lines.

4 Model formulation and solution for flexible scheduling

We will build the full model, which describes the input and output data with regard to the conditions and requirements of Section 2. Without loss of generality, the constraints will be formulated for a test $i \in I$.

4.1 Constraints for tests as model input

Referring to the problem description in Section 2.2, the input data can be represented in a table of length $|I|$. Every test $i \in I$ has constraints and requirements that are indicated in this table. There is no specific requirement for the input format or data type as it depends on the realised implementation.

4.2 Objectives for the mathematical model

The objective of the optimisation in this work is to calculate the start time s_i, given the input table, so that the makespan is minimised as defined in Eq. 1, formally

$$\min_{\{s_i\}_{i\in I}} e_{\max} \;=\; \min_{\{s_i\}_{i\in I}} \max_{i\in I} e_i. \tag{2}$$

This ensures that the testing schedules are as cost efficient as possible, whereby cost is related to minimal run time.

To force the constraint R_{dir} of the direct predecessor pre_i of test $i \in I$, we require

$$e_{pre_i} = s_i. \tag{3}$$

Next, we formulate the requirement R_{pre} for the set of predecessors P_i of test $i \in I$,

$$e_p \leq s_i \quad \forall p \in P_i. \tag{4}$$

So far, this is a strictly linear model and a solution Θ_C for such could be found using the Simplex method in linear programming (Vanderbei 2020), given a linear objective function. Note that Eq. 2 is not linear.

The helper variables have to be defined for the mutual exclusion requirements R_{mutex} of test $i \in I$. We define

$$\tilde{M} := 1 + \sum_{i \in I} t_i \tag{5}$$

as a helper variable in order to process the constraints '*either or*': To enforce the mutual exclusion requirement for test $i \in I$, every test in M_i must '*either end before i*' or '*start after i*'. With the help of a binary variable $\tilde{h}_{i,m} \in \{0, 1\}$, the '*either or*' constraint is translated to

$$e_m \leq s_i + \tilde{M} * \tilde{h}_{i,m}, \quad \text{and} \tag{6a}$$

$$e_i \leq s_m + \tilde{M} * (1 - \tilde{h}_{i,m}), \quad \forall m \in M_i. \tag{6b}$$

We forego a formal proof, but will explain shortly: According to Definition 5, it holds that

$$e_{max} \leq \sum_{i \in I} t_i < \tilde{M}. \tag{7}$$

$\sum_{i \in I} t_i$ is the upper limit of the makespan, but note that the trivial chaining of tests is not necessarily a feasible solution. However, every feasible solution Θ_C is shorter or equal in time.

We start with $\tilde{h}_{i,m} = 0$. Eq. 6b is then automatically fulfilled so that test m must end before test i starts.

Vice versa, assuming $\tilde{h}_{i,m} = 1$. As $e_m \leq e_{max}$ and $s_i \geq 0$, it holds that for fixed $i, m \in I$, Eq. 6a is

$$\begin{aligned} & e_m \leq s_i + \tilde{M} \\ \iff \quad & e_m - s_i \leq \tilde{M} \\ \overset{Def.\ 5}{\iff} \quad & e_m - s_i \leq \sum_{i \in I} t_i \ < \ \tilde{M}, \end{aligned}$$

and therefore, the constraint will always be fulfilled no matter what s_i, s_m is. In that case, Eq. 6b is enforced, so that test m must begin only after test i is finished.

We have successfully translated '*either* Eq. 6a *or* Eq. 6b' into the system using the binary helper variable $\tilde{h}_{i,m} \in \{0, 1\}$. The system of constraints containing the

mutual exclusion requirements is no longer a strictly linear model, but rather a mixed-integer linear (MILP) system. It must be solved with suitable methods, e.g., a combination of the Simplex method and Branch and Bound method of Integer Programming (Hu and Kahng 2016).

In the next step, we want to translate the status requirement R_{status} into mathematical constraints, which will keep the system a MILP problem. To do so, we have to interpret what possible status condition the test $i \in I$ can have. As defined in R_{status} for a test $i \in I$ and a status $l \in L$, the test $i \in I$ can either require the status condition to be '*off/on*', but can also turn the condition '*off/on*'. Note that it may also hold the '*any*' requirement, which defines the status condition l to be of no concern to test i. Note that by definition, the status condition, if changing, is updated when the test is completed and that during that test, no other test which is not of status '*any*' can be run. An intuitive explanation will follow for these conditions based on the Pseudo code illustrated in Algorithm 1 of Appendix A.

At the start of the testing procedure, all status conditions are '*off*' by definition. If there is a test $i_1 \in I$ with condition '*require_on*', there are two possible allocations of the test. Either i_1 is executed before any other test with the status condition '*turn_on*' or i_1 is executed after a test with the status condition '*turn_off*', but before any test with '*turn_on*' is executed in between. On the other hand, a test $i_2 \in I$ with status condition '*require_on*' can only run if a test with status condition '*turn_on*' is run beforehand and no test with status condition '*turn_off*' may run in between. Note that those constraints must hold for every kind of status condition $l \in L$.

There may be multiple '*turn_on*'–'*turn_off*' tests, which is why every possible combination must be taken into account by using '*either or*' constraints as a help again. As the '*either or*' in this case can also be '*at least one of many*', we use a binary helper variable for every possible case and the constraint, so that

$$\sum_{j \in |helper^l|} h_{j,l} := |helper^l| - 1. \tag{8}$$

Allow us to illustrate an example: let $i_1, i_2, i_3, j \in I$ and we want j to only start if at least one of the other tests i_1, i_2, i_3 is finished. The system would be

$$e_1 \leq s_j + \tilde{M} * h_1\,, \tag{9a}$$

$$e_2 \leq s_j + \tilde{M} * h_2\,, \tag{9b}$$

$$e_3 \leq s_j + \tilde{M} * h_3\,, \tag{9c}$$

$$\sum_{k \in \{1,2,3\}} h_k = |\{1,2,3\}| - 1 = 2\,. \tag{9d}$$

It can be seen that at least one of h_1, h_2, h_3 must be equal to 0, and therefore one of Eqs. 9a, 9b or 9c are enforced, while the other two equations are of no concern.

For the resource constraint R_{res}, it would be possible to keep the model strictly piecewise linear and solve it with an MILP method. This has the disadvantage of having an exponential number of constraints depending on the number of tests that use a resource. We therefore introduce a logical constraint $\tilde{r}_{t,i,j}$, which forces the model to be solved as a SAT. For every test $i \in I$ we create an interval $T_i = [s_i, e_i]$ and define

$$\tilde{r}_{t,i,j} := \begin{cases} r_{i,j} & \text{if } t \in T_i \\ 0 & \text{else,} \end{cases}$$

to enforce

$$\sum_{i=1}^{|I|} \tilde{r}_{t,i,j} \leq 100 \quad \forall j \in J \quad \forall t \in [0, e_{max}]. \tag{10}$$

Implementing this constraint depends on the SAT solver or library that is used. In this work, the solver used by the implemented library of Perron and Furnon (2019) is the conflict-driven clause learning (CDCL) algorithm, which was developed by Marques-Silva and Sakallah (1999).

We have formally established the constraints of the system and will now be able to use a SAT solver to find an optimal solution.

4.3 Schedule of diagnostic tests as model output

The optimised schedule includes the start times s_i of the tests $i \in I$. As the run time t_i for a test $i \in I$ is only a prediction based on historic data, real-world testing time may depend on many external non-controllable factors such as worker interaction. Thus, an output format is needed that still forces the test schedule to adhere to all requirements. Hence, an adjacency list is introduced which contains a list of predecessors for every test. In order for test i to be started, every test in the predecessor list of test i must have finished. An exemplary adjacency list is illustrated in Table 3. The derived schedule begins with tests 1 and 2, which have no predecessors. Thus, they can be run in parallel. When test 1 is finished, test 3 can start. When tests 2 and 3 are finished, test 4 can finally start.

We provide a Pseudo code in Algorithm 2 of Appendix A to translate the optimised start times to an adjacency list. The makespan of the adjacency schedule is

Table 3 Adjacency list as representation of achieved schedule

Test		Predecessors
1	←	None
2	←	None
3	←	1
4	←	2 3

equal to the makespan of the start time $e_{\max}$, if the predicted times $\{t_i\}_{i \in I}$ are used to calculate the makespan.

5 A novel automated scheduling procedure

Having derived the mathematical scheduler in Section 4, we now describe the overall procedure to schedule diagnostic tests for the testing and commissioning of electronic components in automotive manufacturing. The mathematical model is hereby the central element.

The novel procedure consists of seven steps and two decisions (compare Fig. 2). We run through the procedure with an exemplary set of N tests $I = \{1, \ldots, N\}$ where $I \subset I_{all}$ and I_{all} are the set of all existent tests over all cars. Moreover, we use automation systems as FMS, most of which are structured hierarchically in the so-called automation pyramid according to ISA-95 (ISA 2000).

The procedure for scheduling diagnostic tests on a selected car starts with *Step 1*. The set of tests I, that are performed on a specific car, depends on the configuration codes of the car. The configuration codes are linked to the tests, and this linked information is the input data for the first step. In *Step 2*, the location of the car on the assembly line is determined with the help of a production system at the supervisory level (ISA 2000). In *Step 3*, a production system at the planning level offers the constraints C for the configuration code-dependent tests of the car on the assembly line. The constraints C are necessary to schedule N tests for a car with respect to the business and engineering requirements R_{time}, R_{dir}, R_{pre}, R_{mutex}, R_{res} and R_{status}.

In *Step 4*, the tests are scheduled with the objective of minimising the overall test run time. The mathematical model behind *Step 4* has been highlighted extensively in Section 4. The schedule is represented as an adjacency list.

In *Step 5*, the resulting schedule of N tests for the car on the assembly line is transferred for execution with a controller at the supervisory level. A code repository can support with binaries for each test. In *Step 6*, the results of the executed schedule are analysed and documented. In *Decision 1*, a check is carried out as to whether there is at least one test with a result equal to 'not OK'. If so, *Decision 2* checks whether a repetition of the test or tests with the result equal to 'not OK' is feasible. The repetition can depend on the new position of the car in the assembly line production at the time of *Decision 2*. If the repetition is feasible, the tests with the result equal to 'not OK' are repeated for the car on the given assembly line by going back to *Step 2*. If all test results are 'OK', *Decision 2* is skipped. In *Step 7*, the N test processes have been carried out on the car. The procedure is finished.

6 Industrial evaluation

We would now like to describe the industrial evaluation of the mathematical model of Section 4 and the procedure from Fig. 2 for an automotive case study with an OEM.

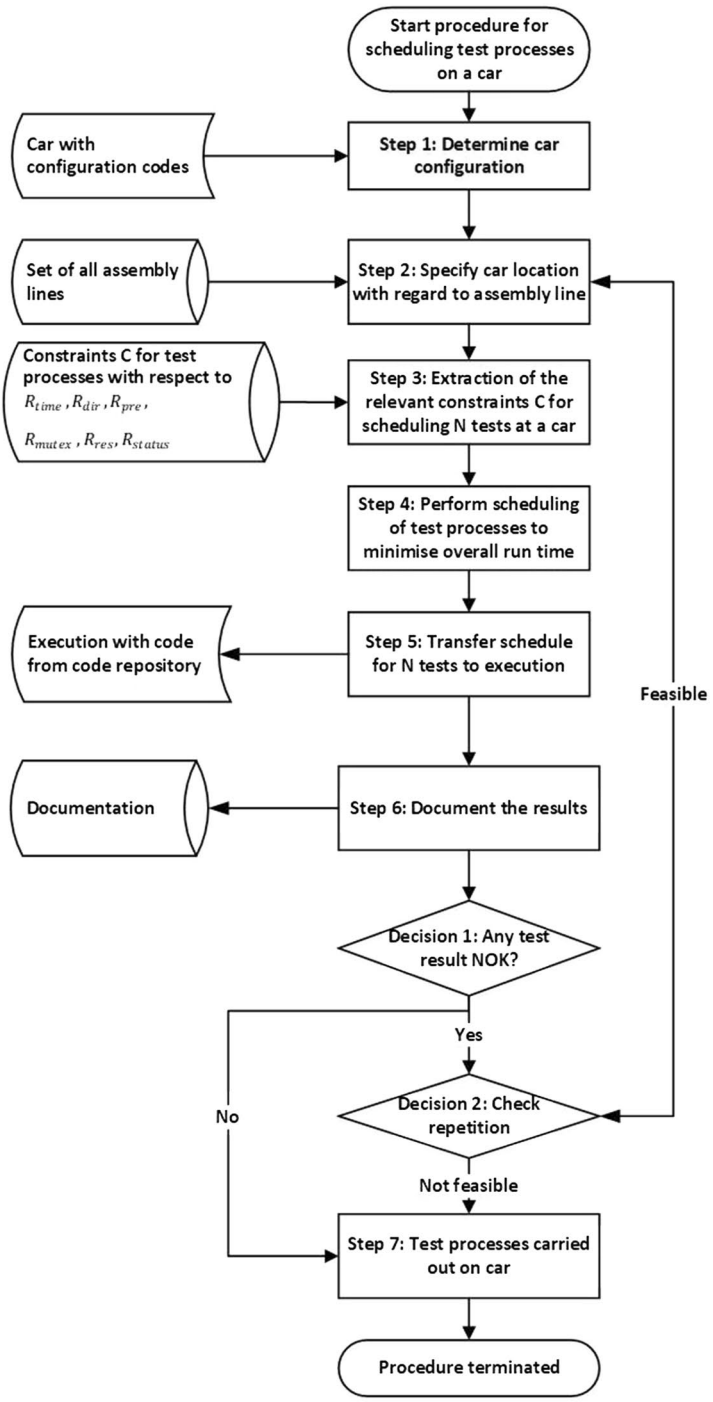


Fig. 2 Flowchart of scheduling procedure to run diagnostic tests more flexibly and reduce overall run time

6.1 Details of computational implementation environment

A prototype of the mathematical model in Section 4 was implemented to validate this work. We used a Lenovo ThinkPad T570 with 24GB RAM and an Intel i5-6300U processor to illustrate the run time efficiency of the method. In the computation environment, we used Google OR - Tools (Perron and Furnon 2019) for implementation in Python 3.8 as the number of constraints may rise to 10^3. As suggested in the benchmark test of Da Col and Teppan (2019), there is no real-world run time problem, even if one would expect the number of constraints to rise gradually with the development of electrified cars and more complex assembly lines.

6.2 Case study-based industrial evaluation

Fig. 2 illustrates three types of input parameters: the car with its configuration codes, the set of all test locations in the factory, and the boundary constraints for the tests that are performed on the car at a test location. In the following real-world case study, we therefore look at the procedure from two angles: first, we select a specific test location at the automotive assembly line. The data related to the tests at the chosen test location, are illustrated in Table 4. This case study allows us to evaluate the mathematical model presented in Sect. 4 in detail.

Second, we look at the associated relationships with the tests and the number of all cars produced in the selected production hall at the German location over a fixed period. We hereby evaluate the procedure illustrated in Fig. 2.

We begin by looking at the test location in the interior installation of the assembly line production that contains representative tests of automatic and worker-guided test scopes. Table 4 gives an overview of the tests for the case study.

We find $I = \{1, \ldots, N = 21\}$ tests. Each of which is described by an indication of time (expressed through the column *time* [*s*]), necessary repetition (column *run*), predecessor-successor relationships (columns *precond* and *previous*), exclusion of related tests (column *mutex*), resources of automotive gateway systems (columns *gate1load_resource* in unit [%] and *gate2load_resource* [%]). The case study contains worker-guided tests 3, 4, 5, 6, 16, and 17 (column *worker_status*).

In this setting, there is only one worker available at the test location, so that only one worker-guided test can be run in parallel. We consider no tests requiring communication with test benches. Automated tests can contain functions such as coding and flashing. Tests 18, 19, 20, and 21 are help tests in the chosen test location: test 18 turns the ignition of the car on and test 19 turns it off. Test 20 requires the existence of a worker and test 21 releases it.

All tests in Table 4 except for tests 18, 19, 20, and 21 are carried out with the ignition on (column *ign_condition_status*). Which test can be tested on a car depends on the car configuration. There are 128 possible schedules in this case, which will be reduced to feasible combinations when matching them with the actual equipment codes of a fixed car volume. On the one hand, the test of the exit lights is performed in tests 9 and 10, and can be carried out automatically depending on a

Table 4 Implementation details for industrial case study on an automotive assembly line

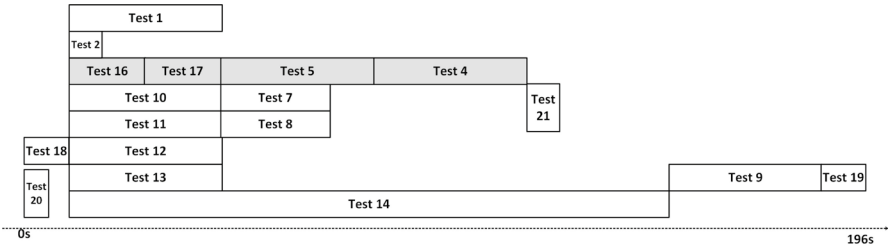
Tests I	time [s]	precond	previous	mutex	gate1load_ resource [%]	gate2load_ resource [%]	worker_status	ign_condition_status
1	10	none	none	none	1	0	any	req_on
2	1	none	none	none	1	0	any	req_on
3	10	4, 12	none	3, 4, 5, 6, 16, 17	1	0	req_on	req_on
4	10	5, 13	none	3, 4, 5, 6, 16, 17	1	0	req_on	req_on
5	10	6, 11	none	3, 4, 5, 6, 16, 17	1	0	req_on	req_on
6	10	10	none	3, 4, 5, 6, 16, 17	1	0	req_on	req_on
7	7	12	none	none	1	0	any	req_on
8	7	10	none	none	1	0	any	req_on
9	10	none	none	none	100	100	any	req_on
10	0	none	none	none	2	0	any	req_on
11	10	none	none	none	2	0	any	req_on
12	10	none	none	none	2	0	any	req_on
13	10	none	none	none	2	0	any	req_on
14	180	none	none	15	20	20	any	req_on
15	10	none	none	14	20	20	any	req_on
16	5	none	none	3, 4, 5, 6, 16, 17	1	0	req_on	req_on
17	5	none	none	3, 4, 5, 6, 16, 17	1	0	req_on	req_on
18	3	none	none	none	0	0	any	turn_on
19	3	1 – 17	none	none	0	0	any	turn_off
20	0	none	none	none	0	0	turn_on	any
21	0	3, 4, 5, 6, 16, 17	none	none	0	0	turn_off	any

specific configuration code. On the other hand, the exit lights are executed manually in tests 3 and 6. In the following, we consider a set of the volume of E Class models produced at a German plant over the period February-May 2021.

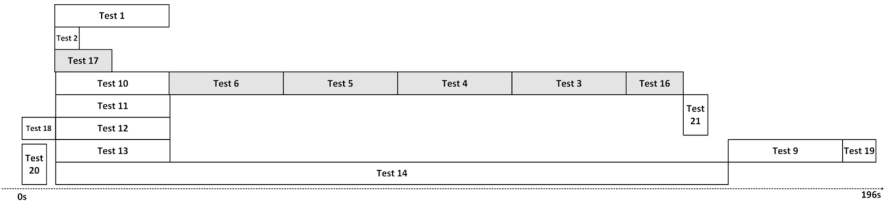
6.3 Results of the case study at the OEM based on the implementation of the scheduling procedure

We highlight *Steps 1-3* of the procedure (see Fig. 3). Considering the fixed car volume in the German production plant, two out of 128 feasible condition sets from Table 4 remain. This means that we have two scheduling variants. We determine, that test 15 is not relevant for the cars on hand in the period since the underlying configuration code is not installed. On the one hand, the test of the exit lights in tests 9 and 10 can be carried out automatically depending on a specific configuration code. On the other hand, the exit lights are executed manually in tests 3 and 6 if the specific configuration code is not present for a car. In 3% of the fixed car volume, the test set $I_1 = \{1, 2, 4, 5, 7, 8, 9, 10, 11, 12, 13, 14, 16, 17, 18, 19, 20, 21\}$ is scheduled in *Step 4*, whereas the set $I_2 = \{1, 2, 3, 4, 5, 6, 7, 8, 11, 12, 13, 14, 16, 17, 18, 19, 20, 21\}$ is scheduled in 97% of the cars.

Having applied the mathematical model presented in Section 4 on test sets I_1, I_2, the resulting schedules can be visualised in Fig. 3. The model terminated with the optimal status, meaning the shortest possible schedule, has been found in both test sets. Note that it is possible for more than one solution to be of optimal length. As already



(a) Version I_1 with two more automated tests 9,10



(b) Version I_2 with two more worker-assisted tests 3,6

Fig. 3 Two schedules as Gantt charts for the presented automotive case study – worker assistance is illustrated in grey, test 14 is not to scale

mentioned, the individual test times are only predictions. If the test times differ from the predictions in a scheduled procedure, a different schedule may have been quicker given the real world times. Hence, it is important to be as precise as possible when predicting the test run times.

The optimisation problem with respect to I_1 is solved on average over 10 runs in $0.0092ms$ and I_2 in $0.0165ms$.

6.4 Discussion

The results of the previous Section 6 are discussed and the work is assessed against the requirements formulated in Section 2.2 on a per-concept basis.

We have created a mathematical model to address the run time of a test (R_{time}), direct predecessors (R_{dir}), the set of all predecessors (R_{pre}), mutual exclusive tests (R_{mutex}), resource constraints (R_{res}) and status conditions (R_{status}) per definition. The resulting schedules of Fig. 3 have been confirmed manually, since the example that is illustrated in Table 4, is rather simple. The research question $RQ1$ is answered by definition.

Without considering the resource constraint, a branch-and-cut would be most appropriate, as has been discussed extensively in Padberg and Rinaldi (1991). The scalability and complexity is also well understood and discussed (Basu et al. 2021). Adding the logical constraint of the resources makes the optimisation more complex. In terms of the scope in the automotive industry, the limits of the model are out of reach as thorough benchmark tests have been performed and analysed in Da Col and Teppan (2019). Hence, even if added complexity is assumed in the future by a more complex car architecture, the scalability will not pose a problem in this scope.

In terms of abstract complexity, scalability is dependent on the computational complexity of algorithms expressed in Big O-notation. General SAT problems are proven to be NP-complete by Cook (1971). Therefore, solving will have complexity $\mathcal{O}(2^N)$ in the worst case, but since no methods exist to reduce complexity and increase efficiency, benchmark tests as in Da Col and Teppan (2019) are of greater significance.

Furthermore, we proposed a procedure with key elements to schedule the tests on an assembly line for automotive companies based on the mathematical model provided in Fig. 2. We decided to apply static scheduling to each car, so that *Steps 1-6* of the procedure are straightforward. In the evaluation study, we were unable to test *Decisions 1 and 2* as the rescheduling policy since our test setting was not event-driven. However, the outlined procedure shows future possibilities to integrate the static scheduler into an event-driven production system. As the given scheduling procedure does not focus on a specific technology, it is transferable to other scenarios in different industries with scheduling processes. This brings us back to the research question $RQ2$ that we analysed as well.

7 Conclusion and future work

In this paper, we have described the development, implementation and successful evaluation of a mathematical optimisation model as basis for the flexible scheduling of automotive diagnostic tests on an assembly production line. The approach considers multiple scheduling constraints on the car engineering and assembly line related to time, direct predecessors, set of all predecessors, mutual exclusion criteria, resource and status conditions.

The model presented here is the main contribution of this work and is formulated as a Boolean satisfiability problem. It has been successfully validated on the final assembly line of a German OEM plant. We embedded the model into a procedure to schedule the tests for each car with configuration codes at the test location. By integrating the model into the procedure, we showed that complex scheduling with multi-constraints can be flexible and automated. In general, the schedules generated with the proposed method were better in terms of run time and work effort than the solutions generated by the current manual procedure. After successful test implementation, the OEM estimates the annual cost savings to be substantial.

This is the first use case that describes flexible scheduling on parallel identical machines with complex constraints for testing and commissioning in car manufacturing. Thereby, this work presents a way to integrate the procedure into an event-driven production system. The paper provides a basis for further research into optimisation methods for automated workflow management in automotive manufacturing. The dynamic scheduling of diagnostic tests will be of interest in future.

Appendix A

Data: Lists/dicts/objects
$\{s_i\}, L, I, \{c_{i,l}\}, \{require\_on^l\}_{l \in L}, \{require\_off^l\}_{l \in L}, \{turn\_on^l\}_{l \in L}, \{turn\_off^l\}_{l \in L}$, model
Result: Constraints forcing the system for status conditions
for *l in L* **do**
  for *i in* $require\_on^l$ **do**
    for *p in* $turn\_on^l$ **do**
      model.Add($s_p + t_p \leq s_i + M * hc_{l,i,p}$);
      for *k in* $turn\_off^l$ **do**
        model.Add($s_k + t_k \leq s_p + M * hc_{l,i,p} + M * hc_{l,i,p,k}$);
        model.Add($s_i + t_i \leq s_k + M * hc_{l,i,p} + M * (1 - hc_{l,i,p,k})$);
      end
    end
    model.Add($\sum_{p \in t\_On_j} hc_{l,i,p} = |turn\_on^l| - 1$);
  end
  ;
  for *i in* $require\_off^l$ **do**
    if $len(turn\_off^l) > 0$ **then**
      for *p in* $turn\_off^l$ **do**
        model.Add($s_p + t_p \leq s_i + M * (1 - hc_{l,i,p}) + M * hc_{l,i,p,off}$);
        for *k in* $turn\_on^l$ **do**
          model.Add($s_i + t_i \leq s_k + M * hc_{l,i,p} + M * hc_{l,i,p,off}$);
          model.Add($s_k + t_k \leq s_p + M * (1 - hc_{l,i,p}) + M * hc_{l,i,p,off} + M * hc_{l,i,p,k}$);
          model.Add($s_i + t_i \leq s_k + M * (1 - hc_{l,i,p}) + M * hc_{l,i,p,off} + M * (1 - hc_{l,i,p,k})$);
        end
      end
      model.Add($\sum_{p \in turn\_off^l} hc_{l,i,p} = |turn\_off^l| - 1$);
    else
      for *p in* $turn\_on^l$ **do**
        model.Add($s_i + t_i \leq s_p$);
      end
    end
  end
end

Algorithm 1: Pseudo code for handling of status conditions

Data: List $\{s_i\}$
Result: Adjacency list as dictionary <key>:<predecessors of key>
$adj\_list = \{\}$;
for i *in* I **do**
  $adj\_list[i] = []$;
  for j *in* I **do**
    if $e_j \leq s_i$ *and* $i \stackrel{!}{=} j$ **then**
      $adj\_list[i] \leftarrow j$;
    else
  end
end

Algorithm 2: Pseudo code translation to adjacency list

Author Contributions All authors contributed to the study conception and design. Material preparation, data collection and analysis were performed by SK, MR and FGA. The first draft of the manuscript was written by SK and MR and all authors commented on previous versions of the manuscript. All authors read and approved the final manuscript.

Funding Open Access funding enabled and organized by Projekt DEAL. No funding was received to assist with the preparation of this manuscript.

Data availability All of the data generated for the scheduling method during this study are included in this published article. Confidential data are not included.

Code availability The code that supports the findings of this study are available from Mercedes-Benz AG, but restrictions apply to the availability of the code. However, the code is available from the authors on reasonable request and with the permission of Mercedes-Benz AG.

Declarations

Conflict of interest The authors declare that they have no conflict of interests.

References

Alouneh S, Abed S, Al Shayeji MH, Mesleh R (2019) A comprehensive study and analysis on sat-solvers: advances, usages and achievements. Artifi Intelli Rev 52(4):2575–2601. https://doi.org/10.1007/s10462-018-9628-0

Bartels JH, Zimmermann J (2009) Scheduling tests in automotive r&d projects. Eur J Operation Res 193(3):805–819. https://doi.org/10.1016/j.ejor.2007.11.010
Basu A, Conforti M, Di Summa M, Jiang H (2021) Complexity of branch-and-bound and cutting planes in mixed-integer optimization - ii. In: Singh M, Williamson DP (eds) Integer Programming and Combinatorial Optimization. Springer, Cham, pp 383–398
Błaewicz J (2007) Handbook on scheduling: From theory to applications. International handbooks on information systems. Springer, Berlin
Brucker P (2007) Scheduling Algorithms, 5th edn. Springer-Verlag GmbH, Berlin Heidelberg, https://doi.org/10.1007/978-3-540-69516-5
Buergin J, Helming S, Andreas J, Blaettchen P, Schweizer Y, Bitte F, Haefner B, Lanza G (2018) Local order scheduling for mixed-model assembly lines in the aircraft manufacturing industry. Product Eng 12(6):759–767. https://doi.org/10.1007/s11740-018-0852-x
Cook SA (1971) The complexity of theorem-proving procedures. In: Harrison MA, Banerji RB, Ullman JD (eds) Proceedings of the third annual ACM symposium on Theory of computing - STOC '71, ACM Press, New York, New York, USA, pp 151–158, https://doi.org/10.1145/800157.805047
Da Col G, Teppan E (2019) Google vs IBM: A constraint solving challenge on the job-shop scheduling problem. Electron Proceed Theoret Comp Sci 306:259–265. https://doi.org/10.4204/eptcs.306.30
Dörmer J, Günther HO, Gujjula R (2015) Master production scheduling and sequencing at mixed-model assembly lines in the automotive industry. Flex Serv Manuf J 27(1):1–29. https://doi.org/10.1007/s10696-013-9173-8
Ebert C, Favaro J (2017) Automotive software. IEEE Software 34(3):33–39. https://doi.org/10.1109/MS.2017.82
ElMaraghy H, Caggiano A (2016) Flexible manufacturing system. In: Produ TIAf, Laperrière L, Reinhart G (eds) CIRP Encyclopedia of Production Engineering, Springer Berlin Heidelberg, Berlin, Heidelberg, pp 1–7, https://doi.org/10.1007/978-3-642-35950-7_6554-4
Hillier FS, Herrmann JW (2006) Handbook of Production Scheduling, vol 89. Springer, US, Boston, MA,. https://doi.org/10.1007/0-387-33117-4
Hu TC, Kahng AB (2016) Linear and Integer Programming Made Easy. Springer International Publishing, Cham,. https://doi.org/10.1007/978-3-319-24001-5
Huang H, Zhou S (2018) An efficient sat algorithm for complex job-shop scheduling. In: Proceedings of the 2018 8th International Conference on Manufacturing Science and Engineering (ICMSE 2018), Atlantis Press, Paris, France, https://doi.org/10.2991/icmse-18.2018.126
ISA (2000) Enterprise-control system integration (isa-95.00.01), models and terminology
Lawlor A (1973) Works Organisation. Macmillan Education UK, London,. https://doi.org/10.1007/978-1-349-01782-9
Marques-Silva J, Sakallah K (1999) Grasp: a search algorithm for propositional satisfiability. IEEE Trans Comput 48(5):506–521. https://doi.org/10.1109/12.769433
Padberg M, Rinaldi G (1991) A branch-and-cut algorithm for the resolution of large-scale symmetric traveling salesman problems. SIAM Rev 33:60–100
Perron L, Furnon V (2019) Or-tools By Google. Version 7:2
Pulina L, Seidl M (2020) Theory and Applications of Satisfiability Testing - SAT 2020, vol 12178. Springer, Cham,. https://doi.org/10.1007/978-3-030-51825-7
Quilliot A, Sarbinowski A, Toussaint H (2021) Vehicle driven approaches for non preemptive vehicle relocation with integrated quality criterion in a vehicle sharing system. Ann Operat Res 298:1–24. https://doi.org/10.1007/s10479-019-03497-4
Shi Y, Reich D, Epelman M, Klampfl E, Cohn A (2017) An analytical approach to prototype vehicle test scheduling. Omega 67:168–176. https://doi.org/10.1016/j.omega.2016.05.003
Shmoys D, Wein J, Williamson D (1991) Scheduling parallel machines on-line. SIAM J comput 24:131–140. https://doi.org/10.1109/SFCS.1991.185361
Spieckermann S, Gutenschwager K, Voß S (2004) A sequential ordering problem in automotive paint shops. Int J Prod Res 42(9):1865–1878. https://doi.org/10.1080/00207540310001646821
Vanderbei RJ (2020) Linear Programming. International Series in Operations Research & Management Science, Springer, Cham,. https://doi.org/10.1007/978-3-030-39415-8
Wang S, Liu M (2015) Multi-objective optimization of parallel machine scheduling integrated with multi-resources preventive maintenance planning. J Manuf Sys 37:182–192. https://doi.org/10.1016/j.jmsy.2015.07.002

Weckenborg C, Kieckhäfer K, Spengler TS, Bernstein P (2020) The volkswagen pre-production center applies operations research to optimize capacity scheduling. INFORMS J Appl Anal 50(2):119–136. https://doi.org/10.1287/inte.2020.1029

Simone König is currently Development Engineer at the research and predevelopment of production technologies at Mercedes-Benz AG with a main focus on data science and business process management. She studied Mathematics and is enrolled in the PhD programme of the TUM School of Engineering and Design at Technical University of Munich.

Maximilian Reihn is currently enrolled in the Master's programme of Mathematical Finance at University Konstanz. During his studies he started as working student to support a R&D team at Mercedes-Benz AG and developed algorithmic code for data science projects. His academic interest focuses on numerical optimisation and optimal control problems.

Felipe Gelinski Abujamra is currently Development Engineer at the research and predevelopment of production technologies at Mercedes-Benz AG, focusing on development concepts for production using process analysis. Felipe studied Business Information Systems at the Stuttgart Technology University of Applied Sciences. At Mercedes-Benz AG, he wrote his graduation thesis in 2019, in which he dealt with automated scheduling in the automotive manufacturing.

Alexander Novy is currently Development Engineer at the Charging Functions R&D Department at Mercedes-Benz AG. His focus is the development of digital services for electric vehicles. Before developing cloud services, Alexander worked on the research and predevelopment of production technologies at Mercedes-Benz with a main focus on business intelligence and data science. Methodologically, he focuses on machine learning methods, artificial intelligence and data modelling.

Birgit Vogel-Heuser is a full professor and director of the Institute of Automation and Information Systems at the Technical University of Munich. Her main research interests are systems engineering, software engineering, and modeling of distributed and reliable embedded systems. She is core member of TUM's MDSI (Munich Data Science Institute), member of TUM's MIRMI (Munich Institute of Robotics and Machine Intelligence), member of the German Academy of Science and Engineering, chair of the VDI/VDE working group on industrial agents and vice chair of the IFAC TC 3.1 computers in control.