Ref. Ares(2024)8514457 - 29/11/2024

EUROPEAN COMMISSION

HORIZON EUROPE PROGRAMME – TOPIC: HORIZON-CL5-2022-D2-01

FASTEST

Fast-track hybrid testing platform for the development of battery systems

Deliverable D6.2: Scheduling Software Solution

Primary Author [Dr. Shuchen Liu]

Organization [FEV]

Date: [29.11.2024]

Doc.Version: [V1.0]

Funded by the European Union

UK Research and Innovation

Co-funded by the European Union and UKRI under grant agreements N° 101103755 and 10078013, respectively. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or the European Climate, Infrastructure and Environment Executive Agency (CINEA). Neither the European Union nor CINEA can be held responsible for them

Document Control Information	
Settings	**Value**
Work package:	6 – Development of hybrid testing platform
Deliverable:	Scheduling software solution
Deliverable Type:	R – Document, report
Dissemination Level:	PU - Public
Due Date:	30.11.2024 (Month 18)
Actual Submission Date:	29.11.2024
Pages:	< 28 >
Doc. Version:	V1.0
GA Number:	101103755
Project Coordinator:	Bruno Rodrigues│ ABEE (bruno.rodrigues@abeegroup.com)

Formal Reviewers		
Name	**Organization**	**Date**
Eliana Giovannitti	COMAU	22.11.2024
Philipp Brendel	FHG	20.11.2024

Document History			
Version	**Date**	**Description**	**Author**
0.9	06.11.2024	First Draft	Dr. Shuchen Liu (FEV) Felix Pischinger (FEV) Bruno Rodrigues (ABEE) Antonio Silvio de Letteriis (Flash Battery) Silvia Delbono (Flash Battery)
1.0	29.11.2024	Submission Version	Dr. Shuchen Liu (FEV) Felix Pischinger (FEV) Bruno Rodrigues (ABEE) Antonio Silvio de Letteriis (Flash Battery) Silvia Delbono (Flash Battery)

Project Abstract

Current methods to evaluate Li-ion batteries safety, performance, reliability and lifetime represent a remarkable resource consumption for the overall battery R&D process. The time or number of tests required, the expensive equipment and a generalized trial-error approach are determining factors, together with a lack of understanding of the complex multiscale and multi-physics phenomena in the battery system. Besides, testing facilities are operated locally, meaning that data management is handled directly in the facility, and that experimentation is done on one test bench.

The FASTEST project aims develop and validate a fast-track testing platform able to deliver a strategy based on Design of Experiments (DoE) and robust testing results, combining multi-scale and multi-physics virtual and physical testing. This will enable an accelerated battery system R&D and more reliable, safer and long-lasting battery system designs. The project's prototype of a fast-track hybrid testing platform aims for a new holistic and interconnected approach. From a global test facility perspective, additional services like smart DoE algorithms, virtualized benches, and DT data are incorporated into the daily facility operation to reach a new level of efficiency.

During the project, FASTEST consortium aims to develop up to TRL 6 the platform and its components: the optimal DoE strategies according to three different use cases (automotive, stationary, and off-road); two different cell chemistries, 3b and 4 solid-state (oxide polymer electrolyte); the development of a complete set of physic-based and data driven models able to substitute physical characterization experiments; and the overarching Digital Twin architecture managing the information flows, and the TRL6 proven and integrated prototype of the hybrid testing platform.

LIST OF ABBREVIATIONS, ACRONYMS AND DEFINITIONS

Acronym	Name
AKS	Azure Kubernetes Services
AI	Artificial intelligence
DT	Digital Twin
UUT	Unit Under Test
Dx.x	Deliverable within the FASTEST Context
B&B	Branch and Bound Algorithm
SQL	Structured Query Language
DB	Database
HTTP	Hypertext Transfer Protocol
DoE	Design of Experiments
MIP	Mixed-Integer Programming
ACR	Azure Container Services
POC	Proof Of Concept
LIMS	Laboratory Inventory Management System
MQTT	Message Queuing Telemetry Transport

LIST OF TABLES

LIST OF FIGURES

Table of Contents

1. EXECUTIVE SUMMARY

The objective of Deliverable D6.2 in the FASTEST project was to develop and implement a scheduling software solution to optimize battery testing processes. This report documents the design and deployment of a task scheduling algorithm integrated within the Laboratory Information Management System (LIMS). The primary objective of the scheduling algorithm was to ensure efficient allocation of resources across both virtual and physical test benches, reduce idle time, and minimize delays, thereby enhancing the productivity and cost-effectiveness.

The scheduling algorithm, based on the sequence-based Compact Model, advances beyond traditional methods by leveraging the solver's multithreading capabilities and custom branching heuristics. This allowed for the parallelization of tasks and improved prioritization of high-priority tests. Notably, these optimizations reduce the computational complexity typically associated with Branch and Bound (B&B) algorithms in scheduling contexts, achieving faster processing times and higher-quality solutions.

The key progress achieved includes the algorithm's successful deployment, validated through designed test cases that demonstrated significant reductions in both delay of testing and resource vacancies. The system was further extended to handle complex scheduling requirements by introducing constraints for resource limitations, prioritization, and unexpected events. This comprehensive integration of the scheduling algorithm into the LIMS workflow marks a substantial improvement over conventional approaches, setting a new standard for task scheduling efficiency in battery testing environments.

2. OBJECTIVES

The development of the task scheduling algorithm was a crucial step in the FASTEST project, as it aimed to optimize the utilization of testing resources and minimize the time and cost associated with battery testing.

Defining the Algorithm's Objectives

The first step was to clearly define the objectives of the scheduling algorithm. In the context of the FASTEST project, the objectives of the task scheduling algorithm are as follows:

- Efficiently allocate testing resources: This included physical test benches, simulation resources. The algorithm was designed to assign the right resources to the right tasks at the right time.
- Optimize testing schedules: The algorithm was designed to create efficient testing schedules that minimized the overall testing time and cost. This included considering factors such as resource availability, and deadlines.
- Improve test centre efficiency: The algorithm aimed to contribute to the overall efficiency of the test centre by maximizing resource utilization and reducing idle time.

3. INTRODUCTION

Deploying the mathematical model involved integrating the task scheduling algorithm into the testing process, ensuring seamless operation with both virtual and physical environments. This required incorporating the algorithm into the test centre's data model, which contained essential information about test demands, configurations, and overall planning. By accessing this data, the algorithm could make informed scheduling decisions.

The algorithm interacted with both virtual test benches—simulation environments for virtual tests—and physical test benches used for actual batteries or components. It scheduled tests efficiently by considering resource availability. Effective data exchange and communication protocols were established to ensure the algorithm received necessary information and could send scheduling instructions appropriately. In some cases, real-time adjustments were possible, allowing the algorithm to update scheduling decisions based on the status of test benches and resource availability. After deployment, performance metrics such as resource utilization, testing time, and cost were monitored to evaluate and refine the algorithm's effectiveness.

Integrating the solution into the Laboratory Information Management System (LIMS) was the final step. The LIMS was connected to the Digital Twin (DT) system to access comprehensive information about the Unit Under Test (UUT), including lifecycle data, descriptive details, existing models, and prior test data. The scheduling algorithm was seamlessly incorporated into the LIMS workflow, managing test demands, resource allocation, and scheduling within a unified platform. Effective communication protocols ensured the LIMS could retrieve necessary information from the DT system to optimize scheduling decisions. A user-friendly interface allowed lab personnel to interact with the algorithm, visualize the testing schedule, and track test progress. The "as fast as possible" data exchange between the LIMS and the DT system ensured that any changes in the UUT's status or resource availability were immediately reflected in scheduling decisions. This integration operationalized the optimized scheduling capabilities developed in the project, achieving a more efficient and cost-effective battery testing process.

The remaining content of this document is structured as following:

In the rest of the Chapter 3, further background information is provided. This information is required to understand the further approach in Chapter 4. For a more detailed understanding of this information, it is recommended to take a deeper look into D6.1, where the foundations for this task have been developed.

Chapter 4 covers the development of the executable scheduling algorithm. This includes the development of the mathematical model and the implementation with suitable tools and software.

3.1 Retrospect on Theoretical Foundation and Concept of Resource Scheduling

Selecting Appropriate Optimization Techniques

Once the objectives were defined, the next step was to select the appropriate optimization techniques. Some possible optimization techniques that were considered for the FASTEST project included:

- Linear programming: This technique is used to optimize a linear objective function, subject to linear equality and inequality constraints.
- Integer programming: This technique is used when some or all the variables are restricted to be integers.
- Constraint programming: This technique is used to find solutions that satisfy a set of constraints.
- Heuristic algorithms: These algorithms are used to find good solutions to optimization problems in a reasonable amount of time, even if they are not guaranteed to be optimal.
- Artificial intelligence (AI) methods: AI methods, such as machine learning, can be used to learn from historical data and make predictions about future testing needs.

The choice of optimization technique depended on the specific characteristics of the scheduling problem, such as the number of tasks, the number of resources, and the complexity of the constraints.

As outcome of the detailed analysis in D6.1, it was decided to formulate the scheduling problem as mixed-integer programming (MIP) problem which will be solved using the Branch and Bound (B&B) algorithm.

3.2 Key Optimization Strategies

For a specific task scheduling problem, Wang's (Wang, 2018) sequence-based Compact Model is utilized. The goal is to reduce computational complexity and improve solution quality by aligning with the model's decision variables and objectives, which will be introduced in next chapter in detail and summarized as follows:

- Efficient Use of Decision Variables: By focusing on sequence-related binary variables that represent task precedence, the number of constraints and branching nodes is reduced. Simplifying time-related variables to only starting and completion times lessens computational burden.
- Effective Branching Strategies: Prioritizing branching on sequence-related variables directly addresses ordering constraints, leading to quicker pruning of infeasible sequences. Heuristics that prioritize tasks with tight deadlines or high delay penalties help find feasible solutions more rapidly.

- Objective Function Optimization: Introducing techniques like linearization or lexicographic optimization balances conflicting objectives such as minimizing vacancies and delays. Adjusting weighting parameters focuses the algorithm on subproblems that significantly impact the objective function.
- Parallelization with Optimization Solver: Leveraging the solver's multithreading capabilities allows for parallel exploration of branches, accelerating convergence. Custom branching strategies that prioritize critical tasks and resources improve solution quality.
- Post-Solution Analysis: Using Gantt charts for visual feedback helps identify inefficiencies like idle times, enabling further refinement. Implementing adaptive gap tolerances allows for early termination when acceptable solution quality is reached, which is beneficial for large instances.

3.3 Gap Analysis from Theory to Practice

Implementing the Algorithm in a Prototype Software Solution

The final step was to implement the algorithm in a prototype software solution.

In the FASTEST project, the scheduling algorithm was implemented in a software solution that could be integrated with the test centre's data model and the LIMS. This allowed the algorithm to access the necessary information about test demands, configurations, and planning, as well as the status of virtual and physical test benches.

The prototype software solution was tested and validated using designed use cases to ensure it met the objectives of the scheduling algorithm. This involved evaluating the algorithm's performance in terms of efficiency, scalability, and accuracy. The development of the task scheduling algorithm was an iterative process that involved continuous refinement and improvement. The prototype software solution was regularly updated and enhanced based on feedback from the project partners and the results of testing and validation.

4. Development of Task Scheduling Algorithm

In the previous delivery D6.1 the B&B algorithm was selected as a solution approach. In this chapter, it is applied to the planning problem of identically parallel machines. The first chapter introduces the tools that are used to solve the problem. Subsequently, the general procedure is explained and the modelling of the planning situation in a mixed integer program is derived step by step.

4.1 Implementation of the solution approach

In this chapter, a mathematical description is given in the form of a mixed-integer program. The latter then represents the basis on which the B&B algorithm is applied. For the B&B algorithm, the optimization software Gurobi is used (Gurobi Optimizer LLC, Reference Manual, 2024). Gurobi is, among other things, a framework for solving mixed-integer programs. In the following, Gurobi is introduced in more detail.

Gurobi is a solver designed for handling numerical programming tasks. Beside the B&B algorithm it consists of further procedures for the reduction of the complexity and acceleration of the solution finding. These procedures are:

- The *Pre-solve process* checks in the first step whether restrictions of the modelling can be narrowed so that constraints coincide. This reduces the complexity of the problem and speeds up algebraic operations.
- With the help of a relaxation, the mixed integer program is converted into a purely linear program. This means that the integer condition is removed for all variables. Optimal admissible solutions are then calculated for the relaxed program. These can then serve as bounds for the mixed integer program.
- The cutting plane method is applied to a relaxed version of the program. During the solution process, additional constraints are introduced near admissible solutions to further restrict the solution space. In the context of the Gurobi solver, Gomory mixed integer, mixed integer rounding, flow-over and lift-and-project cuts, among others, are applied.
- In addition to branching the solution tree in the context of the B&B algorithm help heuristics to improve the quality of the solution. On the one hand, via Start Heuristics generate new admissible solutions and on the other hand Improvement Heuristics improve found solutions. In the context of Start Heuristics, Rounding Heuristics, Feasibility Pumps and Fix-and-dive Heuristics are applied and in the context of Improvement Heuristics the methods Local Branching, Crossover and 1-Opt and 2-Opt are used.

The pre-solve process is run once before the algorithm is applied. The algorithm then iterates through the next steps until the optimal solution is found or a termination criterion is reached. The procedure of the algorithm is shown in the right part of Figure 1. In the first step, a node is selected in the solution tree. Then, solutions for the subproblem are determined for this node. Based on the first solution, the relaxation of the problem starts. The cutting plane method is then applied to the relaxed problem before heuristics are used to determine further admissible and improved solutions in the penultimate step. Finally, the solution tree is re-branched and thus extended by new nodes, whereupon a new iteration step starts, and the new nodes are selected and solved.

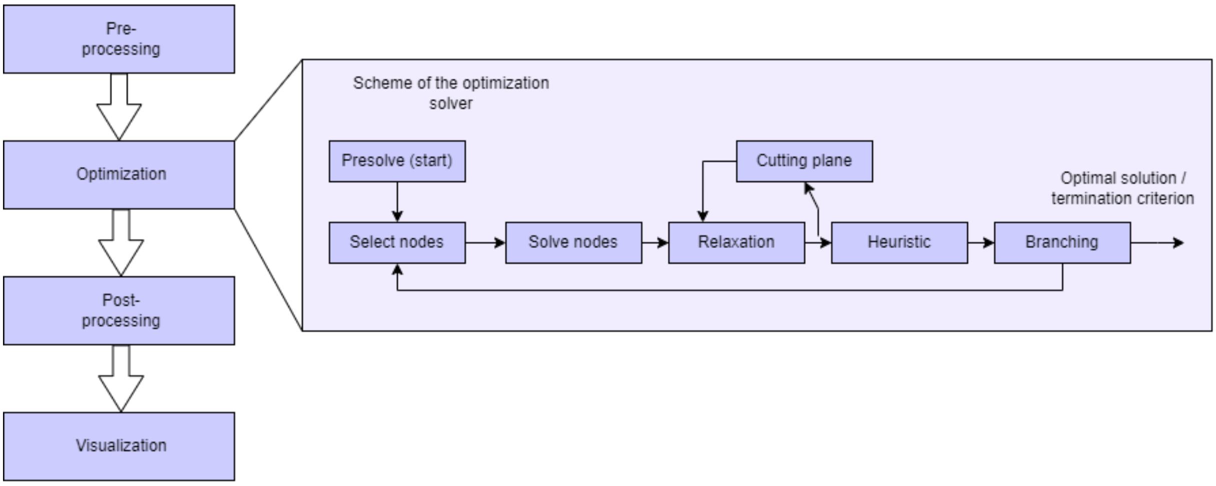

Figure 1. Scheme of the optimization solver (Gurobi Optimization LLC, MIP, 2024)

In this work, the modelling is implemented using the Gurobi framework in the Python programming language. For the solver to solve the planning problem the information about orders and test stands must be prepared. This is done in the *preprocessing* step. For this purpose, orders and test benches are to be queried from the information management system and stored in data structures in Python. Orders and test bench information are fetched from an SQL database. During pre-processing, the order pool is defined for each order by comparing the test bench and order properties. Once all the information has been transferred to the solver, optimization is performed using the scheme described in the previous paragraph. After the optimization follows the step Postprocessing. Here the information about the solution is read out from the solver. This includes reading out the start and end times of the orders, the assignment to which machine which order is carried out with which resource and the number of delays and vacancies. In the last step, this information is visualized in a Gantt chart. The optimization procedure is illustrated in Figure 1.

4.2 Deploying the mathematical model

In this section, the planning situation of a battery test field is to be abstracted in a linear program. A linear program consists of an objective function, constraints, variables and parameters. A linear program is basically of the following form:

Objective function	minimize	$\sum_{a \in A} c_a \cdot x_a$
Constraints	subject to	$\sum_{a \in A} k_{p,a} \cdot x_a = h_p, \forall p \in P$

$$l_a \leq x_a \leq u_a, \forall a \in A$$

Variables	x_a :	Decision variables
	$c_a,\ k_{p,a},\ h_p,\ l_a,\ u_a$:	Parameters
	a, p:	Indices
	A, P:	Number of jobs and test benches

With the help of the linear program, all planning-relevant information is transferred into linear relationships. First, the decision variables are defined. This is followed by the definition of the objective function and restrictions. A planning problem can be modelled in different ways. The choice of the decision variables has a decisive influence on the form of the modelling and the later runtime behaviour of the algorithm. The difficulty is to choose a modelling that meets all planning requirements and works efficiently at the same time.

4.2.1 Subdividing the planning problem

In addition to the choice of decision variables, the size of the model also has a significant impact on the runtime of the algorithm. To address this, it may be useful to break down the planning situation into subproblems, reducing the complexity of the planning problem. However, it is known from D6.1. that dividing a planning problem into subproblems can reduce overall optimality. Therefore, it is essential to weigh whether the advantages of a subdivision outweigh the disadvantages. Basically, the more independent the subproblems are from each other, the better a subdivision is possible. Conversely, if there are interactions between the subproblems, synergies cannot be exploited. If two problems are addressed separately, resources and information cannot be shared, and in the case of an optimization problem, the objective variables for the two subproblems are optimized independently. This independence is not a problem if there are no dependencies between the two objective variables. However, if dependencies exist, it can reduce the optimality of the global solution.

In the context of battery test field, dividing the planning problem into the three domains of endurance, environment and misuse is a potential approach. As noted in D6.1 these three domains differ in many ways. A separate consideration of the three domains would be feasible if the interactions between the domains are small. That is, the three areas are likely to share orders or resources of the same type only to a minor extent. The first step is to evaluate the extent to which the areas share the same orders. For the abuse area, eligible orders can be carried out exclusively within that domain, and orders that are eligible for the other two areas cannot be processed there. Thus, the Abuse area does not share any orders with the Environment and Endurance areas. However, the same cannot be said for orders in the endurance and environmental areas. The environmental area includes a small number of temperature and climate chambers for preconditioning of test specimens. Technically, these are the same chambers as they also exist in the endurance range. It would therefore be possible, in the event of overload of the chambers in the environmental area to chambers in the continuous operation area and vice versa. However, the proportion of such shared orders is so small that this case does not need to be automated in the planning. Resource sharing present a more significant challenge. For the complete test field, three types of resources are considered: technical devices, energetic capacities and personnel capacities. While most technical devices are specific to their areas, energetic and personnel capacities are shared across domains. Treating these areas separately, would require a separate set of energetic and personnel capacities for each area. If there were unused capacities in one area, these could then not be assigned to another area within the framework of automated planning. Finally, it cannot be judged exactly whether the subdivision of the planning situation justifies the loss of optimality. To avoid giving the definite answer to this question now and to retain flexibility for future adjustments, a modelling with components is used.

A base modelling is developed that meets the basic requirements of all three domains. Any further domain-specific requirements are handled in the form of additional components that can be integrated into the base model. The base model can thus be applied either to individual areas or to the complete test field. In addition, this offers the possibility of expanding or restricting the individual areas with components at a later point in time.

4.2.2 Choice of decision variables

While modelling, the first step is to define the decision variable. In principle, two different formulations are possible, which differ in terms of their indexation. *Seelbach* (Seelbach, 1975) distinguishes between time-related and sequence-related decision variables. In the context of time-related decision variables, it is determined for each test stand and each order at a time unit whether an assignment is made or not. An example is the binary variable $x_{a,p,t}$, which takes the value 1 if the order a is carried out on the test bench p at the time unit t. A sequence-related decision variable, on the other hand, specifies the relative sequence of two orders.

A binary variable of the form $y_{a1,\ a2}$ can be used to specify whether the order a_1 before the Order a_2 is executed ($y_{a1,\ a2}$ = 1) or not ($y_{a1,\ a2}$ = 0). For sequence-related decision variables, the temporal structure must also be described using other variables. In addition to time- and sequence-related decision variables, a distinction is made between binary decision variables ($x \in \{0, 1\}$) and integer decision variables ($x \in$ Z). A modelling via exclusively binary variables usually leads to a high number of variables. Binary variables can only take one of two values, whereas integer variables can take any value from Z. This makes it difficult for the solver to handle integer variables. For the solver, this complicates the handling of integer variables. Ultimately, which choice of variables is appropriate depends on the specific problem. *Wang* (Wang, 2018) evaluated five different modelling approaches for a scheduling problem of identically parallel machines with precedents as part of his dissertation. Four of the modelling approaches use sequence-related decision variables and one modelling approach uses time-related decision variables. All five approaches are based on a discrete time understanding.

For this purpose, the time horizon is divided into time slots as shown in Figure 2. Each time slot is defined starting from the now time $t = 0$. Starting from $t = 0$, the job a_1 starts in $t = s_{a1}$ and ends in $t = c_{a1}$.

The test results from Wang (Wang, 2018) show that sequence-based modelling is characterized by significantly faster processing times. The best result is achieved by what he calls the "Compact Model", which is based on modelling with sequence-related decision variables and, through clever indexing, can be implemented with fewer restrictions.

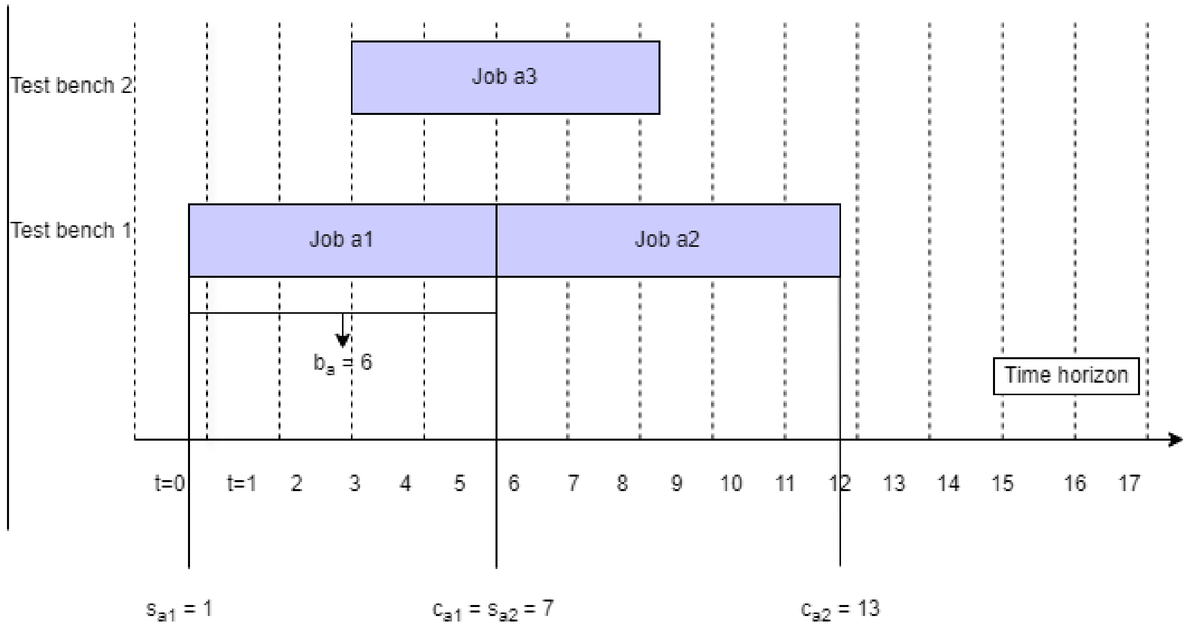

Figure 2. Representation of an order assignment

In the following, the decision variables of the Compact Model are therefore adopted. In connection with this, a discrete time understanding is also chosen for the modelling of the basic model. It should be noted that a time slot stands for a period that can be defined arbitrarily. Depending on the situation, a time slot can be an hour, a day or the length of a shift. Within the model, however, all time slots have the same length.

The following table summarizes all notation agreements made for the base model.

Table 1 List of Notations

Indexes	a	Test order $a \in A$
	p	Test bench $p \in P$
Parameters	A	Quantity of all test orders
	P	Quantity of all test benches

	r_a	Release time of order $a \in A$, $r_a \in N_0$
	b_a	Processing time of order $a \in A$, $b_a \in N_0$
	d_a	Deadline of order $a \in A$, $d_a \in N_0$
	g_L	Weighting of the sum of all vacancies Σ_L , $g_L \in [0, 1]$
	g_V	Weighting of the sum of all delays Σ_V , $g_V \in [0, 1]$.
	M	Sufficiently large number
Supporting variables	c_a	Completion time of job $a \in A$, $c_a \in N_0$
	v_a	Delay or missed deadline of order $a \in A$, $v_a \in N_0$
	e_p	Completion time of the last order on the test bench $p \in P$, $e_p \in N_0$
	l_p	Number of vacancies on test bench $p \in P$, $l_p \in N_0$
	Σ_B	Sum of the processing times of all orders of a test bench $p \in P$, $\Sigma_B \in N_0$
Decision variables	s_a	Start time of job $a \in A$, $s_a \in N_0$
	$x_{a,p}$	$:= \begin{cases} 1, & \text{if job a} \in \text{A is processed on test bench p} \in \text{P} \\ 0, & \text{otherwise} \end{cases}$
	$y_{a1,a2}$	$:= \begin{cases} 1, & \text{if order } a_1 \in \text{A is executed before order } a_2 \in \text{A} \\ 0, & \text{otherwise} \end{cases}$
Objectives	Z	The total objective function value, $Z \in R^+$
	Σ_L	Sum of all vacancies on all test beds, $\Sigma_L \in N_0$
	Σ_V	Sum of all delays for all orders, $\Sigma_V \in N_0$

4.2.3 Definition of the base model

In the following, a model is derived from the requirements defined in D6.1. The modelling consists of two parts. The first part is the definition of the base model. The base model covers the requirements (1)-(6) from D6.1. In the second part, the model is extended to include the boundary conditions that are not yet covered. For the base model, the following auxiliary variables are introduced with the following definition:

$$
\begin{aligned}
c_a &= s_a + b_a, && \forall a \in A \\
v_a &= \begin{cases} c_a - d_a, & \text{for } c_a - d_a \geq 0 \\ 0, & \text{otherwise} \end{cases}, && \forall a \in A \\
l_p &= e_p - \Sigma_B, && \forall p \in P \\
\Sigma_B &= \sum_{a \in A} x_{a,p} \cdot b_a, && \forall p \in P \\
\Sigma_L &= \sum_{p \in P} l_p \\
\Sigma_V &= \sum_{a \in A} v_a
\end{aligned}
$$

A description of the auxiliary variables follows: The completion time c_a is the sum of the start time s_a and the processing time b_a of an order *a.* The delay v_a is determined from the difference between the deadline min d_a and the actual completion date c_a. The definition is chosen in such a way that exceeding the deadline takes on positive values. If an order is completed before its deadline date, the delay is set to zero. The amount of vacancy of a test stand l_p is defined by the time when the last job on a test stand is completed, e_p minus the sum of all processing durations of the same test stand. Σ_B is the sum of all processing durations on a test bench, equation. Σ_L is the sum of all idle times on a test stand. Σ_V is the sum of all delays.

The definition of the delay uses a case distinction. This case distinction is translated into linear equations with the help of two binary variables w_a and $\overline{w}_a$. For this purpose, w_a and $\overline{w}_a$ are defined as follows:

$$
\begin{aligned}
w_a &= \begin{cases} 1, & \text{for } c_a - d_a > 0 \\ 0, & \text{otherwise} \end{cases}, && \forall a \in A \\
\overline{w}_a &= \begin{cases} 1, & \text{for } c_a - d_a \leq 0 \\ 0, & \text{otherwise} \end{cases}, && \forall a \in A
\end{aligned}
$$

With the help of w_a it can be guaranteed that $v_a = c_a - d_a$ only holds if $c_a - d_a > 0$. For this purpose, following equations are added to the model.

$$
\begin{aligned}
v_a &= (c_a - d_a) \cdot w_a, && \forall a \in A \\
c_a - d_a &\leq M \cdot w_a, && \forall a \in A \\
d_a - c_a &< M \cdot \overline{w}_a, && \forall a \in A \\
w_a + \overline{w}_a &= 1, && \forall a \in A
\end{aligned}
$$

The second and third equation above guarantee that w_a and $\overline{w}_a$ take the correct values depending on c_a and d_a. The last equation guarantees that only one of the two binary variables is active for an order. Here, *M* is a sufficiently large number to ensure that the irrelevant inequality does not further restrict the solution space. This approach is known as the "Big M" method.

4.2.4 Definition of the objective function

For optimization models with multiple objectives, Gurobi (Gurobi Optimizer LLC, Reference Manual, 2024) offers an approach to manage and configure those:

"The main challenge you face when working with multiple, competing objectives is deciding how to manage the trade-offs between them. Gurobi provides tools that simplify the task: Gurobi allows you to blend multiple objectives, to treat them hierarchically, or to combine the two approaches. In a blended approach, you optimize a weighted combination of the individual objectives. In a hierarchical or lexicographic approach, you set a priority for each objective and optimize in priority order. When optimizing for one objective, you only consider solutions that would not degrade the objective values of higher-priority objectives. Gurobi allows you to enter and manage your objectives, to provide weights for a blended approach, and to set priorities for a hierarchical approach."

Within the framework of the basic model, the number of all vacancies and the deadline overruns are considered in the objective function in accordance with the planning requirements (5) and (6) from D6.1. It would also be possible to consider schedule overruns in the form of constraints. However, this does not make sense for the planning situation, since one unavoidable delay would make the model unsolvable in this case.

The number of vacancies is captured by the auxiliary variable Σ_L. For the sum of delays, the variable Σ_V was introduced. Both variables are minimized. To link the two target quantities, the multi objective functionality of Gurobi can be used ((Gurobi Optimizer LLC, Reference Manual, 2024)– *Multiple Objects).* This functionality allows to set several configurations such as prioritization for each objective:

Model.setObjectiveN()

setObjectiveN (expr, index, priority=0, weight=1, abstol=1e-6, reltol=0, name="")

Set an alternative optimization objective equal to a linear expression.

Arguments:

expr (LinExpr): New alternative objective.

index (int): Index for new objective. If you use an index of 0, this routine will change the primary optimization objective.

priority (int, optional): Priority for the alternative objective. This initializes the ObjNPriority attribute for this objective.

weight (float, optional): Weight for the alternative objective. This initializes the ObjNWeight attribute for this objective.

abstol (float, optional): Absolute tolerance for the alternative objective. This initializes the ObjNAbsTol attribute for this objective.

reltol (float, optional): Relative tolerance for the alternative objective. This initializes the ObjNRelTol attribute for this objective.

name (string, optional): Name of the alternative objective. This initializes the ObjNName attribute for this objective.

Thus, the objective functions results in:

```
Model.setObjectiveN(vacancies, index=1, priority=1, reltol=0.0, name='Vacancy')
Model.setObjectiveN(delays,    index=0, priority=2, reltol=0.2, name='Delay')
```

Setting the index of the delays to 0 will make the minimization of the delays the primary objective for the optimization. The "reltol" will allow to exceed the optimal delay solution to optimize the second objective vacancies.

4.2.5 Definition of boundary conditions

For two consecutive jobs, the next job can only start when the previous job is finished. In the following, the one-machine case is considered. For two jobs a_1 and a_2, two situations can occur in this case. Either job a_1 is executed before job a_2, also $a_1 \prec a_2$ is written, or vice versa, $a_1 \succ a_2$. For both cases, a sequence condition can be defined. Therefore holds:

$$s_{a_2} \geq s_{a_1} + b_{a_1} \lor s_{a_1} \geq s_{a_2} + b_{a_2}, \forall a_1, a_2 \in A, a_1 \neq a_2$$

This condition must be transformed into a linear inequality. For this purpose, the binary variable $y_{a1,\, a2}$ is introduced as defined in Table 1 in the section *Decision variables.* The binary variable can be used to formulate the following two linear inequalities.

$$s_{a_2} \geq s_{a_1} + b_{a_1} - M \cdot \left(1 - y_{a_1,a_2}\right), \forall a_1, a_2 \in A, a_1 \neq a_2$$
$$s_{a_1} \geq s_{a_2} + b_{a_2} - M \cdot \left(y_{a_1,a_2}\right), \forall a_1, a_2 \in A, a_1 \neq a_2$$

The binary variable ensures that only one of the two inequalities is considered. *M* is again a sufficiently large number. For the transition from the single-machine case to the multiple-machine case, the binary variable $x_{a,p}$ is additionally introduced, which assumes 1, when job *a* is performed on test bench *p.* A definition of the variable $x_{a,p}$ is given in Table 1. To complete the definition of the base model, two more equations are added to the modelling. These Equations ensure that the start time of an order is after its release time. This results in the basic model as follows:

$$\begin{array}{ll}
\mathbf{min} & Z \\
\mathbf{s.t.} & s_{a_2} \geq s_{a_1} + b_{a_1} - M \cdot \left(1 - y_{a_1,a_2} + 1 - x_{a_1,p} + 1 - x_{a_2,p}\right), \quad \forall a_1, a_2 \in A, \forall p \in P \\
 & s_{a_1} \geq s_{a_2} + b_{a_2} - M \cdot \left(y_{a_1,a_2} + 1 - x_{a_1,p} + 1 - x_{a_2,p}\right), \quad \forall a_1, a_2 \in A, \forall p \in P \\
 & \sum_{p \in P} x_{a,p} = 1, \forall a \in A \\
 & s_a \geq r_a, \forall a \in A
\end{array}$$

The binary variable $y_{a1,\, a2}$ is, in the context of the basic model, on the one hand a decision variable, which is set by the solver when the order of two orders is specified. However, $y_{a1,\, a2}$ can additionally be used to implement precedents according to requirement (4) from D6.1. In this case, when building the model, the variable $y_{a1,\, a2}$ is set equal to 1 for two concrete orders a_1 and a_2, provided that order a_1 is to be performed before order a_2. In this case, $y_{a1,\, a2}$ is no longer a decision variable, but a parameter. In general, this means that for a set Q of tuples (a_1, a_2) with $a_1 \prec a_2$ three equations can be defined, so that the desired precedence relations are ensured:

$$\begin{aligned} s_{a_2} &\geq s_{a_1} + b_{a_1}, && \forall(a_1, a_2) \in Q \\ y_{a_1,a_2} &= 1, && \forall(a_1, a_2) \in Q \\ y_{a_1,a_2} + y_{a_2,a_1} &= 1, && \forall(a_1, a_2) \in Q \end{aligned}$$

This point concludes the modelling of the basic model. Thus, the requirements (1) - (6) from D6.1 have been successfully transferred into a mixed-integer program. All previous equations are also fully translated into program code and incorporated into the syntax of Gurobi.

4.3 Integration of the solution approach into LIMS

Since LIMS is already using Azure services, the decision was made to deploy the algorithm as Azure Function (MS Azure, 2024) and take advantage of the integration of the different services.

All information related to LIMS are stored in an Azure SQL Database. Any information needed for the algorithm is fetched directly from the Database. After the algorithm has calculated the solution, the database will be updated with relevant information.

The algorithm is triggered by an http-trigger from LIMS whenever relevant data has been added from a user. The following chart illustrates the interaction of LIMS and the algorithm:

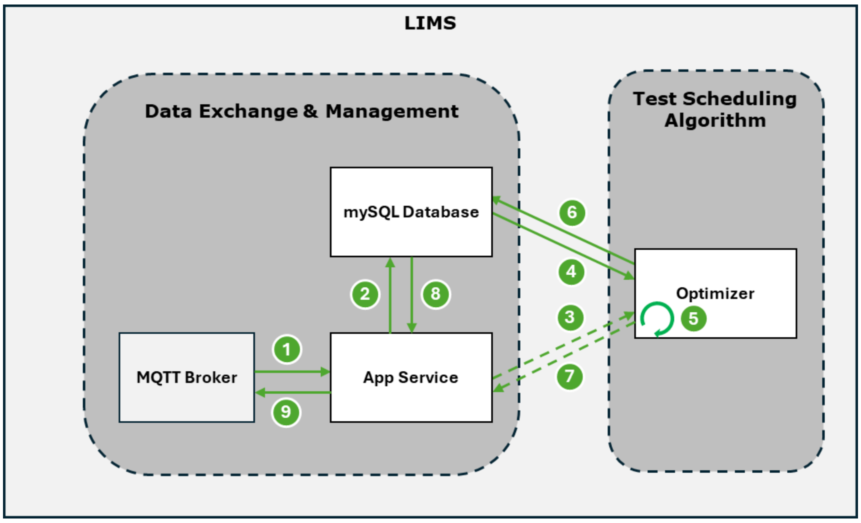

Figure 3. Concept of integrating the test scheduling algorithm into LIMS

Notably, this concept is part of the FEV Azure infrastructure, which is explained in D6.5 in detail and interacts with other FASTEST components with help of LIMS data exchange and management. How the test scheduling algorithm interacts with other components within LIMS is summarized as following table.

Table 2 Description of actions and activities

#	**Type**	**Action**
1	Data transfer	MQTT broker forwards the test request with preferred timing from other FASTEST components
2	Data transfer	App service saves the data and update the SQL database
3	Trigger	App service informs the scheduling algorithm that the database is updated, and new test(s) shall be scheduled
4	Data transfer	The scheduling algorithm fetches the latest records from the database
5	Execution	The scheduling algorithm executes a new optimization
6	Data transfer	The scheduling algorithm fetches the scheduling result and update the SQL database
7	Trigger	The scheduling algorithm informs App service about the updated database
8	Data transfer	App service fetches the data from SQL database
9	Data transfer	App service forwards the data to the MQTT broker and distribute the test to virtual and physical test benches

5. Results

This chapter describes specific tests that are designed to verify the main functionality of the implantation. This involves testing the proper integration of the scheduling algorithm into the LIMS environment and the verification of the scheduling algorithm. Objectives, descriptions and further metadata is listed in a table at the start of each test. This is followed by a description of the test results and an evaluation of the test.

5.1 Connection to LIMS Data Management to Scheduling Algorithm

5.1.1 Test HTTP Trigger

Partners	**System Components**	**Date, Place**	**Status**
FEV	LIMS	07.10.2024. Online (Azure)	Passed
Objective of the Test			
• Test the deployment and the communication to the azure function via the http trigger			
Test Description			
• Check the response of http trigger within Azure Portal directly from the Azure Function Test/Run Menu (Action #3 and #7)			
Open Issues			
• None			

Results and Evaluation

After triggering the function, the response shows that the trigger is responding and the function is sending a response message.

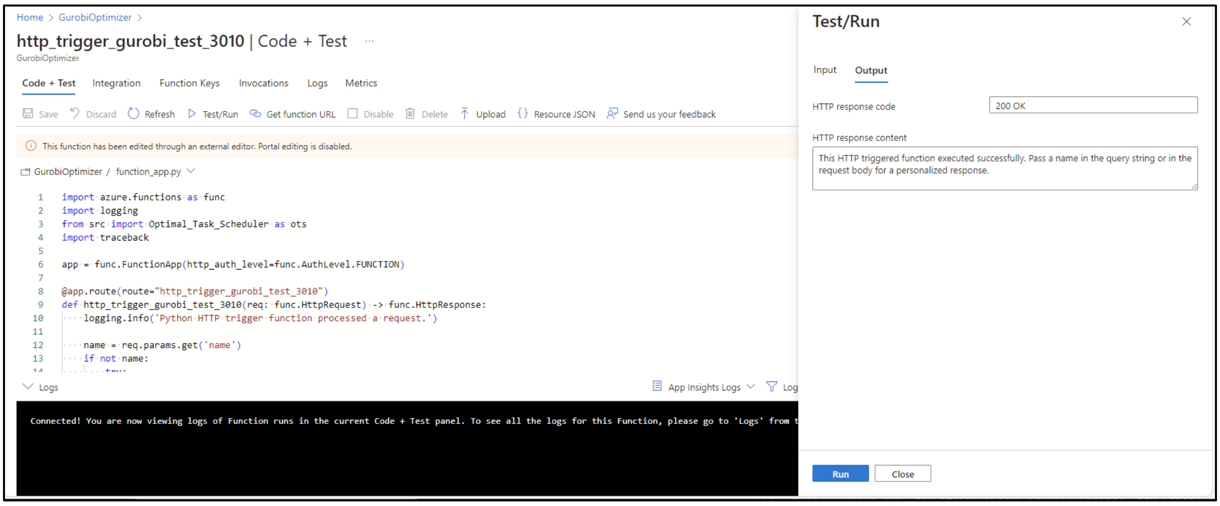

Figure 4. Azure Function HTTP Trigger Test

The response message and successful execution can be verified in Figure 4 and Figure 5.

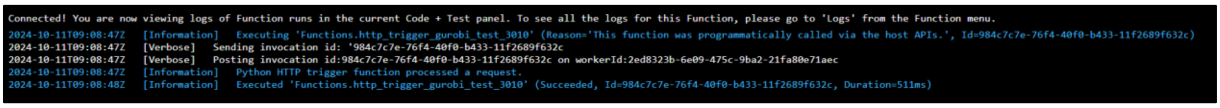

Figure 5. Azure Function HTTP Trigger Test - Log

5.1.2 Test SQL database commit

Partners	**System Components**	**Date, Place**	**Status**
FEV	LIMS	07.10.2024. Online (Azure)	Passed
Objective of the Test			
• Ensure that the output of the algorithm is committed to the database (Action #4 and #6)			
Test Description			
• Add test data to the database and check the output in the database after running the algorithm			
Open Issues			
• None			

Results and Evaluation

The algorithm fetches the relevant information from the database. The information is then processed in the optimization algorithm and the optimized data is returned.

The results are committed to a related table "t_optimized_test_schedule".

Results | Messages

	Id	TestId	dt_user_preferenced_virtual_schedule	dt_optimized_virtual_schedule	dt_user_preferenced_physical_schedule	dt_optimized_physical_schedule	dt_date_removed	dt_date_updated	related_to_test
1	18	9	2024-10-09 14:30:45.0000000	2024-10-10 16:33:39.0000000	NULL	NULL	NULL	2024-10-10 14:33:44.8908080	voltage_draining_01
2	19	10	2024-10-09 14:30:45.0000000	2024-10-12 03:40:39.0000000	NULL	NULL	NULL	2024-10-10 14:33:44.8908080	temperature_shock
3	20	11	2024-10-01 18:13:00.0000000	2024-10-10 16:33:39.0000000	NULL	NULL	NULL	2024-10-10 14:33:44.8908080	electronic_isolation_01
4	21	12	2024-10-14 14:30:45.0000000	2024-10-14 15:49:39.0000000	NULL	NULL	NULL	2024-10-10 14:33:44.8908080	voltage_draining_01
5	22	13	2024-10-03 11:51:00.0000000	2024-10-10 16:33:39.0000000	NULL	NULL	NULL	2024-10-10 14:33:44.8908080	voltage_draining_01
6	23	16	2024-10-11 14:30:45.0000000	2024-10-11 19:38:39.0000000	NULL	NULL	NULL	2024-10-10 14:33:44.8908080	test1
7	24	17	2024-10-12 14:30:45.0000000	2024-10-12 18:29:39.0000000	NULL	NULL	NULL	2024-10-10 14:33:44.8908080	test1
8	25	18	2024-10-14 14:30:45.0000000	2024-10-14 14:31:39.0000000	NULL	NULL	NULL	2024-10-10 14:33:44.8908080	test2

Figure 6. Updated database - "t_optimized_test_schedule"

This table Figure 6 contains the column "dt_optimized_virtual_schedule" which differs from the prescheduled input date. This is the suggested optimized start date which considers the available testbenches and distributes the pending tests among them. The updated table indicates that new data was successfully committed to the database.

5.2 Verification of the Scheduling Algorithm

5.2.1 Test Description

Partners	**System Components**	**Date, Place**	**Status**
FEV	LIMS	31.10.2024. Online (Azure)	Passed
Objective of the Test			
Test if the output of the algorithm is valid (Action #5) • Verification criteria: 1. No overlaying occupation of test benches 2. Minimum delay 3. Minimum vacancy			
Test Description			
Predefined and generated data is added to the database manually. After that the Algorithm is executed and the results are analysed towards the verification criteria.			
Open Issues			
• None			

5.2.2 Test Results and Evaluation

In the following part of the database, the relevant information for the algorithm is shown. This example data serves as input for the algorithm.

Id	s_test_name	s_test_type	s_test_bench	dt_prescheduled_test_date	status	BatteryID	s_prescheduled_test_duration	dt_due_date
2	voltage_draining_01	electric	virtual	31.10.2024 11:39	pending	1	76	31.10.2024 21:01
8	voltage_injection_01	electric	virtual	31.10.2024 11:39	pending	4	25	31.10.2024 19:08
9	voltage_draining_01	electric	virtual	31.10.2024 11:39	pending	3	35	31.10.2024 15:02
10	temperature_shock	temperature	virtual	31.10.2024 11:39	pending	4	28	31.10.2024 17:12
11	electronic_isolation_01	electronic	virtual	31.10.2024 11:39	pending	3	40	31.10.2024 15:09
12	voltage_draining_01	electric	virtual	31.10.2024 11:39	pending	1	73	31.10.2024 19:03
13	voltage_draining_01	electric	virtual	31.10.2024 11:39	pending	3	46	31.10.2024 16:38
16	test1	electric	virtual	31.10.2024 11:39	pending	1	33	31.10.2024 18:49
17	test1	electric	virtual	31.10.2024 11:39	pending	1	25	31.10.2024 16:41
18	test2	electric	virtual	31.10.2024 11:39	pending	1	29	31.10.2024 18:22

Figure 7. Input dummy data from database

For this test only the virtual tests are scheduled *(s_test_bench == "virtual")* and only pending tests *(status == "pending")*.

From the initial test date *"dt_prescheduled_test_date"*, the estimated test duration *"s_prescheduled_test_duration"* and the due date *"dt_due_date"* the algorithm calculates an occupation time plan considering the objectives.

As mentioned in Sec 4.2.4 Definition of the objective function the highest priority objective is the minimization of the delays. The second priority is the minimization of the vacancies.

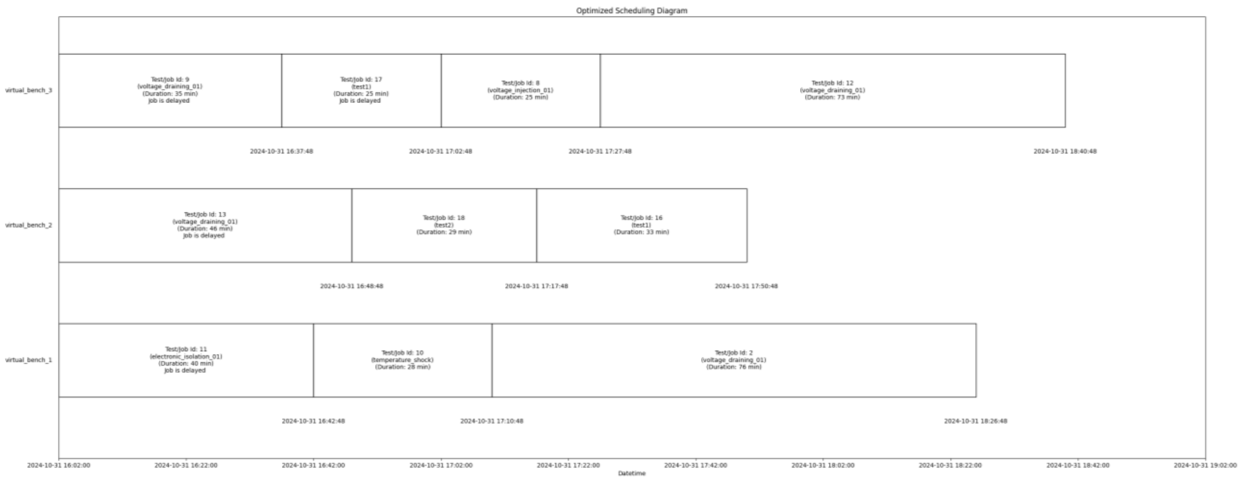

Figure 8. Optimized scheduling diagram

After running the algorithm, the results are illustrated as Figure 8. Optimized scheduling diagram.

Each test in the diagram is shown as a block, where the x-axis marks the start and end date of the test. The y-axis is defined as the test bench, each test is distributed to. On each block information about the test can be found. If the test exceeds the predefined due date, an extra information "Job is delayed" is added to the block. This diagram only serves for a better readability and as base for the evaluation in this deliverable.

To make this data available in LIMS, this information is added to the database as well. Below in the table an excerpt of the result table is shown to highlight the relevant information for this test:

Id	TestId	dt_user_preferenced_virtual_schedul	dt_optimized_virtual_schedul	related_to_test	scheduled_test_bench
18	9	31.10.2024 11:39	31.10.2024 16:02	voltage_draining_01	virtual_bench_3
19	10	31.10.2024 11:39	31.10.2024 16:42	temperature_shock	virtual_bench_1
20	11	31.10.2024 11:39	31.10.2024 16:02	electronic_isolation_01	virtual_bench_1
21	12	31.10.2024 11:39	31.10.2024 17:27	voltage_draining_01	virtual_bench_3
22	13	31.10.2024 11:39	31.10.2024 16:02	voltage_draining_01	virtual_bench_2
23	16	31.10.2024 11:39	31.10.2024 17:17	test1	virtual_bench_2
24	17	31.10.2024 11:39	31.10.2024 16:37	test1	virtual_bench_3
25	18	31.10.2024 11:39	31.10.2024 16:48	test2	virtual_bench_2
26	2	31.10.2024 11:39	31.10.2024 17:10	voltage_draining_01	virtual_bench_1
27	8	31.10.2024 11:39	31.10.2024 17:02	voltage_injection_01	virtual_bench_3

Figure 9. Verification results

The test was executed at 31.10.2024 16:02. The due date and execution date have been chosen intentionally in a way that some delays are inevitable to check the behaviour of the algorithm in that case.

Verification of the task scheduling

1. Occupation:

Considering the result table and the result diagram it is evident that there is no overlaying occupation of test benches.

Interpreting results regarding the minimization:

2. Delay:

The Tests with Test ID 9, 11, 13 and 17 are delayed.

For test 9 and 11 it is not possible to be in time since the due date is already in the past.

For test 13 the due date is at 16:38 on 31.10.2024 which is after the start time of 16:02. But looking at the test duration of 46 min, the earliest finish date would be 16:42, which is 4 minutes late.

For test 17 the due date is at 16:41 on 31.10.2024 and the test duration is 25 min. This results in the earliest finish date of 16:27, which would have been in time. Still the algorithm schedules the test in a way which results in a delay. (test 17 finishes at 17:02)

The reason for the delay of test 17 is that the primary objective is not minimize the number of delays, but the total sum of delay times. This ensures that no test will be scheduled way back in the end to prefer other tests, e.g. the algorithm could have started with Test 17 on "virtual_bench_1" instead of Test 9. In that case there would have been one less delayed test. On the other hand, Test 9 would have been delayed even more, which is also not wanted.

The optimization in this test run can also be seen that Test 17 is assigned to the first available machine after Test 9, 11 and 13 have finished, which is after Test 9.

3. Vacancy:

The total vacancy of all Tests is zero.

The *"dt_prescheduled_test_date"* in this example data is before the execution date of the algorithm. In this case all tests are immediately released and therefore any vacancy would indicate a malfunction of the algorithm.

6. Conclusion & Next steps

This deliverable presents a comprehensive solution that integrates an optimized scheduling algorithm into the FASTEST project's LIMS, enabling a more agile and effective approach to battery testing. The work achieved through the algorithm's development, deployment, and integration represents significant advancements beyond traditional scheduling methods. The innovations include a refined B&B algorithm adapted for multithreaded environments, customized heuristics for task prioritization, and seamless real-time interaction between the LIMS and DT systems. These achievements have led to substantial gains in reducing idle time, minimizing delays, and enhancing resource utilization across test benches.

The project has contributed new insights into the use of real-time data integration in scheduling and demonstrated the feasibility of a hybrid approach that incorporates both virtual and physical testing environments. Future steps will focus on expanding the algorithm's capabilities to incorporate with the development of WP2 (DoE) and WP5 (DT) in FASTEST project.

7. Bibliography

Gurobi Optimization LLC, MIP. (2024). *URL: https://www.gurobi.com/resources/mixed-integer-programming-mip-a-primer-on-the-basics/.*

Gurobi Optimizer LLC, Reference Manual. (2024). *Reference Manual.* URL: https://www.gurobi.com/documentation/current/refman/index.html.

MS Azure. (2024). *Azure Functions.* URL: https://learn.microsoft.com/en-us/azure/azure-functions/.

Seelbach, H. (1975). *Ablaufplanung. Bd. 8. Physica Paperback.* Heidelberg: Physica-Verlag.

Wang, T. (. (2018). *Parallel machine scheduling with precedence constraints (Diss.).* Ecole centrale de Nantes.