

COST-AWARE TASK SEQUENCING WITH FLEXIBLE TIME WINDOWS

PÉTER VERES¹–ZOLTÁN BARÁTH²–HRVOJE CAJNER³

Abstract: Classical scheduling methods are generally designed to optimize a single objective, such as minimizing completion time or the number of delayed jobs. However, practical production environments are frequently characterized by multiple and often conflicting objectives, including delivery performance and economic considerations. In this study, an extended scheduling framework is proposed in which, besides processing times and due dates, task-specific delay costs are incorporated into the decision-making process. A representative production scenario consisting of 10 and 100 tasks is used to evaluate the proposed approach. Three optimization models are developed and implemented using Microsoft Excel Solver. The first model maximizes the number of tasks completed within their due dates, the second minimizes the total due time, while the third minimizes the total delay-related cost. The obtained schedules are further compared with the results of several classical dispatching rules. It is demonstrated that different optimization objectives lead to substantially different production sequences and scheduling performance. The proposed framework provides a simple and transparent decision-support approach that can be implemented without specialized optimization software and can be readily adapted to various production environments.

Keywords: Production scheduling; Multi-criteria optimization; Time windows; Task sequencing

1. IMPORTANCE OF SEQUENCING ALGORITHMS AND STRATEGIES

Scheduling is one of the fundamental optimization problems encountered in operations research and industrial engineering. It appears in a wide range of application areas, including manufacturing systems, transportation, healthcare, cloud computing, construction management, and service operations. The objective of a scheduling problem is generally to determine an appropriate execution sequence for a set of tasks while satisfying predefined constraints and optimizing one or more performance measures. Depending on the application, these measures may include completion time, lateness, resource utilization, throughput, or operational cost [1, 2].

Over the past decades, numerous solution methods have been developed for scheduling problems. Although exact optimization techniques are capable of producing optimal solutions, their computational complexity often increases rapidly with problem size. Consequently, simple dispatching rules remain widely used in industrial practice due to their low computational requirements, ease of implementation, and satisfactory performance. Classical priority rules, such as First Come First Served (FCFS), Shortest Processing Time (SPT), Earliest Due Date (EDD), Minimum Slack Time (MST), Critical Ratio (CR), and the Moore–Hodgson algorithm, have been extensively investigated and are still frequently applied in production planning and shop-floor control [3]. In many practical situations, these

¹ Associate professor, University of Miskolc, Hungary
peter.veres@uni-miskolc.hu

² PhD student, University of Miskolc, Hungary
zoltan.barath@student.uni-miskolc.hu

³ Professor, University of Zagreb, Croatia
hrvoje.cajner@fsb.hr

methods are capable of producing near-optimal schedules while requiring only a fraction of the computational effort of more sophisticated optimization techniques.

Scheduling plays a particularly important role in logistics and manufacturing systems, where sequencing decisions directly influence productivity, delivery reliability, and operating costs. Typical logistics applications include production scheduling, workforce assignment, machine allocation, warehouse operations, loading and unloading activities, and vehicle arrival scheduling at production facilities or distribution centres. In these environments, sequencing decisions affect not only production efficiency but also inventory levels, customer satisfaction, and the utilization of available resources [4].

In real industrial environments, however, scheduling decisions are rarely based solely on processing times or due dates. Different production orders may represent substantially different economic values, contractual obligations, or penalty costs. Consequently, the same amount of delay may result in entirely different financial consequences depending on the task being postponed. To address these practical requirements, numerous weighted scheduling models have been proposed, where jobs are assigned different priorities according to their importance, processing costs, customer value, or penalty functions. Such approaches are commonly formulated as weighted completion time, weighted delays, or cost-oriented scheduling problems [5, 6].

Besides exact optimization approaches, heuristic and dispatching-rule-based methods continue to dominate industrial applications because they provide a favourable balance between solution quality and computational efficiency. Their transparency also makes them attractive for practitioners, particularly in small and medium-sized enterprises where specialized optimization software is often unavailable. Furthermore, modern spreadsheet-based optimization tools have made it possible to solve relatively complex scheduling problems using widely accessible software environments, such as Microsoft Excel Solver, thereby reducing implementation costs while maintaining acceptable solution quality.

Motivated by these considerations, the present study investigates a cost-oriented scheduling framework implemented in Microsoft Excel Solver. Three optimization objectives are examined: maximizing the number of tasks completed before their due dates, minimizes the total delay time, and minimizing the total delay-related cost. The obtained schedules are compared with the solutions generated by several classical dispatching rules, including FCFS, SPT, EDD, MST, CR, and Moore–Hodgson. The comparison is performed using a representative production scenario consisting of 10 and 100 tasks in order to illustrate how different optimization objectives influence the resulting production sequence and scheduling performance.

2. CLASSICAL SCHEDULING METHODS USED FOR PRODUCTION SEQUENCING

Numerous scheduling methods have been proposed in the literature, ranging from simple dispatching rules to sophisticated optimization algorithms. In industrial practice, classical dispatching rules continue to be widely applied because of their simplicity, low computational requirements, and ease of implementation. Although these methods generally optimize only a single performance criterion, they often provide satisfactory solutions for practical scheduling problems and therefore serve as useful benchmark approaches. The most commonly applied classical scheduling methods considered in this study are briefly presented below.

2.1 First Come First Served (FCFS)

The First Come First Served (FCFS) rule processes tasks according to their arrival order. Owing to its simplicity and fairness, the method is frequently applied in production systems, warehouses, and transportation processes. However, no optimization criterion is considered.

2.2 Shortest Processing Time (SPT)

The Shortest Processing Time (SPT) rule always selects the available task with the shortest processing time. For a single-machine environment, it minimizes the average completion time.

$$j^* = \arg \min_{j \in A(t)} p_j \quad (1)$$

where the following notation is used throughout this chapter:

- j is the current task index.
- j^* is the selected task according to the applied scheduling rule.
- $j \in A(t)$ is the set of tasks available for processing at decision time (t).
- t is the current decision time.
- p_j is the processing time of task (j).
- d_j is the due date of task (j).
- w_j is the weight or priority assigned to task (j).
- $\arg \min$ is a function that returns the index (or identifier) of the element for which the expression reaches its minimum value.

2.3 Earliest Due Date (EDD)

The Earliest Due Date (EDD) rule prioritizes the task having the earliest due date. It is optimal for minimizing the maximum lateness in classical single-machine scheduling.

$$j^* = \arg \min_{j \in A(t)} d_j \quad (2)$$

2.4 Minimum Slack Time (MST)

The Minimum Slack Time (MST) rule dynamically calculates the remaining scheduling flexibility for each available task and gives priority to the smallest value. The task having the minimum slack is processed first.

$$SL_j = d_j - t - p_j \quad (3)$$

$$j^* = \arg \min_{j \in A(t)} SL_j \quad (4)$$

2.5 Critical Ratio (CR)

The Critical Ratio (CR) method evaluates the urgency of each task by comparing the remaining available time with its processing time. The task with the smallest critical ratio is assigned the highest priority.

$$CR_j = \frac{d_j - t}{p_j} \quad (5)$$

$$j^* = \arg \min_{j \in A(t)} CR_j \quad (6)$$

2.6 Weighted Shortest Processing Time (WSPT)

The Weighted Shortest Processing Time (WSPT) rule extends the SPT algorithm by incorporating task priorities or economic weights. Tasks are ordered according to the ratio of processing time and weight. The method is commonly applied in cost-oriented production scheduling.

$$j^* = \arg \min_{j \in A(t)} \frac{p_j}{w_j} \quad (7)$$

2.7 Moore–Hodgson Algorithm

The Moore–Hodgson algorithm minimizes the number of delayed jobs on a single machine. The procedure consists of the following steps:

1. Sort all tasks according to the EDD rule.
2. Process tasks sequentially.
3. Whenever a due date is violated, remove the task with the longest processing time from the current schedule.
4. Place all removed tasks at the end of the sequence.

2.8 Heuristic Optimization and Excel Solver

As scheduling problems become larger and more constrained, classical dispatching rules are often unable to provide satisfactory solutions. Consequently, heuristic and metaheuristic optimization methods have become widely adopted for solving complex scheduling problems within acceptable computational times. Among the most frequently applied approaches are Genetic Algorithms (GA), Simulated Annealing (SA), Tabu Search (TS), Ant Colony Optimization (ACO), Particle Swarm Optimization (PSO), and Variable Neighborhood Search (VNS) [7, 8]. Although these methods do not guarantee globally optimal solutions, they are capable of producing high-quality schedules for large-scale and NP-hard optimization problems [9].

For small and medium-sized scheduling problems, however, general-purpose optimization tools often provide an attractive alternative. Microsoft Excel Solver is one of the most widely available optimization environments and supports linear, nonlinear, and evolutionary optimization methods. It requires no specialized programming knowledge, can easily be integrated into existing spreadsheet-based planning systems, and allows different objective functions and operational constraints to be evaluated within the same optimization framework [10]. Since the scheduling scenarios investigated in this study consist of a limited number of production tasks, Excel Solver provides sufficient computational performance while offering a transparent and easily reproducible implementation. Consequently, it was selected as the optimization platform for the proposed scheduling framework.

3. PROPOSED SCHEDULING FRAMEWORK

To evaluate the applicability of different scheduling strategies, a representative single-machine production scenario was developed. The case study consists of ten independent production tasks, each characterized by an arrival time, an estimated processing time, a due date, and a task-specific delay cost. Unlike classical scheduling problems, where only processing times and due dates are considered, the proposed framework also incorporates the economic consequences of delayed task completion.

Each task (j) is defined by the parameter set

$$J_j = (r_j; p_j; d_j; c_{\{1,j\}}; c_{\{2,j\}}) \quad (8)$$

where (r_j) denotes the arrival (release) time, (p_j) represents the expected processing time, (d_j) is the due date, while ($c_{\{1,j\}}$) and ($c_{\{2,j\}}$) denote the unit delay costs before and after a predefined delay threshold (dt_j). The total delay cost of task (j) is therefore calculated as

$$DC_j = \begin{cases} 0; & TD_j = 0 \\ c_{\{1,j\}} \times TD_j; & TD_j \leq dt_j \\ c_{\{2,j\}} \times TD_j; & TD_j > dt_j \end{cases} \quad (9)$$

The processing time represents the estimated duration required to complete a task under normal operating conditions. Since production environments are inherently dynamic, the estimated values are assumed to be known during the scheduling process. Furthermore, tasks cannot be started before their specified arrival times.

Unlike most classical scheduling models, two different delay cost levels are assigned to each production task. The first cost level is applied during the initial delay period, whereas the second, typically higher, cost level is activated once the delay exceeds a predefined time threshold. In the case studies presented in this paper, this threshold is uniformly set to 60 minutes; however, it can easily be modified according to the characteristics of the investigated production environment. This assumption reflects practical industrial situations in which the negative consequences of delays may increase disproportionately over time. Initially, delays may result only in temporary production disturbances or reduced operational efficiency. However, prolonged delays can lead to significantly greater economic losses.

It should be emphasized that the proposed delay cost does not necessarily represent a direct financial penalty. Instead, it reflects the estimated overall economic impact associated with a delayed task. Such costs may include contractual penalties, additional labour costs, production rescheduling, machine idle time, customer dissatisfaction, complaint handling, administrative effort, loss of reputation, or even future customer loss. Consequently, the delay cost can be interpreted as a generalized economic priority assigned to each production order.

The two-level delay cost function allows different business priorities to be represented without significantly increasing the complexity of the scheduling model. Orders with low economic importance may therefore tolerate longer delays, while strategically important or customer-critical orders become increasingly expensive to postpone.

The proposed scheduling framework is subsequently evaluated using three different optimization objectives implemented in Microsoft Excel Solver. The objective functions and optimization models are presented in the following section.

3.1 Test Datasets

To evaluate the scheduling methods at different levels of problem complexity, two datasets were developed. The same task parameters and cost structure were applied in both cases, while the number of tasks was increased from 10 to 100.

The first dataset contains 10 production tasks and represents a relatively small scheduling problem. This size was intentionally selected because the complete problem can still be interpreted manually, and the effects of individual sequencing decisions can be followed without computational assistance. Consequently, the resulting schedules can be easily compared, and the reasons behind differences between the investigated methods can be identified. This dataset is therefore primarily used to demonstrate the behaviour of the scheduling rules and the proposed optimization objectives in a transparent manner.

The second dataset consists of 100 production tasks and represents a considerably more complex scheduling environment. At this problem size, manual sequencing and systematic evaluation of alternative task orders become impractical due to the rapidly increasing number of possible sequences. For (n) independent tasks, the number of theoretically possible sequences is given by

Thus, while the 10-task problem already provides 10! possible sequences, the 100-task problem results in 100! potential permutations. Although classical dispatching rules can still generate solutions efficiently, the larger dataset provides a more demanding test of their ability to produce favourable schedules under heterogeneous arrival times, processing times, deadlines, and delay costs.

Table I.
Input parameters of the 10-task scheduling dataset

Task no.	Arrival time (min)	Latest completion time (min)	Expected duration (min)	Delay cost 0–60 min (EUR/min)	Delay cost above 60 min (EUR/min)
1	0	205	34	1.7	2.9
2	8	75	47	0.6	0.6
3	18	310	39	0	0
4	30	145	61	1.1	2.1
5	45	260	52	0.8	0.8
6	65	185	68	2	3.5
7	90	450	41	0.3	1
8	115	235	57	1.4	2.6
9	145	385	49	0	0
10	175	295	63	0.9	1.8

Both datasets were generated to contain tasks with substantially different scheduling characteristics. Narrow and wide time windows were included, while the deadlines were intentionally not arranged according to arrival order. Different processing times and delay-cost levels were also assigned to the tasks. Therefore, no single simple sequencing rule was expected to perform favourably according to all investigated performance indicators.

The 10-task dataset is used primarily for the detailed interpretation and comparison of individual schedules, whereas the 100-task dataset is employed to examine whether the observed differences between the scheduling approaches remain visible when the problem

size is increased. In both cases, the same scheduling assumptions, delay-cost calculation, and evaluation criteria are applied. Due to its size, the complete 100-task dataset cannot be reasonably presented within the page limitations of this paper. Therefore, the full dataset, including all input parameters, is made available separately at the following link: https://geik.uni-miskolc.hu/intezetek/LOG/content/206/206_4.xlsx. Only the final results obtained for the 100-task case are presented and compared in Section 4.

4. RESULTS OF CASE STUDIES

The investigated scheduling methods were first evaluated using the 10-task dataset. The performance of each method was assessed according to three indicators: the number of tasks completed on time, the summarized delay time, and the total delay cost. In addition to the seven scheduling rules introduced previously, three Solver-based solutions were generated. The Solver optimization was performed separately by maximizing the number of tasks completed on time (Solver-DtD), minimizing the summarized delay time (Solver-Dt), and minimizing the total delay cost (Solver-DC). This separation allows the influence of different scheduling objectives on the resulting sequence to be directly examined.

4.1 Results of the classical scheduling methods of 10 task case study

The FCFS algorithm can be considered the basic reference case, since tasks are processed according to their arrival order without considering processing time, deadline, or economic priority. In the investigated case, 4 of the 10 tasks were completed on time, while a summarized delay of 601 minutes and a total delay cost of €1,264.4 were obtained. As expected, FCFS provided a feasible and transparent sequence, but no specific performance indicator was optimized.

Table II.
FCFS algorithm on the 10-task scheduling dataset

Seq.	Start time (min)	End time (min)	Due time (min)	DtD?	Delay time (min)	Delay Cost (€)
1	0	34	205	YES	0	0
2	34	81	75	NO	6	3.6
3	81	120	310	YES	0	0
4	120	181	145	NO	36	39.6
5	181	233	260	YES	0	0
6	233	301	185	NO	116	406
7	301	342	450	YES	0	0
8	342	399	235	NO	164	426.4
9	399	448	385	NO	63	0
10	448	511	295	NO	216	388.8
SUM:				4	601	1264.4

The SPT rule also resulted in 4 tasks being completed on time. However, the summarized delay increased considerably to 881 minutes and the total delay cost reached €2,172.5, representing the highest cost among all investigated solutions. This result illustrates an important limitation of processing-time-oriented sequencing. Although short tasks are

prioritized, the deadlines and economic consequences of postponing longer tasks are ignored. Therefore, good performance in terms of processing efficiency does not necessarily result in favourable delivery or economic performance.

Table III.
SPT algorithm on the 10-task scheduling dataset

Seq.	Start time (min)	End time (min)	Due time (min)	DtD?	Delay time (min)	Delay Cost (€)
1	0	34	205	YES	0	0
3	34	73	310	YES	0	0
7	73	114	450	YES	0	0
2	114	161	75	NO	86	51.6
9	161	210	385	YES	0	0
5	210	262	260	NO	2	1.6
8	262	319	235	NO	84	218.4
4	319	380	145	NO	235	493.5
10	380	443	295	NO	148	266.4
6	443	511	185	NO	326	1141
SUM:				4	881	2172.5

A considerably different behaviour was observed for WSPT. Although the same number of tasks was completed on time as with FCFS and SPT, the total delay cost was reduced to only €387.3. The summarized delay remained relatively high at 712 minutes. This result is consistent with the underlying principle of WSPT, since economically important tasks are given greater priority. Consequently, delay was shifted towards tasks for which postponement resulted in lower or zero economic consequences. The comparison between SPT and WSPT is particularly illustrative: both methods completed 4 tasks on time, but the delay cost obtained by WSPT was approximately 82% lower.

Table IV.
WSPT algorithm on the 10-task scheduling dataset

Seq.	Start time (min)	End time (min)	Due time (min)	DtD?	Delay time (min)	Delay Cost (€)
1	0	34	205	YES	0	0
6	34	102	185	YES	0	0
8	102	159	235	YES	0	0
4	159	220	145	NO	75	157.5
5	220	272	260	NO	12	9.6
10	272	335	295	NO	40	36
2	335	382	75	NO	307	184.2
7	382	423	450	YES	0	0
3	423	462	310	NO	152	0
9	462	511	385	NO	126	0
SUM:				4	712	387.3

Identical results were obtained using MST, CR, and EDD. In all three cases, only 2 tasks were completed on time, while the summarized delay was 466 minutes and the total delay

cost was €363.2. Although the mathematical principles of these methods are different, the characteristics of the investigated dataset caused the same sequence to be selected. The results also demonstrate that a relatively low summarized delay does not necessarily correspond to a high number of on-time tasks. In these cases, delay was distributed among several tasks rather than concentrated on a small number of tasks.

Table V.
MST, CR & EDD algorithms on the 10-task scheduling dataset

Seq.	Start time (min)	End time (min)	Due time (min)	DtD?	Delay time (min)	Delay Cost (€)
1	0	34	205	YES	0	0
2	34	81	75	NO	6	3.6
4	81	142	145	YES	0	0
6	142	210	185	NO	25	50
8	210	267	235	NO	32	44.8
5	267	319	260	NO	59	47.2
10	319	382	295	NO	87	156.6
3	382	421	310	NO	111	0
9	421	470	385	NO	85	0
7	470	511	450	NO	61	61
SUM:				2	466	363.2

The Moore–Hodgson algorithm produced a substantially different result. Eight tasks were completed on time, which was the highest value achieved by any of the classical methods. However, the two remaining tasks accumulated considerable delays, resulting in 479 minutes of summarized delay and a total delay cost of €1,309.3. This behavior is consistent with the objective of the Moore–Hodgson algorithm: the number of delayed tasks is minimized rather than the magnitude or economic impact of their delay. Thus, a small number of tasks may be heavily postponed if this allows a larger number of other tasks to be completed within their deadlines.

Table VI.
Moore–Hodgson algorithm on the 10-task scheduling dataset

Seq.	Start time (min)	End time (min)	Due time (min)	DtD?	Delay time (min)	Delay Cost (€)
2	0	47	75	YES	0	0
4	47	108	145	YES	0	0
1	108	142	205	YES	0	0
8	142	199	235	YES	0	0
5	199	251	260	YES	0	0
3	251	290	310	YES	0	0
9	290	339	385	YES	0	0
7	339	380	450	YES	0	0
6	380	448	185	NO	263	920.5
10	448	511	295	NO	216	388.8
SUM:				8	479	1309

4.2 Solver-based optimization results of 10 task case study

Three additional sequences were generated using the proposed Solver-based optimization approach. Each optimization was performed using a different objective function while the same task parameters and scheduling constraints were maintained.

When the number of tasks completed on time was maximized (Solver-DtD), 8 tasks were completed within their deadlines. This is equal to the result obtained using Moore–Hodgson. However, the summarized delay increased to 534 minutes and the total delay cost reached €1,681.8. This demonstrates that maximizing the number of on-time tasks alone does not control the severity or economic consequences of delays affecting the remaining tasks.

Table VII.
Solver-Metahuristics-DtD algorithm on the 10-task scheduling dataset

Seq.	Start time (min)	End time (min)	Due time (min)	DtD?	Delay time (min)	Delay Cost (€)
2	0	47	75	YES	0	0
4	47	108	145	YES	0	0
1	108	142	205	YES	0	0
5	142	194	260	YES	0	0
10	194	257	295	YES	0	0
3	257	296	310	YES	0	0
9	296	345	385	YES	0	0
7	345	386	450	YES	0	0
8	386	443	235	NO	208	540.8
6	443	511	185	NO	326	1141
SUM:				8	534	1681.8

When summarized delay time was minimized (Solver-Dt), a substantially different solution was obtained. Five tasks were completed on time, while the total delay was reduced to 368 minutes, representing the lowest summarized delay among all investigated methods. The associated delay cost was €718.7. Therefore, minimizing overall delay produced a more balanced distribution of delays but did not maximize the number of tasks completed on time.

Table VIII.
Solver-Metaheuristics-Dt algorithm on the 10-task scheduling dataset

Seq.	Start time (min)	End time (min)	Due time (min)	DtD?	Delay time (min)	Delay Cost (€)
2	0	47	75	YES	0	0
4	47	108	145	YES	0	0
6	108	176	185	YES	0	0
1	176	210	205	NO	5	8.5
5	210	262	260	NO	2	1.6
3	262	301	310	YES	0	0
8	301	358	235	NO	123	319.8
9	358	407	385	NO	22	0
7	407	448	450	YES	0	0
10	448	511	295	NO	216	388.8
SUM:				5	368	718.7

Finally, the Solver-DC model was optimized directly according to the economic consequences of delay. Four tasks were completed on time and the summarized delay was 482 minutes. Nevertheless, the total delay cost was reduced to only €182.4, which represents the lowest value among all investigated methods. Compared with the basic FCFS solution, the delay cost was reduced by approximately 85.6%, despite the same number of tasks being completed on time. The result clearly indicates that economically favourable scheduling does not necessarily require the minimization of physical delay. Instead, delays can deliberately be assigned to tasks with low economic consequences while more expensive tasks are prioritized.

Table IX.
Solver-Metaheuristics-DC algorithm on the 10-task scheduling dataset

Seq.	Start time (min)	End time (min)	Due time (min)	DtD?	Delay time (min)	Delay Cost (€)
2	0	47	75	YES	0	0
4	47	108	145	YES	0	0
6	108	176	185	YES	0	0
1	176	210	205	NO	5	8.5
8	210	267	235	NO	32	44.8
10	267	330	295	NO	35	31.5
5	330	382	260	NO	122	97.6
7	382	423	450	YES	0	0
9	423	472	385	NO	87	0
3	472	511	310	NO	201	0
SUM:				4	482	182.4

4.3 Comparison of the scheduling results of 10 task case study

The comparison reveals that no single scheduling method provides the best result according to all three performance indicators. This observation is particularly important because the investigated objectives are partially conflicting. Maximizing the number of tasks completed on time tends to concentrate delay on a smaller number of tasks, whereas minimizing summarized delay distributes smaller delays across a larger number of tasks. Cost minimization introduces another perspective by considering the economic importance of individual delays rather than their duration alone.

Among the classical methods, Moore–Hodgson was clearly the most effective when the number of tasks completed on time was considered, while MST, CR, and EDD provided relatively low summarized delay and delay cost. WSPT demonstrated the advantage of incorporating economic priority, producing a considerably lower cost than FCFS and SPT without changing the number of on-time tasks.

The Solver results further emphasize the importance of selecting an appropriate objective function. Each of the three Solver variants provided the best or near-best result for the indicator that was explicitly optimized. Solver-DtD matched the maximum number of on-time tasks obtained by Moore–Hodgson, Solver-Dt achieved the lowest summarized delay, and Solver-DC produced by far the lowest delay cost. The latter reduced the delay cost by approximately 50% compared with the best classical result (€182.4 compared with €363.2).

Table X.
Summarizing table on the algorithms on the 10-task scheduling dataset

Methods	Tasks done on time	Summarized delay time (min)	Summarized delay cost (€)
FCFS algorithm	4	601	1264.4
SPT algorithm	4	881	2172.5
WSPT algorithm	4	712	387.3
MST algorithm	2	466	363.2
CR algorithm	2	466	363.2
EDD algorithm	2	466	363.2
Moore–Hodgson algorithm	8	479	1309.3
Solver-Metahuristics-DtD	8	534	1681.8
Solver-Metahuristics-Dt	5	368	718.7
Solver-Metahuristics-DC	4	482	182.4

From a computational perspective, well-implemented classical scheduling algorithms remain extremely efficient. For example, EDD and Moore–Hodgson can be implemented with a computational complexity of $O(n \log n)$, making them highly suitable even for considerably larger scheduling problems. However, such algorithms must either be specifically implemented or accessed through an existing software library. In contrast, the Solver-based approach requires practically no dedicated algorithm development. For the investigated 10-task case, the optimization model could be configured in Microsoft Excel within approximately one minute once the input structure had been prepared. All three optimization variants were solved within a few seconds, and no individual Solver run required more than 10 seconds. Therefore, although the computational efficiency of specialized scheduling algorithms is clearly superior from a theoretical perspective, the spreadsheet-based approach offers a considerable practical advantage in terms of implementation simplicity, accessibility, and the possibility of rapidly changing the objective function.

4.4 Comparison of the scheduling results of 100 task case study

The 100-task dataset was used to investigate whether the main observations obtained from the smaller case remained valid when the problem size was considerably increased. The summarized results are presented in Table XI.

Similarly to the 10-task case, no single method provided the best performance according to all investigated criteria. Moore–Hodgson achieved the highest number of tasks completed on time, with 79 tasks, confirming its effectiveness when this indicator is considered the primary objective. Solver-Metaheuristics-DtD followed with 71 on-time tasks. In contrast, Solver-Metaheuristics-Dt provided the lowest summarized delay of 32,554 minutes, while Solver-Metaheuristics-DC resulted in the lowest delay cost of €26,787.59. The latter represents an approximately 79.8% cost reduction compared with FCFS and a 58.4% reduction compared with the best classical result obtained by CR.

The larger dataset therefore confirms the main conclusion of the 10-task experiment: optimization of different objectives results in substantially different task sequences. A high number of on-time tasks does not necessarily correspond to low summarized delay or low

economic impact. In particular, the cost-based optimization was able to shift delays towards tasks for which postponement had less severe economic consequences.

A considerable difference was observed in computational requirements. While the 10-task Solver models were solved within a few seconds, substantially longer solution times were required for the 100-task case. The longest Solver run required approximately 1 minute and 42 seconds. Furthermore, the same solution quality was not always obtained in the first run, and repeated optimization occasionally resulted in improved solutions. This behavior is expected from the metaheuristic Solver approach and represents an important difference compared with deterministic classical scheduling rules, which provide their corresponding sequence directly and with considerably lower computational effort.

Table XI.
Summarizing table on the algorithms on the 100-task scheduling dataset

Methods	Tasks done on time	Summarized delay time (min)	Summarized delay cost (€)
FCFS algorithm	38	91771	132425.98
SPT algorithm	49	65733	118063.29
WSPT algorithm	7	44247	75314.23
MST algorithm	7	45245	76462.78
CR algorithm	7	37887	64349.32
EDD algorithm	51	85149	72309.87
Moore–Hodgson algorithm	79	49377	92151.21
Solver-Metahuristics-DtD	71	80068	156724.92
Solver-Metahuristics-Dt	36	32554	47991.63
Solver-Metahuristics-DC	49	50894	26787.59

5. DISCUSSION AND CONCLUSIONS

In this study, classical scheduling rules and a flexible Solver-based optimization approach were compared using 10-task and 100-task scheduling problems. The results demonstrated that no single scheduling method can be considered universally superior, as the obtained performance strongly depends on the selected objective. Classical methods such as EDD, CR, SPT, WSPT, and particularly Moore–Hodgson can generate useful schedules with very low computational requirements. Moore–Hodgson was especially effective when the number of tasks completed on time was considered, confirming that well-designed specialized algorithms remain highly competitive even when compared with more flexible optimization approaches.

The main advantage of classical scheduling rules is their computational efficiency, deterministic behaviour, and simplicity once they have been implemented. Their main limitation is that they are generally designed for a specific objective and cannot easily represent more complex business considerations. The investigated Solver-based approach provides the opposite trade-off. Its computational requirements are higher and, particularly for the 100-task problem, repeated runs may be required to obtain a sufficiently good solution. Nevertheless, the model can be implemented and modified without developing a

dedicated scheduling algorithm, while different objectives and operational constraints can easily be incorporated.

The cost-oriented results were particularly relevant. Minimizing delay time alone was shown to be different from minimizing the economic consequences of delay. By allowing different tasks to have different delay costs, delays can intentionally be shifted towards tasks for which postponement has a lower operational or economic impact. Therefore, relatively small changes in the physical scheduling performance may result in substantial reductions in the overall delay cost.

Based on the obtained results, heuristic or metaheuristic optimization should not be considered a replacement for classical scheduling algorithms. Instead, it represents a useful complementary approach when the scheduling problem contains multiple objectives, task-specific costs, or constraints that cannot conveniently be handled by conventional priority rules. For small and medium-sized problems, spreadsheet-based optimization may therefore provide an accessible and sufficiently effective decision-support solution, particularly when flexibility and ease of implementation are more important than obtaining a theoretically optimal solution.

The present study is limited by the use of two artificially generated datasets and a single-machine scheduling environment. Future research should therefore include real industrial datasets, additional problem sizes, dynamic changes in processing times and task arrivals, and more complex production environments with multiple machines and resource constraints.

ACKNOWLEDGEMENT

The research presented in this article was supported in part by Project No. 2023-1.2.4-TÉT-2023-00027, implemented with the support of the Ministry of Culture and Innovation of Hungary from the National Research, Development and Innovation Fund and financed under the 2023-1.2.4-TÉT funding scheme, and by the project Simulation and Modelling of Material Handling Systems (SIM_MFS), co-funded by the Ministry of Science, Education and Youth of the Republic of Croatia within the bilateral Hungarian–Croatian cooperation programme.

REFERENCES

- [1] Pinedo, M. (1995). *Scheduling. theory, algorithms and systems*. Prentice Hall.
- [2] Baker, K. R. & Trietsch, D. (2018). *Principles of sequencing and scheduling*. John Wiley & Sons. <https://doi.org/10.1002/9781119262602>
- [3] Stanković, A. & Petrović, G. (2025). Comparative analysis of job priority rules in single-machine scheduling under uncertainty. *Innovative Mechanical Engineering*, 4(3), 65-77.
- [4] Szentesi, S. & Nagy, G. (2025). The impact of inventory management innovations on Industry 4.0 manufacturing strategies. *Tehnički glasnik*, 19(1), 100-105, <https://doi.org/10.31803/tg-20250410143130>
- [5] Chan, F. T. S., Chan, H. K. & Lau, H. C. W. (2002). The state of the art in simulation study on FMS scheduling: a comprehensive survey. *The International Journal of Advanced Manufacturing Technology*, 19(11), 830-849, <https://doi.org/10.1007/s001700200095>
- [6] Ramasesh, R. (1990). Dynamic job shop scheduling: a survey of simulation research. *Omega*, 18(1), 43-57, [https://doi.org/10.1016/0305-0483\(90\)90017-4](https://doi.org/10.1016/0305-0483(90)90017-4)
- [7] Kareem, S. W., Ali, K. W. H., Askar, S., Xoshaba, F. S. & Hawezi, R. (2022). Metaheuristic algorithms in optimization and its application: A review. *JAREE (Journal on Advanced Research in Electrical Engineering)*, 6(1), 7-12, <https://doi.org/10.12962/jaree.v6i1.216>
- [8] Hayat, I., Tariq, A., Shahzad, W., Masud, M., Ahmed, S., Ali, M. U. & Zafar, A. (2023). Hybridization of particle swarm optimization with variable neighbourhood search and simulated

- annealing for improved handling of the permutation flow-shop scheduling problem. *Systems*, **11**(5), 221, <https://doi.org/10.3390/systems11050221>
- [9] Itu, A. (2025). Industrial scheduling in the digital era: Challenges, state-of-the-art methods, and deep learning perspectives. *Applied Sciences*, **15**(19), 10823, <https://doi.org/10.3390/app151910823>
- [10] Kantianis, D. (2025). Practical Spreadsheet-Based Modelling for Effective Construction Project Scheduling. *Urban Planning and Construction*, **3**(2), 77-96, <https://doi.org/10.55121/upc.v3i2.1058>