applied sciences

MDPI

Article

Efficient Task Scheduling Using Constraints Programming for Enhanced Planning and Reliability

JaeBong Cho [1], Soonil Jung [1], Kyungmo Yang [1], Dohun Kim [2] and WonJong Kim [2,*]

[1] HCNC, Seongnam 13486, Republic of Korea; jbcho@hcnc.co.kr (J.C.); sijung@hcnc.co.kr (S.J.); kmyang@hcnc.co.kr (K.Y.)
[2] Electronics & Telecommunications Research Institute, Seongnam 13488, Republic of Korea; dohun@etri.re.kr
* Correspondence: wjkim@etri.re.kr

Abstract: This paper presents an efficient schedule method for maintenance, repair, and overhaul (MRO) tasks for aircraft engines using a constraint programming algorithm. Using data obtained from Korean Air's MRO maintenance logs, we analyze and predict the optimal scheduling of regular inspections and fault repairs for various engine types. By proposing a proper modeling of the problem and preparing data for the constraint programming algorithm, we demonstrate superior performance in scheduling efficiency and resource utilization. The experimental results show an average utilization of 99.35%, and the method can even achieve 100% utilization in some cases.

Keywords: MRO scheduling; aircraft engines; constraint programming; maintenance optimization and repair

Citation: Cho, J.; Jung, S.; Yang, K.; Kim, D.; Kim, W. Efficient Task Scheduling Using Constraints Programming for Enhanced Planning and Reliability. *Appl. Sci.* **2024**, *14*, 11396. https://doi.org/10.3390/app142311396

Academic Editors: Juan Carlos Seck Tuoh Mora, Liliana Lizárraga-Mendiola, Joselito Medina-Marín and Norberto Hernández-Romero

Received: 31 October 2024
Revised: 28 November 2024
Accepted: 4 December 2024
Published: 6 December 2024

1. Introduction

Aircraft engine maintenance, repair, and overhaul (MRO) are critical operations that ensure flight safety, operational reliability, and efficiency. The scheduling of MRO tasks is a complex challenge due to the intricate nature of maintenance procedures, the variety of engine types, and the need to minimize aircraft downtime. Efficient MRO task scheduling is vital from economic, technical, and operational perspectives, directly impacting the profitability and reliability of airlines.

From an economic standpoint, efficient MRO scheduling can significantly reduce operational costs. Aircraft downtime due to maintenance is costly, as it involves both direct costs, such as labor and parts, and indirect costs, such as lost revenue from grounded aircraft. Optimized scheduling minimizes these downtimes, leading to a better utilization of assets and resources. Furthermore, effective scheduling can prevent overstocking or stockouts of critical spare parts, thus optimizing inventory levels and reducing carrying costs.

Technically, MRO scheduling requires a deep understanding of the maintenance tasks and their dependencies. Aircraft engines are complex systems, and their maintenance involves various procedures, including disassembly, inspection, repair, and reassembly. Each of these tasks has specific requirements, such as skilled labor, specialized equipment, and precise timing. Efficient scheduling ensures that all necessary resources are available when needed, preventing delays and bottlenecks. It also facilitates the integration of advanced diagnostic tools and predictive maintenance technologies, which can forecast potential failures and optimize maintenance intervals.

Operationally, MRO scheduling is closely linked with flight schedules. Airlines must ensure that maintenance activities do not disrupt flight operations. This involves careful planning to align maintenance windows with aircraft availability, considering factors such as flight routes, turnaround times, and regulatory requirements. Efficient MRO scheduling helps maintain a balance between keeping aircraft in the air and ensuring they are safe and well-maintained. It also supports operational flexibility, allowing airlines to adapt quickly to unforeseen events such as unscheduled repairs or changes in flight schedules. MRO

scheduling can be integrated with a digital twin-based smart manufacturing framework such as in [1].

The objective of this study is to develop an efficient MRO task scheduling system for aircraft engines. By analyzing data from Korean Air's MRO maintenance logs, we aim to optimize the scheduling of regular inspections and fault repairs for various engine types. Our proposed algorithm improved scheduling efficiency and resource utilization, ultimately enhancing the economic, technical, and operational aspects of MRO activities.

In summary, efficient MRO task scheduling is essential for minimizing costs, ensuring technical accuracy, and maintaining operational continuity. This study addresses these challenges by proposing a novel algorithm that can optimize MRO scheduling for aircraft engines.

2. Related Works

The scheduling of maintenance, repair, and overhaul (MRO) tasks for aircraft engines has been extensively studied, with various methods proposed to address the complexity and resource-intensive nature of these operations. This section reviews the most relevant works in the field, highlighting their contributions and limitations, and provides a comparison with the approach proposed in this paper.

Aircraft MRO processes are inherently complex, involving multiple interdependent tasks that must be carefully scheduled to minimize downtime and reduce operational costs. The scheduling challenges in MRO operations, particularly focusing on the relationship between Original Equipment Manufacturers (OEMs) and MRO providers, and the operational difficulties caused by unexpected component failures are described in [2]. While this study provided a comprehensive overview of the key factors influencing MRO efficiency, it primarily relied on qualitative analysis, offering limited solutions for optimizing task scheduling under uncertainty. This paper aims to build upon these insights by proposing a quantitative, constraint-based optimization approach to handle such challenges more effectively.

The task scheduling problem with resource constraints, such as job shop scheduling, is a NP-hard problem, as proved in [3]. There are many heuristic approaches to solve NP-hard problems.

Many studies have employed heuristic and metaheuristic methods to solve this NP-hard scheduling problem. For example, a genetic algorithm was used to optimize long-term aircraft maintenance scheduling under uncertainty, demonstrating a 7% reduction in heavy maintenance checks and a 4.4% increase in aircraft utilization [4]. Although this approach showed promising results, heuristic methods like genetic algorithms often struggle with finding truly optimal solutions, especially in dynamic environments where task dependencies and resource availability fluctuate frequently.

Metaheuristic approaches like genetic algorithms, simulated annealing, or particle swarm optimization often provide near-optimal solutions but are computationally expensive and may require the fine-tuning of parameters for specific cases. In contrast, our study leverages constraint programming, which, while computationally more straightforward for certain problem sizes, guarantees optimal solutions within a reasonable time for the given MRO task size. By focusing on a more structured optimization approach, our model efficiently handles task dependencies and resource constraints, which are crucial in real-world MRO settings.

A NeuroEvolution of Augmenting Topologies (NEAT) algorithm for dynamic hybrid flow shop scheduling problems characterized by uncertain processing times, dynamic job arrivals, and flexible maintenance was proposed in [5]. The study introduced a multi-agent framework for efficient scheduling policy development and demonstrated that the NEAT algorithm outperforms traditional dispatching rules and Deep Q-Network (DQN) in terms of robustness, adaptability, and generalization across diverse test cases. This approach emphasizes real-time decision making and provides a robust methodology for addressing scheduling challenges in smart manufacturing environments.

Machine learning and deep learning methods have gained popularity in MRO scheduling for their ability to model complex relationships in large datasets. A reinforcement learning (RL) was applied to optimize aircraft maintenance scheduling, showing that RL could outperform traditional methods in certain stable environments [6]. Similarly, adaptive RL for task scheduling in aircraft maintenance, introducing a dynamic rescheduling mechanism that responds to real-time changes in task requirements was explored in [7].

A Multi-Policy Deep Reinforcement Learning (DRL) framework for solving the multi-objective multiplicity flexible job shop scheduling problem (MOMFJSP) was introduced in [8]. The study employs a Multi-Policy Proximal Policy Optimization (MPPPO) algorithm with a co-evolution mechanism to simultaneously optimize makespan and total tardiness. The approach leverages multiple policy networks to generate diverse Pareto-optimal solutions, guided by an adaptive reward mechanism. Comparative analyses show that MPPPO outperforms traditional dispatching rules and other DRL-based methods in terms of efficiency, convergence, and diversity, providing robust solutions for complex scheduling scenarios.

However, while deep learning and RL approaches can offer flexible, data-driven solutions, they often require large amounts of training data and computational resources, making them less practical for smaller-scale problems or environments with limited data availability. Additionally, these methods may struggle to incorporate strict task dependencies and constraints without significant modifications to the learning algorithms. In contrast, our proposed constraint programming approach does not rely on extensive data or require lengthy training phases, making it more suitable for structured, deterministic MRO environments where real-time data analysis is a primary focus.

Operational research (OR) tools have been widely applied in the aviation industry to tackle complex scheduling problems [9]. A lookahead approximate dynamic programming (ADP) methodology was introduced to optimize aircraft maintenance check scheduling under uncertainty, achieving a significant reduction in maintenance slots and improving aircraft availability [10]. While ADP and other OR-based approaches are effective for handling large-scale optimization problems with stochastic elements, they often rely on approximations and may not guarantee truly optimal solutions.

Our study differs from these approaches by utilizing constraint programming, which focuses on deterministic optimization. The key advantage of constraint programming in this context is its ability to find exact solutions in cases where the problem size is manageable, such as the MRO task scheduling problem presented in this paper. This contrasts with the approximate nature of ADP and other OR methods, which may not always provide the most efficient schedule.

While prior studies have made significant advances in optimizing MRO scheduling through heuristic, machine learning, and operational research methods, each of these approaches has limitations in handling dynamic and unpredictable MRO environments or in balancing the trade-offs between solution quality and computational efficiency. Our proposed constraint programming model addresses these gaps by focusing on a structured optimization approach that leverages task dependencies, resource constraints, and workplace allocation to achieve high scheduling efficiency and resource utilization.

Compared to heuristic and deep learning-based models, our constraint programming approach offers a more precise solution in a reasonable time frame without requiring extensive computational resources. By using maintenance log data obtained from Korean Air's MRO, we demonstrate that this method can achieve superior resource utilization (up to 100% in some cases) and provide near-optimal solutions for small-to-medium-sized MRO scheduling problems, where real-time data analysis is required.

3. Methodology

This section outlines the methodology used to develop and evaluate the proposed MRO task scheduling algorithm. It includes details on data collection, preprocessing, model development, and the proposed hybrid algorithm.

3.1. Data Preparation

The data used in this study were collected from Korean Air's MRO (Seoul, Republic of Korea) maintenance logs. The dataset includes historical records of various engine maintenance tasks and subtasks, such as disassembly, inspection, repair, and reassembly. Each record contains information about the task type, duration, start time, and completion time.

To estimate the duration of each task and subtask, we calculated the average duration based on historical data. This approach provides a reliable estimate for scheduling purposes. The average duration values were derived from past maintenance records, ensuring they reflect realistic time frames for task completion.

We focused on scheduling MRO tasks for a one-year period. This timeframe allowed for the inclusion of regular inspections, planned maintenance, and unexpected repairs, providing a comprehensive view of the scheduling requirements and challenges.

Table 1 shows an example of a simplified table describing subtasks and their working days for an PW1100G-type engine depending on the level of the work. Each subtask has maintenance levels from 0 to 3 depending on the difficulty of the subtask, which results in working days for the maintenance. There is a set of whole tables like Table 1 for all the engine types.

Table 1. An example table of subtasks and working days of an PW1100G-type engine (unit: day).

Subtask\Level	L0	L1.3	L1.9	L2.3	L2.3	L2.3	L2.3	L2.9	L3
FRG	1	1	7		2	6	6	7	6
FDBG	1	2	5	8				8	8
FDGG	1	1	3	2					4
FICG	1	4	5	5				6	5
2BG	1	1	2	3					4
FCG	1	1	3	7				9	8
LCSG	1	2	5	6				9	8
LCRG	1	1	4	3					4
25BLG	1	1	2	3				4	4
CICG	1	1	3	5				6	6
3BG	1	1	3	4					5
HCRSG	1	2	7	24				25	22
HCFCG	1	1	7	11				22	20
HCRG	1	2	7	21				22	19
DCG	1	1	3	6					8
CTNG	1	1	2	3	4	6	10		11
HTSG	1	2	4	10				13	12
HTRG	1	1	2	3				5	5
TICG	1	2	9	8				10	9
LTSG	1	2	6	37				23	20
LTRG	1	2	4		5	6	7	8	7
TECG	1	3	9	10				11	10
MGBG	1		5	11					16
AGBG	1		2	5					5
Other				10					

Table 2 shows an example task table describing a set of subtasks to be conducted for the checkup of an engine. Working days were extracted from Table 1 based on subtask and level. Most of the regular checkups have similar tables depending on the checkup level or fault type and engine type. So, the table has a name like type_level_version.csv, for example, PW1100G_A_1.0.csv. These tables are maintained as a library for the MRO system.

Table 2. An example checkup task table for PW1100G-type engine.

No.	Subtask	Level	Days
1	HCRSG	L0	1
2	HCFCG	L0	1
3	HCRG	L0	1
4	DCG	L0	1
5	CTNG	L0	1
6	HTSG	L0	1
7	HTRG	L0	1
8	TICG	L1.3	2
9	LTRG	L1.3	2
10	TECG	L1.3	3
11	FDGG	L1.9	3
12	FICG	L1.9	5
13	FCG	L1.9	3
14	LCSG	L1.9	5
15	25BLG	L1.9	2
16	CICG	L1.9	3
17	LTSG	L1.9	6
18	MGBG	L1.9	5
19	AGBG	L1.9	2
20	Other	L2.3	10

To plan MRO tasks for the year, we needed a list of expected tasks as shown in Table 3, where the start day is the expected start day of the task from the beginning of the year. Each csv file has a task table like Table 2.

Table 3. Tasks table example for a period.

No.	Task	Start Day
1	PW1100G_A.csv	10
2	PW1100G_C.csv	15
3	PW1100G_D.csv	20
4	PW1100G_A.csv	30
5	PW1100G_C.csv	40
6	PW1100G_D.csv	50
	.....	

3.2. Application of Constraints Programming

3.2.1. Constraint Definition

We used constraint programming for the scheduling. The constraints considered in the optimization process included the following:

1. Limited Number of Workplaces: The number of workplaces available for performing tasks is limited. Each workplace can perform any MRO task or subtask.
2. Task Dependencies: Certain tasks must be completed before others can begin.
3. Resource Availability: Ensuring that the necessary resources (e.g., tools, personnel) are available for each task.
4. Workplace Allocation: Allocating tasks to available workplaces to maximize efficiency and minimize idle time.
5. Minimum Start Date: Some work will arrive at some points when the check-up time arrives.

3.2.2. Problem Definition

There are N tasks and M workplaces. Each task J_i can only be executed in a specific workplace W_j, and each task has a start time S_i and a duration d_i. The goal is to schedule all tasks across the workplace and minimize the total completion time.

3.2.3. Mathematical Model

Variables

- S_{ij}: Start time of task J_i in workplace W_j (i = 1, 2, . . ., N; j = 1, 2, . . ., M).
- C_{ij}: Completion time of task J_i in workplace W_j.
- C_{max}: Maximum completion time among all tasks.

Domains

- $S_{ij} \in \{0, 1, 2, \ldots, T\}$, where T is the maximum possible time.

Constraints

- Constraint-1: Each task can only be executed in specific workplaces. a_{ij} is a binary variable indicating if task J_i can be executed in workplace W_j (1 if possible, 0 if not possible):

$$S_{ij} \ a_{ij} = S_i \ \forall i,j$$

- Constraint-2: The completion time of each task is the sum of its start time and its duration:

$$C_{ij} = S_{ij} + d_i \ \forall i,j$$

- Constraint-3: Tasks in the same workplace must not overlap. For tasks J_i and J_k in workplace W_j, one of the following must hold:

$$S_{ij} + d_i \leq S_{kj} \text{ or } S_{kj} + d_k \leq S_{ij} \ \forall i \neq k, \forall j$$

- Constraint-4: The overall completion time C_{max} is the maximum of the completion times of all tasks:

$$C_{max} \geq C_{ij} \ \forall i,j$$

Objective Function

- The goal is to minimize the overall completion time: Minimize C_{max}

Figure 1 shows the conceptual visualization of the mathematical model.

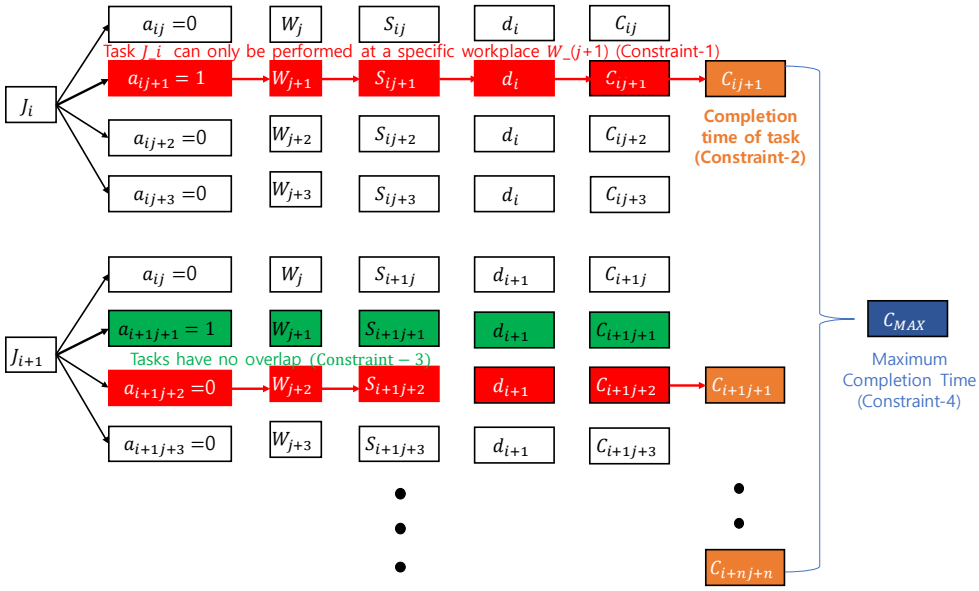

Figure 1. Visualization of mathematical model.

Example

Suppose there are 3 tasks and 2 workplaces:

- Task J_1: duration $d_1 = 3$, possible workplace W_1
- Task J_2: duration $d_2 = 2$, possible workplaces W_1, W_2
- Task J_3: duration $d_3 = 4$, possible workplace W_2

Constraints:

- $a_{11} = 1, a_{12} = 0$
- $a_{21} = 1, a_{22} = 1$
- $a_{31} = 0, a_{32} = 1$

Based on these constraints, the domain, and objective function, a constraint programming solver can find the optimal task schedule.

For example, tasks could be assigned, and their start and completion times determined as follows:

- $S_{11} = 0, C_{11} = 3$
- $S_{m21} = 3, C_{21} = 5$
- $S_{32} = 0, C_{32} = 4$

In this case, the overall completion time $C_{max} = 5$. This modeling approach enables the efficient allocation and scheduling of tasks.

3.3. Experimental Setup

The CP algorithm was evaluated using the dataset from Korean Air, focusing on scheduling MRO tasks for a one-year period. The performance of the CP algorithm was examined for several cases, demonstrating its superior efficiency and resource utilization. For the MRO in Korean Air, Constr-1 was not considered, since any workplace can handle any task for efficient operations.

3.4. Justification for Selecting Constraints Programming

The choice of constraint programming (CP) for solving the maintenance, repair, and overhaul (MRO) task scheduling problem was primarily driven by the nature and complexity of the problem. Unlike large-scale problems commonly addressed by machine learning (ML) or metaheuristic methods, MRO scheduling typically involves a manageable number

of tasks and constraints, making CP an efficient and optimal choice. Below, we outline the key reasons for selecting constraint programming and compare it to alternative methods.

3.4.1. Complexity of the Scheduling Problem

MRO scheduling is inherently a combinatorial optimization problem, with tasks having dependencies, limited resources, and specific temporal constraints. As demonstrated in the literature, such problems are often NP-hard, meaning that finding an optimal solution requires significant computational effort, especially as the problem size grows. While heuristic and metaheuristic methods like genetic algorithms or simulated annealing can be used to provide near-optimal solutions for large, complex problems, they do not guarantee optimality and can require significant parameter fine-tuning.

The worst-case complexity of CP is $O(d^n)$, where d is the domain size of task start times, and n is the number of tasks. However, CP is well-suited for smaller, structured problems with well-defined constraints. CP systematically explores the search space using constraint satisfaction techniques, ensuring that the optimal solution is found when the problem size remains within a practical range. This is particularly advantageous in the MRO context, where the task set is not as large as in typical machine learning or deep learning applications. The finite nature of MRO tasks (e.g., scheduled inspections and repairs) makes CP an efficient method for exploring the solution space exhaustively, rather than for relying on approximation techniques.

3.4.2. Deterministic vs. Stochastic Environments

Many of the alternative approaches, particularly those based on machine learning or reinforcement learning (RL), are designed to handle dynamic or stochastic environments where task requirements may change in real time. For instance, reinforcement learning has shown potential in adapting to unpredictable changes in task availability or resource constraints. However, these methods require extensive training data and computational resources, which may not always be available or practical in MRO settings.

In contrast, the MRO scheduling problem in this study is relatively deterministic, with the task requirements and constraints known in advance. While unforeseen maintenance events (e.g., unscheduled repairs) can occur, the focus of this paper is on optimizing the scheduling of regular inspections and fault repairs, which follow predictable patterns. Constraint programming excels in such deterministic environments, where it can efficiently model the problem using predefined constraints and deliver optimal schedules without the need for large training datasets or dynamic updates. This deterministic nature makes CP a more appropriate choice for this specific problem.

3.4.3. Resource Utilization and Task Dependencies

In MRO scheduling, it is critical to ensure that resources (e.g., workplaces, tools, and personnel) are optimally allocated, and that task dependencies are strictly followed. Metaheuristic approaches like genetic algorithms and simulated annealing can handle these constraints but often do so through approximation, potentially leading to suboptimal resource utilization or violations of task dependencies.

CP, by contrast, is designed to handle these types of complex, interdependent constraints natively. In our approach, the task dependencies, resource availability, and workplace limitations are explicitly modeled as part of the constraint satisfaction process. This allows the CP algorithm to ensure that all tasks are scheduled in accordance with their dependencies and that resource allocation is optimized. The result is a highly efficient schedule that maximizes resource utilization (up to 100% in some cases, as shown in the experimental results) while minimizing idle time across workplaces.

3.4.4. Computational Efficiency

While machine learning and deep learning approaches can handle larger-scale problems, they are often computationally expensive and require significant amounts of training

time and data preprocessing. For smaller-scale problems like the MRO task scheduling problem discussed in this paper, these approaches may be unnecessarily complex and resource-intensive.

The computational efficiency of constraint programming is another key reason for its selection. As shown in the experimental results, our CP-based method is able to generate near-optimal schedules in a matter of seconds to minutes, even as the number of tasks increases. For example, scheduling 24 tasks across 6 workplaces can be completed in under 74 s on a standard PC setup. This is considerably faster than many machine learning approaches, which could take hours or days to train, depending on the dataset size and the complexity of the model. CP provides an optimal solution with significantly lower computational overhead.

3.4.5. Scalability Considerations

While CP is efficient for small to medium-sized problems, its scalability may be a concern for very large-scale MRO environments where the number of tasks and constraints grows significantly. However, in this study, the size of the MRO task scheduling problem is well within the capabilities of CP. Future work could explore hybrid approaches that combine CP with heuristic or machine learning techniques to address scalability in larger environments, but for the current scope, CP offers the best trade-off between computational efficiency and optimal solution quality.

3.4.6. Flexibility in Handling Task Variations

Another advantage of CP is its flexibility in handling variations in task requirements. Unlike heuristic methods, which may require significant re-tuning for different task sets, CP can easily accommodate changes in the number of tasks, task durations, or resource constraints by updating the model's constraints without the need for retraining or extensive modifications to the algorithm. This adaptability makes CP a more straightforward and efficient solution for the structured, repeatable nature of MRO task scheduling.

By selecting constraint programming for the MRO scheduling problem, this study leverages the strengths of CP in handling structured, deterministic optimization problems with well-defined constraints. This choice is validated by the experimental results, which show high scheduling efficiency and resource utilization, making CP an ideal method for the specific challenges of aircraft engine MRO scheduling.

4. Experimental Results

4.1. Performance Metrics

The performance metrics used for evaluation were as follows:

- The number of days required to finish all the given work.
- The number of workplaces required to finish all the given work in a year of 260 working days.
- Resource utilization--evaluated based on the optimal use of available workplaces.

4.2. Case Studies

First, we explain the experimental results with small examples, since real-world cases are difficult to use for demonstration. Figures 2 and 3 show the scheduling results without and with start days constraints, respectively. The numbers in each box represent {Task ID}_{SubTask ID} and the numbers above an arrow show the start day constraints. The horizontal axis shows the timeline in days, and the vertical axis represents the three workplaces (0~2). The color coding or shading differentiate the tasks to help visualize how tasks are distributed across workplaces over time.

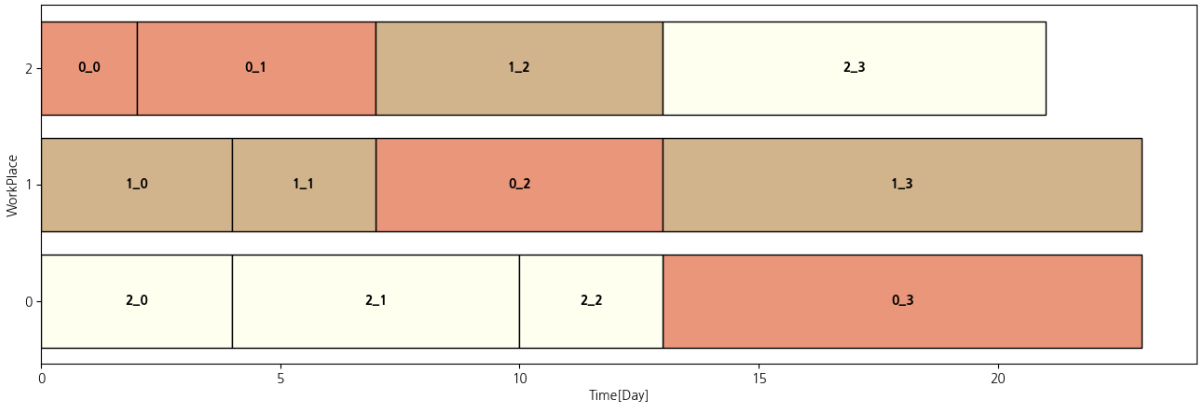

Figure 2. A simple scheduling result without start day constraint.

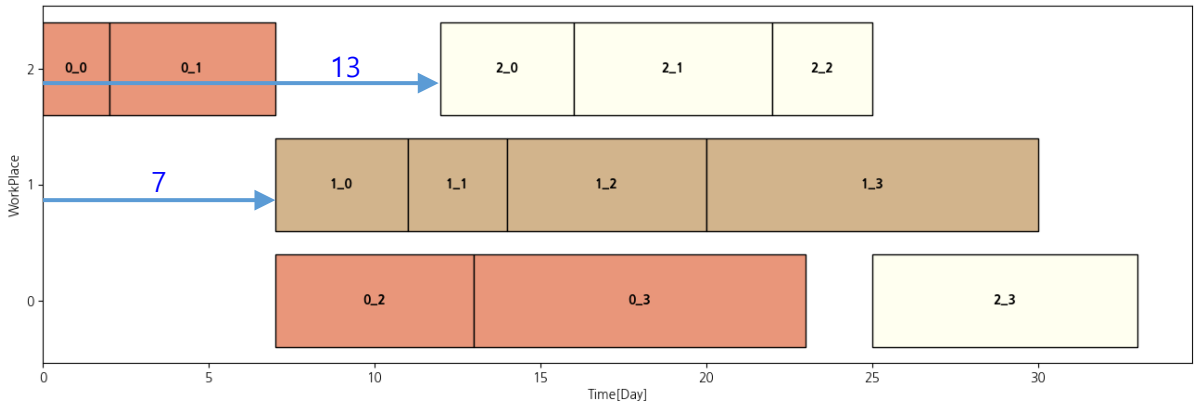

Figure 3. A simple scheduling result with start day constraint as shown in blue arrows.

4.3. Summary of Experimental Results

We tested several cases by duplicating the 3 given works to 6, 9, 12, and 24. Up to 12 works, all the works can be conducted with three workplaces within a year of 260 working days. However, in the case of 24 works, works cannot be performed with three workplaces, and this should be increased to six workplaces, as shown in Table 4. In Table 4, horizon is the maximum days boundary expected when all the tasks are assigned consecutively. CPU time increases when the number of tasks increases. Most of the cases achieved over 99% utilization, except the first case. Some cases even have 100% utilization. The average utilization of all the cases is 99.34%. We used an Intel i7-14700K-based PC with Ubuntu 22.04. In Table 4, we compared the results with a genetic algorithm (GA) using a population size of 50; 30,000 generations; and mutation rate of 0.1. To obtain the optimum solution using the genetic algorithm, we had to tune the parameters by trying several combinations of parameters. When using the parameters for a similar order of CPU time, the GA could not find optimal solutions.

When the minimum start days for each work is given, the total days required increases, as shown in Table 5. The increased days and utilization depend on the given minimum start days.

The observed CPU times follow an expected trend: as the number of tasks increases, the CPU time required to compute the optimal schedule grows. This is due to the increased complexity of the problem, which involves a larger number of task dependencies, resource constraints, and possible combinations of task assignments across workplaces. The computational effort required to explore the solution space and satisfy the constraints grows with the size of the problem, leading to a nonlinear increase in CPU time.

Table 4. Experimental results for various tasks and workplaces without minimum start days.

Tasks	Work Places	Horizon [Days]	CP (Ours)			GA		
			Days	Utilization	CPU [s]	Days	Utilization	CPU [s]
3	3	175	61	95.63%	1.37	90	64.81%	11.63
6	3	350	117	99.72%	2.82	181	64.46%	21.03
9	3	525	175	100.00%	6.90	236	74.15%	30.27
12	3	700	234	99.72%	22.77	292	79.91%	39.62
24	3	1400	467	99.93%	26.93	548	85.16%	76.47
24	4	1400	350	100.00%	56.32	456	76.75%	76.12
24	5	1400	280	100.00%	132.96	365	76.71%	75.83
24	6	1400	234	99.72%	73.49	331	70.49%	75.73

Table 5. Experimental results for various tasks and workplaces with minimum start days given.

Tasks	Work Places	Days	CPU [s]
3	3	76	1.30
6	3	132	11.74
9	3	190	35.07
12	3	249	57.49
24	3	470	26.93
24	4	355	59.30
24	5	289	145.93
24	6	244	52.82

However, the increase in CPU time is manageable, even for the larger problem instances tested. See the following example:

For 3 tasks: The algorithm finds a schedule in just 1.37 s.

For 24 tasks: It takes about 73.49 s to compute the schedule, which is still within a reasonable time frame for practical use in MRO environments where scheduling tasks can span weeks or months.

This scalability trend indicates that the constraint programming approach remains computationally feasible for small-to-medium-sized MRO scheduling problems. However, as the problem size continues to increase, it is likely that the CPU time will grow more significantly, potentially posing challenges for very large-scale problems involving hundreds of tasks or more.

The CPU times reported in this study are encouraging, as they indicate that the proposed constraint programming approach can generate high-quality schedules within seconds or minutes for small-to-moderate problem sizes. In practical MRO environments, where maintenance tasks are typically planned well in advance (e.g., weeks or months), these CPU times are highly acceptable. This makes the algorithm suitable for day-to-day scheduling tasks in MRO operations. Even if urgent checkup interruptions occur, it can provide dynamic task scheduling by using a fast automatic task scheduling which can be rescheduled in minutes.

5. Conclusions

This study demonstrates the effectiveness of a constraint programming algorithm for efficient MRO workspace scheduling of aircraft engines. By leveraging data from Korean Air's MRO maintenance logs, the proposed method achieves a good scheduling efficiency

and resource utilization of up to 100%, with an average of 99.34%, highlighting its potential for broader application in the aviation industry.

The CPU time analysis demonstrates that the constraint programming algorithm proposed in this study is highly efficient for solving small-to-medium-sized MRO scheduling problems. While CPU time increases as the number of tasks and workplaces grows, it remains within practical limits for the scenarios tested. Future research should focus on enhancing scalability through parallel computing, incremental scheduling, or hybrid approaches to handle larger and more complex MRO environments efficiently.

By achieving a near-optimal utilization of resources and maintaining computational efficiency, this method holds significant potential for improving MRO operations in the aviation industry. However, to fully realize its potential, future research must address the current limitations and explore ways to scale the solution for larger, more dynamic environments. With further validation and refinement, this approach could play a crucial role in advancing the efficiency, reliability, and sustainability of aircraft maintenance practices.

Author Contributions: Methodology, S.J., D.K. and W.K.; Software, D.K. and W.K.; Validation, D.K. and W.K.; Formal analysis, W.K.; Investigation, J.C.; Resources, J.C. and K.Y.; Data curation, S.J.; Writing—original draft, W.K.; Writing—review & editing, S.J. and D.K.; Visualization, W.K.; Supervision, J.C. and K.Y.; Project administration, J.C.; Funding acquisition, J.C. All authors have read and agreed to the published version of the manuscript.

Funding: This research was supported by the Ministry of Land, Infrastructure and Transport's technology development program [RS-2023-00239124] and the Ministry of SMEs and Startups project "Development of Open Smart Manufacturing Sharing Platform for Enterprise Linkage in Discrete Process Characteristic Industries" [RS-2022-00140586].

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The data presented in this study are available on request from the corresponding author due to corporate trade secrets.

Conflicts of Interest: Authors Jae Bong Cho, Soonil Jung and Kyungmo Yang were employed by the company HCNC. The remaining authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

References

1. Rauf, M.; Guan, Z.; Mumtaz, J.; Yue, L.; Hao, W. Digital twin-based smart manufacturing system for project-based organizations: A conceptual framework. In Proceedings of the 49th International Conference on Computers and Industrial Engineering (CIE49), Beijing, China, 18–21 October 2019; Available online: https://www.researchgate.net/publication/336716177 (accessed on 3 December 2024).
2. Albakkoush, S.; Pagone, E.; Salonitis, K. Scheduling Challenges within Maintenance Repair and Overhaul Operations in the Civil Aviation Sector. In Proceedings of the TESConf 2020—9th International Conference on Through-Life Engineering Services, Cranfield, UK, 3–4 November 2020.
3. Karp, R.M. Reducibility Among Combinatorial Problems. In *Complexity of Computer Computations*; The IBM Research Symposia Series; Springer: Boston, MA, USA, 1972; pp. 85–103.
4. Van der Weide, T.; Deng, Q.; Santos, B.F. Robust long-term aircraft heavy maintenance check scheduling optimization under uncertainty. *Comput. Oper. Res.* **2022**, *141*, 105667.
5. Chen, Y.; Zhang, J.; Rauf, M.; Mumtaz, J.; Huang, S. Dynamic scheduling of hybrid flow shop problem with uncertain process time and flexible maintenance using NeuroEvolution of Augmenting Topologies. *IET Collab. Intell. Manuf.* **2024**, *6*, e12119. [CrossRef]
6. Andrade, P.; Silva, C.; Ribeiro, B.; Santos, B.F. Aircraft Maintenance Check Scheduling Using Reinforcement Learning. *Aerospace* **2021**, *8*, 113. [CrossRef]
7. Silva, C.; Andrade, P.; Ribeiro, B.; Santos, B.F. Adaptive reinforcement learning for task scheduling in aircraft maintenance. *Nat. Sci. Rep.* **2023**, *13*, 16605. [CrossRef] [PubMed]
8. Ding, L.; Guan, Z.; Rauf, M.; Yue, L. Multi-policy deep reinforcement learning for multi-objective multiplicity flexible job shop scheduling. *Swarm Evol. Comput.* **2024**, *87*, 101550. [CrossRef]

9. Deng, Q.; Santos, B.F.; Curran, R. A practical dynamic programming based methodology for aircraft maintenance check scheduling optimization. *Eur. J. Oper. Res.* **2020**, *281*, 256–273. [CrossRef]
10. Deng, Q.; Santos, B.F. Lookahead approximate dynamic programming for stochastic aircraft maintenance check scheduling optimization. *Eur. J. Oper. Res.* **2022**, *299*, 814–833. [CrossRef]