Engineering Journal issn: 0125-8281

Article

Constraint Programming in Single Machine Scheduling for Minimizing Makespan with Multiple Constraints

Manlika Kiatthadasirikul[1,a], Paveena Chaovalitwongse[1,b,*], Naragain Phumchusri[1,c], and Siravit Swangnop[2,d]

1 Department of Industrial Engineering, Faculty of Engineering, Chulalongkorn University, 10330, Bangkok, Thailand
2 Faculty of Engineering, King Mongkut's University of Technology North Bangkok, Bangkok, Thailand
E-mail: [a]6071468421@student.chula.ac.th, [b,*]paveena.c@chula.ac.th (Corresponding author), [c]naragain.p@chula.ac.th, [d]siravit.s@eng.kmutnb.ac.th

Abstract. This study focuses on developing a scheduling model for sequencing a set of jobs with different release times in a single machine to meet non-similar due dates as well as to reduce total sequence-dependent setup time. A constraint programming (CP) model is proposed to solve the scheduling problem by minimizing makespan under multiple constraints, namely release times, sequence-dependent setup time, and due dates. The proposed constraint programming model is tested and compared with the baseline method derived from as-is scheduling of alloy wheels manufactures. The computational experiments show the proposed constraint programming model outperforms the baseline method in the average improvement in makespan and total setup time. For small-size problems, the proposed scheduling model were optimally solved in a short time, achieving the best average improvement in makespan of 4.8826% and the best average improvement in total setup time of 45.7924%. Despite increasing problem sizes, the proposed scheduling model's computational time deteriorates but continues to provide the best solutions, achieving the best average improvement in makespan of 7.4891% and the best average improvement in total setup time of 55.4033%.

Keywords: Production scheduling, constraint programming, sequencing, single machine, makespan minimization.

ENGINEERING JOURNAL Volume 28 Issue 11
Received 15 November 2023
Accepted 6 November 2024
Published 30 November 2024
Online at https://engj.org/
DOI:10.4186/ej.2024.28.11.81

DOI:10.4186/ej.2024.28.11.81

1. Introduction

Production scheduling is a decision-making process at the operation level [1] that plays an important role in the context of manufacturing and production systems refers to the process of allocating limited manufacturing resources such as machines, to tasks or jobs [3] to specific time periods to accomplish a set of manufacturing processes in the plan to achieve certain objectives. It aims to deal with resource utilization and timespan of the manufacturing operations [4].

Scheduling is the most important function of production planning and control activity in manufacturing and engineering. This refers to the process of determining the timing of an operation, that is, The process of determining the starting and completion times for the variety of jobs to be executed on the machine to meet the desired delivery dates. The effect of scheduling decisions has a significant impact on the productivity of a process. By determining what to make, when to make, and in what quantities, production scheduling aims to maximize operational efficiency [5].

Real-world manufacturing environments are typically complex systems that feature various constraints and characteristics that correspond to fairly specific settings found in related companies. The existing varieties of manufactured products and numerous factors, such as processing times, setup times, precedence relations among jobs, due dates, etc., make scheduling a complex issue. The best possible way for manufacturing companies to remain competitive is through appropriate order sequencing [6].

The problem under consideration corresponds to a real problem in an alloy wheel manufacturing company in the automotive industry. Alloy wheels have a wide variety of models that include sizes ranging from 12 to 24 inches, unique patterns and 4 color variations of light color tones, medium color tones, dark color tones and black color. Each model is considered an individual job to be processed. Usually, multiple jobs are being produced simultaneously, with the primary production processes consist of casting, machining, and painting. In the casting and machining processes, there are many identical parallel machines in operation, each identical parallel machine is capable of producing one job at a time. The variety of different jobs that are manufactured on the same time. The casting and machining process releases jobs of various sizes and patterns that are unique but it is possible that several jobs will be the same color in painting process. Therefore, multiple jobs are released to painting process at several times. Each job can start to be processed after its release time. Jobs are the same color can be sequenced by grouping jobs to reduce the setup time because of a setup time is necessary whenever there is a switch from processing a job in one type of setup to another. Jobs have sequence-dependent setup times; they require setup times that are dependent on the sequence in which they are processed.

The sequencing method affects the scheduling performance through the dispatching rules. Some dispatching rules may result in a higher makespan, lead to late delivery and cause high setup time.

This study aims to answer the questions of single machine scheduling problem with constraints that are frequently encountered in actual practice. The majority of contributions to proposing a model involve sequencing jobs with different release times to meet different due dates and reduce setup time where setup times are sequence-dependent. Many choice situations involve several constraints. Determining a solution that satisfies all constraints simultaneously can be challenging when some constraints pull the solution in opposite directions. The difficulty is to develop a model used for sequencing that can handle several constraints. The improvement in makespan is used to measure the effectiveness of the scheduling model, which shows the gap between the objective values of the schedules found by the scheduling model (the optimal schedule) and the baseline scheduling method (the manufacturer schedule).

The rest of the paper in the next section is organized as follows. Section 2 presents a literature review. Section 3 analyzes the case problem with the baseline scheduling method. In section 4, a constraint programming model for makespan minimization in single machine scheduling with release times, sequence-dependent setup times and due dates are developed. In section 5, a difference in instance sizes experiments and result analysis are conducted, and the last section concludes the paper and considers future works.

2. Literature Review

Single machine scheduling problem is one type of the most important problems in production. It is the basic building block to develop more comprehensive scheduling models [7].

Single machine scheduling problems have received a significant amount of attention in the literature. Using two methods to solve a single machine sequencing problem: exact and heuristic [6]. Various approaches are employed in the single machine scheduling problem depends on different factors. The present research study is interested in a problem with minimizing makespan in the objective function. Thus, the limitation of the review is previous papers with minimized makespan objectives or previous papers with constraints are relevant to the current research.

2.1. Single Machine Problems with Release Times

Many scheduling problems consider that the jobs are available at time zero or the beginning of the time horizon, thus ignoring any job release time. However, release times are also very common in real-world manufacturing scenarios [8]. They present a heuristic and a metaheuristic algorithm for solving the challenging single-machine scheduling problem with release dates and sequence-dependent setup times to minimize the makespan. For small-scale instances [9] proposed an effective branch and

bound (B&B) algorithm integrated with an elaborately designed pruning rule and a lower bound is developed to achieve exact solutions for minimizing the sum of makespan on multi-agent single-machine scheduling with release dates. In the literature, there is increasing attention to addressing the beginning of the time horizon is not zero. Several of the research papers define release times as release dates. Therefore, a job cannot be processed before the release date, and those jobs are sequenced in the order of their release dates.

2.2. Single Machine Problems with Sequence-Dependent Setup Times

One of the most immediate consequences of a changeover is the interruption of production. The time allocated to adjusting settings is time that is not used for the production of products. Production scheduling must prioritize reducing unnecessary changeover from period to period [10]. Traditionally, most papers solve this scheduling problem without considering setup times or including them in the processing times of the jobs, which is valid only when the setup times are sequence-independent. Despite, sequence-dependent setup times are very common in scheduling problems involving the real-world manufacturing systems (Fernandez-Viagas & Costa, 2021) as e.g. painting process. The previous and current jobs in the sequence belong to different setup times and changing the positions of each job requires a setup time. The purpose is to perform a scheduling with the lowest setup time sum. According to [11], the development of a single machine scheduling problem where the jobs to be processed arrive the production process with release times and the setup times are sequence-dependent to obtain a sequence with minimum makespan.

2.3. Single Machine Problems with Due Dates

Different performance measures have been studied. Several production control systems are used to meet customer demands and needs (C. Silva et al., 2017). Single machine scheduling for make-to-order (MTO) must address the company's need for short delivery times and on-time deliveries. Since meeting the due dates is important in a competitive environment, due date related performance measures are gaining importance [12]. The most popular due date assignment policy is to assign all jobs a common due date [13]. In the majority of cases, it is widely accepted to make the assumption that the assigned due date are not limited or constrained. However, in several practical cases, setting due dates too far into the future may violate early agreements between the manufacturer and customers [14]. The value of the due date in the relation to the problem data strongly impacts the accuracy of algorithms formulated for a problem type [15].

In all of the aforementioned research, manufacturing scheduling constraints are considered separately. To achieve better results, Integrated decision-making requires considering all production stages simultaneously [16]. Rosyidi et al. developed an integrated optimization model with the objective of maximizing total profit while considering several constraints of a large steel manufacturing company. The company manufactures a wide variety of steel products to fulfill a certain number of demand. Manufacturers increasingly require customization of their products to meet customer demands and distinguish from competition [17], which causes the production complexity. Because the production complexity is higher, it leads to greater diversity in the choice of job sequencing. It causes a delay in delivery. It is challenging to optimize a real-world problem that involves a large number of jobs. Choosing an appropriate optimization method can be challenging when dealing with a multi-constrained optimization problem [18]. Liu et al proposed an appropriate optimization method that can assist users in setting up the system scale more reasonably and reducing the system's disadvantages. It is evident that there are no simple principles for optimizing the production schedule based on different performance indicators [19].

In conclusion, Table 1 shows the related works are mainly focused on some aspects, which, as said above, are not considered in several constraints simultaneously that make the scheduling model more complex and challenging to solve with the objective is to minimize the makespan. The problem has not received much attention in the recent literature. Consequently, the literature motivated us to develop the scheduling model with the aim of producing solutions that are improved with excellent results for several constraints with the objective is to minimize the makespan. This work develops a scheduling model for sequencing different jobs with different release times in a single machine that minimizes the makespan to meet different due dates and to reduce setup time where setup times are sequence-dependent. It considers single machine scheduling using constraint programming to calibrate several constraints and then running an experiment, which allows analyzing the effect of several constraints on the performance of the scheduling model. Accordingly, considering several constraints make the problem realistic and more challenging from a modeling point of view.

DOI:10.4186/ej.2024.28.11.81

Table 1. The primary characteristics of the models considered in the reviewed papers and in this research

Model Features	This research	Wang et al. (2022) [9]	Toksarı & Toğa, (2022) [20]	Yang et al. (2022) [21]	Fernandez-Viagas & Costa (2021) [8]	Badri et al. (2021) [22]	Molaee et al. (2021) [23]	Lunardi et al. (2020) [24]	Yue et al. (2018) [7]	Herr & Goel (2016) [25]	Kır & Yazgan (2016) [26]	Karray et al. (2015) [27]
Problem size												
• Small	√	√	√					√				
• Medium	√							√			√	√
• Large	√			√	√	√	√	√	√			
Time constraints												
• release times	√				√	√		√	√			
• sequence-dependent setup times	√		√		√		√	√		√	√	
• due dates	√					√				√	√	√
Objective function	Min Makespan	Min Makespan	Min Makespan	Min Total weighted completion time	Min Makespan	Min Makespan	Min max tardiness	Min Makespan	Min max waiting time	Min Total tardiness	Min Total earliness and tardiness	Min Makespan, Cost, Time intervals
Solution method							√					
• Heuristics				√	√	√			√	√	√	√
• Hybrid		√										
• Exact	√		√					√				

3. Problem Statement

The problem under consideration corresponds to a real problem of an alloy wheel manufacturing company for the automotive industry. The manufacturing company starts the manufacturing process only after receiving orders from customers. The manufacturing company needs to guarantee a committed delivery due date for each job order, meaning that the customer of the job order must receive the products on or before this date. After the job orders are received, the manufacturer must determine which ones are to be sequenced. The current production time for the desired run is continuous. Continuous production means operating 24 hours per day, 7 days per week due to the aluminum melting furnace melting continuously for 24 hours. The jobs are the production of alloy wheels. There are 25-35 different jobs to produce per weeks in 4 different colors. Alloy wheel production involves several steps to create high-quality wheels. This study focus on the main process of a painting process are washing, powder spraying, paint spraying and clear powder spraying by simplify a long production line to a single machine as visualized in Fig. 1 for reducing a complex manufacturing process that involves multiple steps into a single machine [28]. This simplification can lead to easier planning.

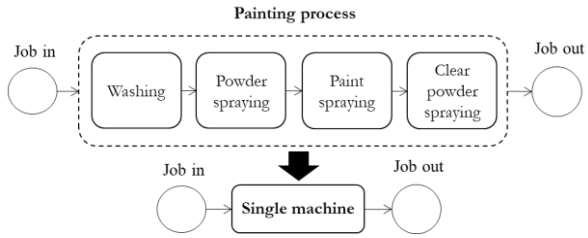

Fig. 1. Simplify a painting process line production to a single machine.

The problem statement is described as follows: This study has focused on the scheduling model aiming to find a desired schedule with the best system performance under the assumption that model parameters are known and deterministic. Consider a sequencing problem that has a set of n jobs, Each job j ($j = 1,\ldots,n$) has a known fixed processing time p_j. The complexity of the problem consideration illustrates by an example of a production scheduling challenge tackled by an existing company. This company produces products of different sizes, patterns and colors. Thus, the setup time of job j on the single machine is sequence-dependent setup time. During a setup time no processing of jobs is possible, i.e.) non-preemptive jobs. A set of jobs is processed in several production stages, but this work focuses on one stage that the jobs must be arranged in a sequence before assigning them to a single machine. Therefore, the release time of job r_j is the completion time Cj of the previous stage, job j cannot be processed before its release time. A different due date d_j is predefined. The objective of the scheduling model is to minimize the makespan while respecting the constraints : release time, sequence-dependent setup time and due date constraints. The scheduling model must be applied to determine good solutions to minimize makespan. A common, and quite popular, measure of the performance of any scheduling procedure, since it combines the desire to secure high utilization of the production resources, together with the desire to ensure early satisfaction of customer demand [29].

4. Constraint Programming Model

In recent decades, industrial engineering focus on the development and implementation of production manufacturing scheduling systems. The focus mentioned arises due to the incorporation of automation in the planning process. Recent technological advances in computer software and hardware allow for reducing the computational time to find near-optimal solutions to complex problem. The proposed modeling approach solved the scheduling problem for many variables as well as all constraints in a short time period.

Constraint Programming is a recent powerful programming paradigm with a great impact on several important areas such as combinatorial optimization problems, especially for complex problems that cannot be modeled easily [30]. It is a highly effective and easy-to-use technology for solving scheduling problems that has shown its capability and efficiency [31].

This work presents a constraint programming (CP) model for makespan minimization in a single machine scheduling with release times, sequence-dependent setup times and due dates. For finding the optimal solution, the proposed modeling approach solved the scheduling problem for many variables as well as all constraints in a short time period.

This facilitates the manufacturer to minimize the makespan of the manufacturing phase while considering the actual constraints on the overall production planning. Assignment, timing and sequencing decisions are derived from the CP model.

The main modeling expression used is interval variable. It represents an interval of time during which an activity is executed, with its required reserved time in the schedule. Interval variables indicate the job operation to be processed on machine, with a start, duration, and end time. Start time is denoted by *startOf* and completion time by *endOf*. While *sizeOf* is given by the problem instance represents the processing time of the job. *startOf* and *endOf* are determined by the CP solver [32].

This model was developed using IBM Decision Optimization CPLEX Modeling for Python, also known as DOcplex supported by the IBM ILOG CPLEX 12.10 optimization studio environment and solved using the CP solver and run on Intel® Core™ i5-4278U CPU @ 2.60GHz running Windows 10.

The development model is based on a single machine with release times, sequence-dependent setup times and due dates problem can be formulated as a constraint programming problem as follows:

Sets/Indexes. *j, j'* : jobs to be sequenced.

Parameters. $processingTimes_j$: processing time of the job *j*. $releasetime_j$: release time of job *j*. $duedate_j$: due date of job *j*. $setupmatrix_{j,j'}$: changeover time between job *j* and job *j'*. $Completion\ time_j$: completion of job *j*.

Notation. Matrix elements are denoted as $setupmatrix_{j,j'}$, where

- *j* is the row index, for all jobs
- *j'* is the column index, for all jobs
- $setupmatrix_{j,j'}$ represents the setup time located in the *j* row and the *j'* column

Example : In the case of the number of jobs with value 5, the matrix is a 5x5 (5 rows, 5 columns), the element in the first row and third column is denoted as $setupmatrix_{1,3}$ = S1. This means if job 3 is sequenced to follow job 1, the setup time value at this position is S1. It can be visualized as Fig. 2:

			To j'				
j	$setuptime_j$	From *j*	1	2	3	4	5
j1	S2	1	0	0	**S1**	S4	S3
j2	S2	2	0	0	S1	S4	S3
j3	**S1**	3	S2	S2	0	S4	S3
j4	S4	4	S2	S2	S1	0	S3
j5	S3	5	S2	S2	S1	S4	0

Fig. 2. Example of $Setupmatrix_{j,j'}$

Variables. $processingTimesVars_j$: interval variable that spans over all the processing times of job j. $sequenceVars_j$: sequence variable represents a sequencing of processing time interval variables associated with job j.

Objective. The main objective of the proposed scheduling model is to minimize the makespan C_{max}, (1) that is, the time of completion of the last job *j*.

$$minimize\ max\ (endOf(Completion\ time_j))\ \forall j \in jobs \quad (1)$$

Constraints. Constraint (2) the duration of each processing time depends on the job assigned to it.

$$processingTimeVars_j = sizeOf(processingTime_j)\ \forall j \in jobs \quad (2)$$

Constraint (3) each job *j* cannot start to be processed before its release time.

$$startOf(processingTimeVars_j) >= releasetime_j\ \forall j \in jobs \quad (3)$$

Constraint (4) avoids lateness of job *j*.

$$endOf(processingTimeVars_j) - duedate_j <= 0\ \forall j \in jobs \quad (4)$$

Constraint (5) possible values are all of the sequence of job j.

$$sequenceVars_j = sequenceVar(processingTimeVars_j)\ \forall j \in jobs \quad (5)$$

Constraint (6) avoids overlapping the execution of jobs and simultaneously inserts the corresponding changeover time between consecutive jobs. All the job variables not overlap with each other which ensures the interval variables in the sequence do not overlap.

$$noOverlap(sequenceVars_j, setupmatrix_{j,j'})\ \forall j \in jobs \quad (6)$$

Table 2. Summary of input data.

Number of jobs ***j***	**Time**			
	$\boldsymbol{duedate_j}$ $\boldsymbol{d_j}$	$\boldsymbol{releasetime_j}$ $\boldsymbol{r_j}$ **[LB,UB]**	$\boldsymbol{setuptime_j}$ $\boldsymbol{st_j}$	$\boldsymbol{processingTimes_j}$ $\boldsymbol{p_j}$
5	[D1,D2]			
10	[D1,D2,D3]			
15	[D1,D2,D3,D4]			
20	[D1,D2,D3,D4,D5]	[0,< $duedate_j$ d_j]	[S1,S2,S3,S4]	200
25	[D1,D2,D3,D4,D5,D6]			
30	[D1,D2,D3,D4,D5,D6,D7]			
35	[D1,D2,D3,D4,D5,D6,D7,D8]			

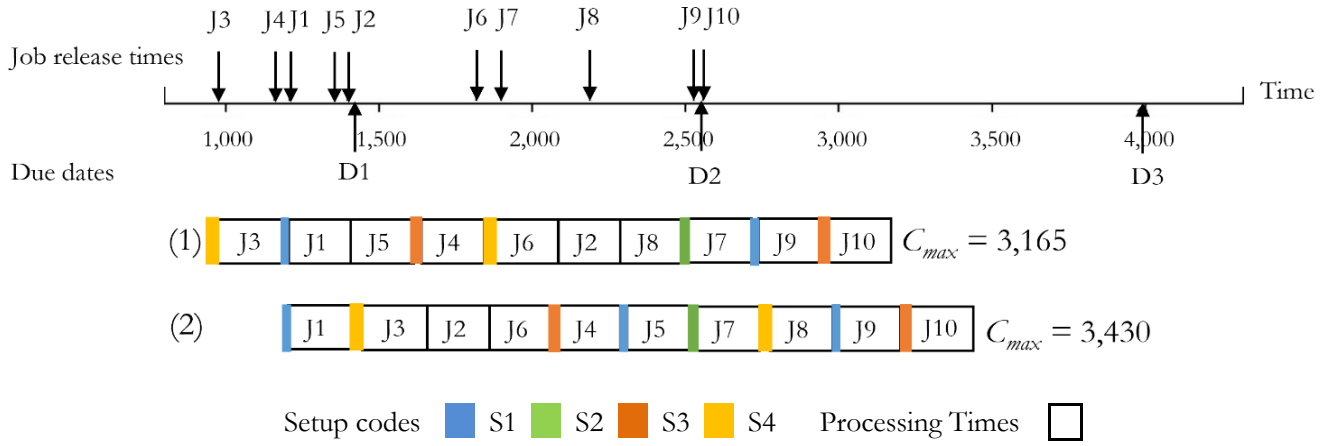

Fig. 3. The example schedule created by (1) the proposed scheduling model; (2) the baseline scheduling method, using the same input data.

5. Numerical Experiments

Experiments are conducted on instances derived from instances of a practical problem in alloy wheel production that motivated the research. The evaluation aims to highlight the difficulty of solving the problem and compare the performance of our proposed scheduling model with the baseline scheduling method is derived from the manufacturer's scheduling method based on common dispatching rules to generate a schedule as the earliest due date rule (EDD).

5.1. Generating Input Data

The jobs are the production of alloy wheels. There are 25-35 different jobs to produce in 4 different colors. Product variety and sequencing scheduling have a large impact on performance. The manufacturer gets orders in the form of a set of n jobs, denoted by job j (j = 1,…,n) that fixed processing times p_j on a single machine because of each job is produced in batches, with one pallet containing approximately 50 pieces of alloy wheel. The release times r_j are given. Different jobs are ready to start at different release times on the different days. Note that the more dispersed the release jobs are, the more challenging the instances are.

shows a summary of input data that a list of jobs must be assigned due dates, release times, setup times, and processing times are generated randomly referred to as a scenario in the following. All jobs [5,10,15,20,25,30,35] with the same processing time equally. **Error! Reference source not found.** shows sample of a set of jobs due dates and release times were created. The due dates d_j are generated randomly from integer uniform distribution within a range and then use that to make a choice from a set of due date. Estimate the shortest and longest possible durations within which jobs can be feasibly completed under the scenario of a tight due dates for demonstrating the impact of due dates constraints on a scheduling model.

In a day consisting of 1,440 minutes, approximately 5 to 7 jobs can be produced. The committed due date agreed upon with the customer is specified by the day. However, the due date for running a model is based on the range of times between the minimum time required before starting the processing of a job and the time required to complete each job varies depending on the number of jobs with values 5 – 35. The release times r_j of the jobs were created within a range based on the nature of the job process time.

The setup times st_j are generated randomly from uniform distributions between 4 types [S1,S2,S3,S4]. For instance, the number of instances would require generating 1,000 different instances that can be solved when the constraint is not very strict for each number of jobs with uniform distribution.

5.2. Comparison of Computational Time

The complexity of the problem e.g., the number of jobs size, multiple constraints can greatly influence the computational time. Instances that are more difficult to solve lead to increased computational time. A sample of 10 instances from each of the number of jobs with values 5, 10, 15, 20, 25, 30, and 35 was selected to test the computational time of instance-solving with different time limits.

To evaluate the proposed scheduling model when increasing CPU time limits.

Table 3 shows the solution found by the proposed scheduling model with a CPU time limits of 10 seconds and 3,600 seconds. The table shows the average gap with bound for the different CPU time limits. In the case of optimality, gap with bound equal 0%. These figures suggest that reasonable quality solutions are found relatively quickly. Computational time results indicated that the CPU time limit 10 seconds performs equal to the CPU time limit 3,600 seconds with respect to the considered performance measures. It is clearly demonstrated that the number of jobs with values 5, 10,

15 can be optimally solved in the CPU time limit 10 seconds.

Interesting aspects were observed in the case of the number of jobs with value 20. When considering extending the time allocated for solving to find an optimal solution from 10 seconds to 3,600 seconds. The proposed scheduling model can find more optimal solutions on number of jobs with a value 20 when the solving time is extended. However, when dealing with larger number of jobs where the proposed scheduling model requires more than 3,600 seconds just to find a feasible solution.

In comparison, Table 4 shows the gap between the size of jobs running with the proposed scheduling model and the time required to solve the optimal solution.

As the number of jobs increases from 5 to 35, the solving time rises relatively with 0.0130 – 3,600 seconds. DOcplex.CP takes a long time to give the optimal solution. Therefore, DOcplex.CP could not show results immediately in a short time.

The CPU time limit 10 seconds is allocated to solve the problem, with the expectation of being able to solve all the instances.

Table 3. Summary of average gap with bound results by the proposed scheduling model of 10 instances for each number of jobs with increasing CPU time limits.

Number of jobs	Computational time	
	10 seconds	3,600 seconds
	Avg. gap with bound (%)	Avg. gap with bound (%)
5	0.0000	0.0000
10	0.0000	0.0000
15	0.0000	0.0000
20	**0.0394**	**0.0000**
25	0.0319	0.0286
30	0.0335	0.0332
35	0.0255	0.0255

Table 4. Summary of average computational time results of the proposed scheduling model of 10 instances for each number of jobs with no time limit.

Number of jobs	Avg. gap with bound (%)	Avg. Computational time (seconds)
5	0.0000	0.0130
10	0.0000	0.0697
15	0.0000	0.3068
20	0.0000	282.6580
25	0.0286	>3,600
30	0.0332	>3,600
35	0.0255	>3,600

5.3. The Baseline Scheduling Method

In the actual world, production schedulers are faced with the challenges of creating a reasonable schedule to deal with multiple constraints.

The manufacturer defines the baseline scheduling method steps are shown in

Therefore, sorting setup time for all of the jobs included in the group of common due dates is sequence-dependent. Eventually, assigning procedure is a sequence of job assignments to a single machine and considering the sequence-dependent setup time of the job sequence for minimizing total setup time.

For more explanation of the baseline scheduling method, **Error! Not a valid bookmark self-reference.** shows the example schedule created by the baseline scheduling method. Example. Consider the baseline schedule in which the number of jobs = 10. The steps of the baseline scheduling method can be presented as follows:

First, Generate a sequence of the job based on ascending order of due date (Earliest due date: EDD).

For example, D1 : The job is only J1 with the release times r_j = 1,200, D2 : The set of jobs is [J2,J3,J4,J5,J6] with the release times r_j = [1,390, 970, 1,150, 1,350, 1,810] and D3 : The set of jobs is [J7,J8,J9,J10] with the release times r_j = [1,890, 2,180, 2,520, 2,550]. After sorting the set of unscheduled jobs in ascending order based on their due dates. J0 is the one job on the earliest due date of D1 among all the jobs. So, the first job of the sequence is J0 and the sequencing procedure on D1 is complete.

Table 5. The baseline scheduling method.

Creating schedule by the baseline scheduling method
Set of jobs = a list of jobs sorted by ascending order of due date d_j based on earliest due date: EDD
for all job j in Set of jobs
for all job j in set of common due dates d_j
Generate a sequence of the job based on ascending order of release time r_j
endfor
endfor
for all job j in Set of jobs
for all job j in set of common due dates dj
Add setup time of job j to set of setup time without duplicates
for Setup time in set of setup time
If Setup time of job j = Setup time in set of setup time
Grouping job
endif
endfor
endfor
endfor

Then, sort the next set of unscheduled jobs on D2 by release times in ascending order. A sequence of jobs on D2 is [J3,J4,J5,J2,J6]. Afterward, consider the setup time depends on the sequence of jobs on D2 by setup codes that operations with the same white color belong to the job.

.

First, sequencing procedure is referred to as priority rules for sequencing or dispatching jobs to a single machine. The manufacturer often only needs to decide the scheduling of jobs when making production plans, ignoring the impact of many factors on production. They are assigned the smallest due dates in the earliest due date rule (EDD) order. The earliest due date rule (EDD) is widely used as a heuristic approach for production schedules because of its simplicity. However, this approach has the limitation that the rules with the best performance differ according to the shop conditions [33]. EDD orders the jobs in the order of increasing due dates for single-machine scheduling problems. So, a job that is due first is started used when companies are sensitive to due date changes. Then, sorting procedure is sorting release time jobs on a common due date by ascending. Jobs are arranged from lowest to highest release time. Subsequently, the time required to perform a setup activity is called setup time. If the required time for the setup activities can change according to the processing sequence of jobs, setup times become sequence-dependent [34].

Therefore, sorting setup time for all of the jobs included in the group of common due dates is sequence-dependent. Eventually, assigning procedure is a sequence of job assignments to a single machine and considering the sequence-dependent setup time of the job sequence for minimizing total setup time.

For more explanation of the baseline scheduling method, **Error! Not a valid bookmark self-reference.** shows the example schedule created by the baseline scheduling method. Example. Consider the baseline schedule in which the number of jobs = 10. The steps of the baseline scheduling method can be presented as follows:

First, Generate a sequence of the job based on ascending order of due date (Earliest due date: EDD).

For example, D1 : The job is only J1 with the release times r_j = 1,200, D2 : The set of jobs is [J2,J3,J4,J5,J6] with the release times r_j = [1,390, 970, 1,150, 1,350, 1,810] and D3 : The set of jobs is [J7,J8,J9,J10] with the release times r_j = [1,890, 2,180, 2,520, 2,550]. After sorting the set of unscheduled jobs in ascending order based on their due dates. J0 is the one job on the earliest due date of D1 among all the jobs. So, the first job of the sequence is J0 and the sequencing procedure on D1 is complete.

Table 5. The baseline scheduling method.

Creating schedule by the baseline scheduling method
Set of jobs = a list of jobs sorted by ascending order of due date d_j based on earliest due date: EDD
for all job j in Set of jobs
for all job j in set of common due dates d_j
Generate a sequence of the job based on ascending order of release time r_j
endfor
endfor
for all job j in Set of jobs
for all job j in set of common due dates dj
Add setup time of job j to set of setup time without duplicates
for Setup time in set of setup time
If Setup time of job j = Setup time in set of setup time
Grouping job
endif
endfor
endfor
endfor

Then, sort the next set of unscheduled jobs on D2 by release times in ascending order. A sequence of jobs on D2 is [J3,J4,J5,J2,J6]. Afterward, consider the setup time depends on the sequence of jobs on D2 by setup codes that operations with the same white color belong to the job.

While setup time is represented in various colors : [yellow, orange, blue, yellow, yellow] then set of setup time without duplicates : [yellow, orange, blue]. Next, grouping jobs in order of set of setup time without duplicates. So, the sequence of jobs on D2 is [J3,J2,J6,J4,J5] with the sequence of setup codes : [yellow, yellow, yellow, orange, blue].

The step was repeated for a sequence of jobs on D3 in the same way as on D2. The whole job is scheduled to be completed by the baseline scheduling method.

5.4. Computational Experiment Results

In this section, the quality of the proposed scheduling model was tested with a CPU time limit of 10 seconds allocated on randomly generated instances by comparing the scheduling solution with the baseline scheduling method. A set of 1,000 different instances with a larger number of jobs is considered to show the potential of the scheduling model.

A comparison of results shows the performance of the proposed scheduling model and the baseline scheduling method in the same instances.

Due to the difference in the instance sizes,

Table 6 shows that the baseline scheduling method is not capable of solving all instances; it can solve some instances but not all. The more it decreases when the number of jobs increases. While the proposed scheduling model is capable of solving numerous instances, with its efficiency gradually decreasing as the number of jobs increases.

A comparison of results shows a greater number of solved instances in the proposed scheduling model than the baseline scheduling method (more unsolved problems in the baseline scheduling method).

Table 6. Number of solved instances.

Number of jobs	**Number of solved instances**	
	The baseline scheduling method	**The proposed scheduling model**
5	990	1,000
10	691	869
15	466	731
20	412	705
25	380	665
30	359	633
35	318	626

Fig. 4 shows the possible results on the types of solutions include optimal solutions, feasible solutions, and no solutions. The proposed scheduling model is capable of solving a greater number of optimal solutions compared to the baseline scheduling method.

Besides, The proposed scheduling model finds solutions that are not only feasible but also best solutions in terms of the objective.

For every instances that model tries to solve. When focusing exclusively on solutions found. The baseline scheduling method finds the optimal solution with a lower percentage average than the proposed scheduling model.

Table 7. %Optimal solution found by the baseline scheduling method.

		The baseline scheduling method			
		Number of solutions found			%Optimal solution found
Number of jobs	Total number of solutions found	Optimal solutions	Feasible solutions	No solutions	
5	1,000	648	342	10	64.80%
10	869	330	361	309	37.97%
15	731	106	360	534	14.50%
20	705	67	345	588	9.50%
25	665	44	336	620	6.62%
30	633	50	309	641	7.90%
35	626	31	287	682	4.95%

Table 8. %Optimal solution found by the proposed scheduling model.

		The proposed scheduling model			
		Number of solutions found			%Optimal solution found
Number of jobs	Total number of solutions found	Optimal solutions	Best solutions	No solutions	
5	1,000	1,000	-	-	100%
10	869	869	-	131	100%
15	731	731	-	269	100%
20	705	547	158	295	77.59%
25	665	310	355	335	46.62%
30	633	256	377	367	40.44%
35	626	285	341	374	45.53%

DOI:10.4186/ej.2024.28.11.81

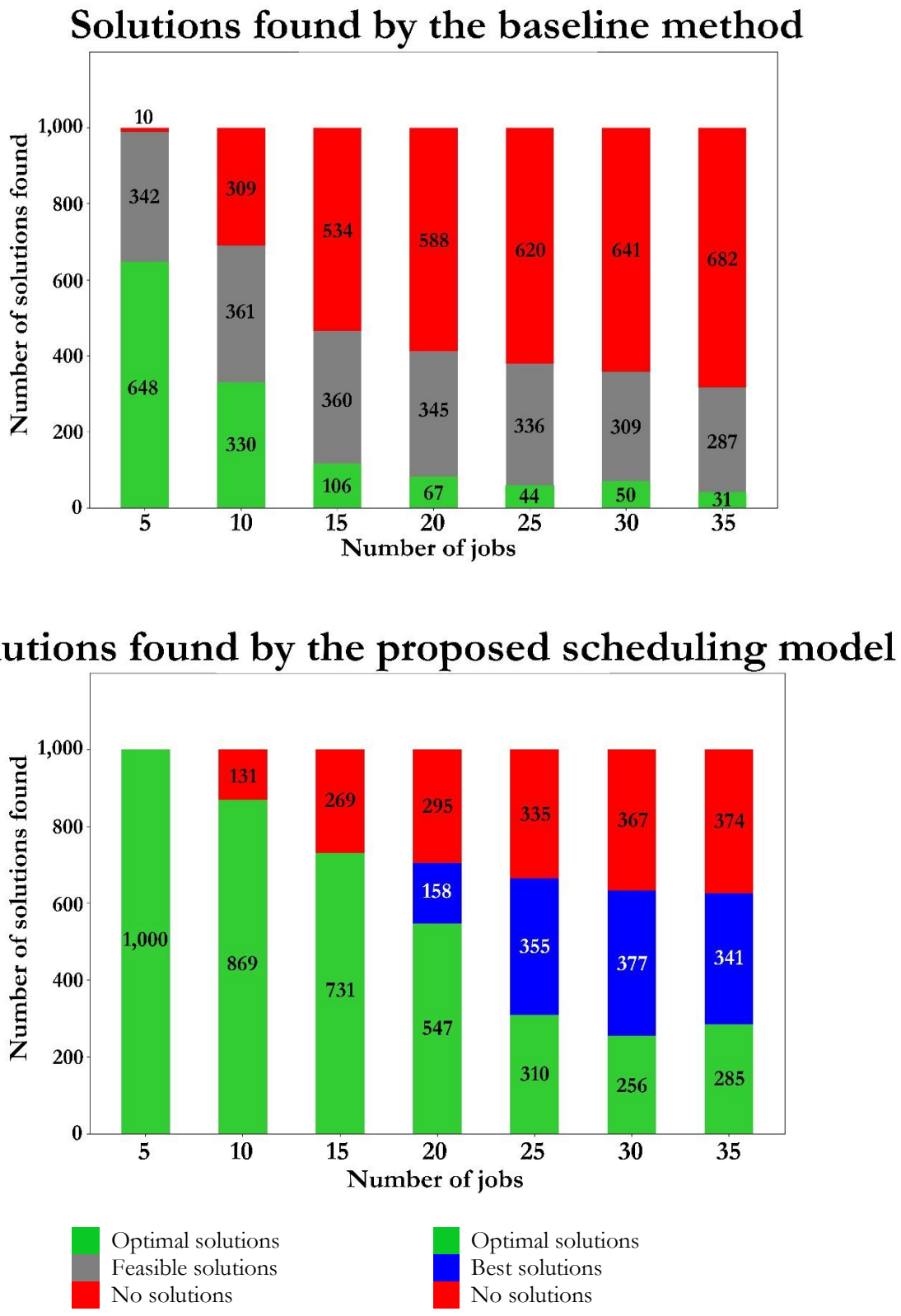

Fig. 4. Comparison of solutions found by (1) the baseline scheduling method compared with; (2) the proposed scheduling model.

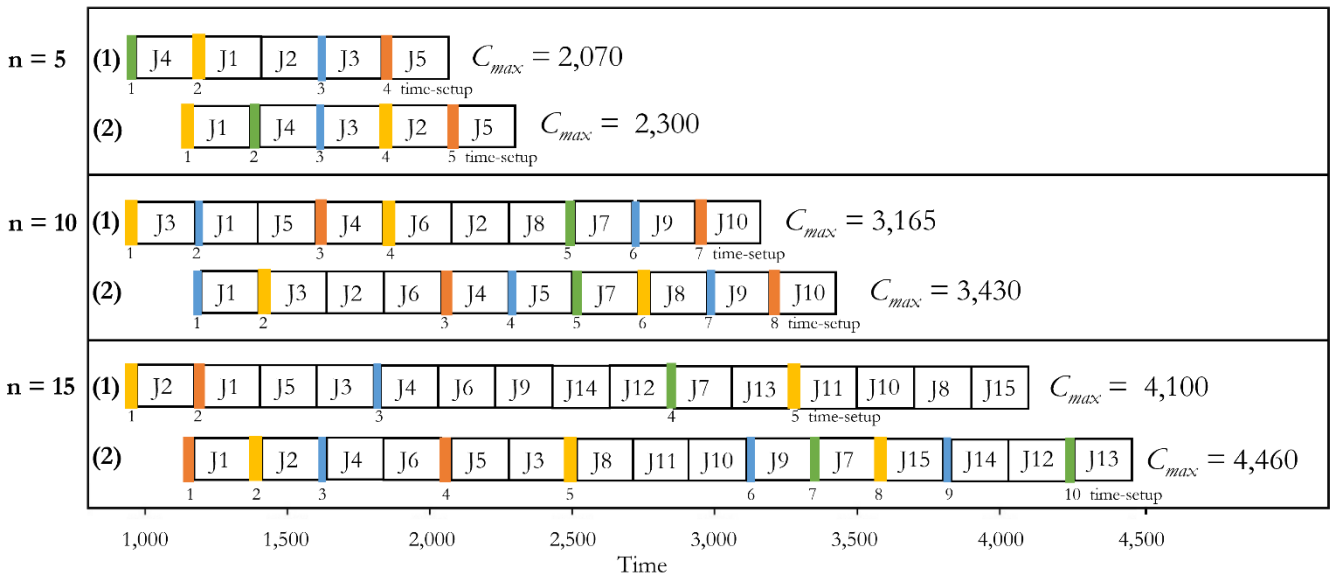

Fig. 5. Sample results in gantt diagram of (1) the proposed scheduling model compared with; (2) the baseline scheduling method.

Table 7 and Table 8 shows that the baseline scheduling method has a percentage average in finding the optimal solution in the number of jobs with values 5, 10, 15, 20, 25, 30 and 35 are 64.80%, 37.97%, 14.50%, 9.50%, 6.62%, 7.90% and 4.95% compare with a percentage average in finding the optimal solution by the proposed scheduling model in the number of jobs with values 5,10,15,20,25,30 and 35 are 100%, 100%, 100%, 77.59%, 46.62%, 40.44%, 45.53%.

The classification of problems into small-size instances and large-size instances depends on %Optimal solution found. The proposed scheduling model is able to find optimal solutions with 100 percentage average for the number of jobs with values 5,10, and 15 , it implies that these are considered small-size instances. Therefore, the number of jobs with values 20,25,30 and 35 are classified as large-size instances implying that the proposed scheduling model includes a greater number of decision variables and constraints. The experiments conducted for the small and larger number of job size instances are presented in 5.4.1 and 5.4.2.

5.4.1. Experiments with small-size instances

Three different values for a number of jobs with values 5, 10, and 15 were considered for small-size instances. Fig. 5 presents the same data but with a switched methodology to schedule to highlight the better performance of makespan. When jobs are available for processing at different release times. The proposed scheduling model uses the constraint programming (CP) to solve instances by taking constraints as input and then searching for solutions involves satisfying all the constraints simultaneously. In contrast, the baseline method is designed to solve a problem sequentially, following the steps of the procedure that focuses on due date constraint first to avoid delays that cause several complications afterward and considers other constraints after an arrangement of due date.

The baseline scheduling method prioritizes jobs based on their due dates. Jobs with the earliest due dates are scheduled first. If jobs are not available on time, a decision in the baseline scheduling method has to wait for jobs to be released. Compared with the proposed scheduling model, sequencing jobs based on their due dates might not be necessary. When jobs are available on time, If sequencing of jobs does not lead to delays in the completion of other jobs, then it will be sequenced.

This is the reason why its results for the proposed scheduling model are better than the baseline scheduling method.

Experiment results show that the solution found by the proposed scheduling model. There are three categories of an average improvement in makespan.

1. The proposed scheduling model is capable of finding a solution for a number of jobs with values 5, 10, and 15 with 10, 178, 265 instances, whereas the baseline scheduling method lacks this ability.

2. The proposed scheduling model outperforms the baseline scheduling method in finding the optimal solutions. Due to the difference in instance sizes, the values of makespan vary wildly between instances. Therefore, experimental tests are executed to compare the values of makespan. Since the 5,10 and 15-job instances of the proposed scheduling model are all solved to optimality in a short time. An improvement of the makespan measured are 3.4555%, 3.0731%, and 4.8826% are shown in Table 9 .

3. The proposed scheduling model and the baseline scheduling method find solutions of equal value of makespan for a number of jobs with values 5, 10, and 15 with 648, 330, 106 instances. This result indicates that the

baseline scheduling method has the potential to be compared to the proposed scheduling model.

Table 9 clearly shows that the proposed scheduling model outperformed the baseline scheduling method. Improvement results indicated that the proposed scheduling model performs better than the baseline scheduling method with respect to the considered performance measures.

Table 9. Summary of average results for optimal solutions solved by the proposed scheduling model on small-size instances compared with the baseline scheduling method.

Number of jobs	Total number of optimal solutions	Number of optimal solutions	Improve-ment of average makespan
5	1,000	342	3.4555%
10	869	361	3.0731%
15	731	360	4.8826%

Furthermore, an appealing aspect to consider is setup time are visualized in Fig. 5. The setup time occurs when jobs are processed and a setup of a time precedes the processing of jobs. It is sequence-dependent if its duration depends on the jobs of both the current and the immediately preceding jobs, and is sequence-independent if its duration depends solely on the current jobs to be processed. No job processing is possible on a machine while a setup is being performed on the machine. If the sequence of the jobs are the same setup time. All the jobs of the same setup time become available for processing and leave the machine together. Start time of the first job is scheduled by the scheduling model starts before start time of the first job is scheduled by the baseline schedule method and fewer changeover times after that at the end of the job sequencing gives a smaller sum of completion time. Sequence of jobs may lead to events that some jobs delay. If a job is delivered after the due date, customer dissatisfaction might lead to a penalty for the company and the possibility of reputational damage [14].

The proposed scheduling model has the ability to reduce the total setup time. Fig. 5 shows the fact that there are tremendous savings when setup times are explicitly incorporated in scheduling decisions in the manufacturing environment. The benefits of the proposed scheduling model involving combine setup times. The setup time between jobs involves the changing of colors, which indicates the significance of setup times.

Appropriate sequencing of the proposed scheduling model leads to a potential for savings in setups from maximum 10 time-setup to 5 time-setup. The complete schedule determined by the proposed scheduling model resulted in a smaller makespan than the makespan determined by the baseline scheduling method. The proposed scheduling model provides a better solution to sequence jobs when facing several constraints.

In conclusion, Table 10 clearly demonstrates the effectiveness of the proposed scheduling model in reducing the total setup time for the number of jobs with values 5, 10 and 15. The total setup time has been decreased by 6.4419%, 26.0663% and 45.7924%.
Summary of average total setup time improvement results for optimal solutions solved by the proposed scheduling model on small-size instances compared with the baseline scheduling method

Table 10. Summary of average total setup time improvement results for optimal solutions solved by the proposed scheduling model on small-size instances compared with the baseline scheduling method.

Number of jobs	Total number of optimal solutions	Improvements of average total setup time
5	1,000	6.4419%
10	869	26.0663%
15	731	45.7924%

5.4.2. Experiments with larger number of jobs instances

Four different values for number of jobs with values 20, 25, 30 and 35 were considered for a larger number of jobs instances. In the 1,000 different larger number of jobs instances, the proposed scheduling model could not prove the optimality of all instances in a short time. Considering the scheduling model, testing shows that the proposed scheduling model using CP model is not very usable for larger instances.

Therefore, a time limitation has been set for solving. A time limitation was imposed 10 seconds. However, the proposed scheduling model outperformed the baseline scheduling method.

These experiments conclude that the proposed scheduling model offers a very good quality solution despite a time limitation.

Experiments show that the solution found by the proposed scheduling model. There are three categories of an average improvement in makespan.

1. The proposed scheduling model is capable of finding a solution for a number of jobs with values 20, 25, 30 and 35 with 250, 151, 107, 143 instances, whereas the baseline scheduling method lacks this ability.

2. The proposed scheduling model outperforms the baseline scheduling method in finding the optimal solutions. Due to the difference in instance sizes, the values of makespan vary wildly between instances. Therefore, experimental tests are executed to compare the values of makespan. Since the 20,25,30 and 35-job instances of the proposed scheduling model are all solved to optimality in a short time. An improvement of the makespan measured are 6.5654%, 7.4891%, 6.9290% and 7.2210% are shown in Table 11.

3. The proposed scheduling model and the baseline scheduling method find solutions of equal value of makespan for a number of jobs with values 20, 25, 30 and

35 with 67, 44, 50, 31 instances. This result indicates that the baseline scheduling method has the potential to be compared to the proposed scheduling model.

Table 11. Summary of average results for optimal solutions solved by the proposed scheduling model on larger number of jobs instances compared with the baseline scheduling method

Number of jobs	Total number of optimal solutions	Number of optimal solutions	Improve-ment of average makespan
20	547	230	6.5654%
25	310	115	7.4891%
30	256	99	6.9290%
35	285	111	7.2210%

A common scenario in manufacturing, where the setup time increases with the number of jobs. Multiple jobs, Different types require different setup time, Between these jobs, changeover must be altered. The more job changes there are, the more cumulative setup time will accrue.

In conclusion, Table 12 clearly demonstrates the effectiveness of the proposed scheduling model in reducing the total setup time for the number of jobs with values 20, 25, 30 and 35. The total setup time has been decreased by 54.1047%, 53.4973%, 46.6404%, 55.4033%.

Table 12. Summary of average total setup time improvement results for optimal solutions solved by the proposed scheduling model on larger number of jobs instances compared with the baseline scheduling method

Number of jobs	Total number of optimal solutions	Improvement of average total setup time
20	547	54.1047%
25	310	53.4973%
30	256	46.6404%
35	285	55.4033%

6. Conclusion and Future Work

This paper focused on developing the scheduling model for sequencing a set of jobs with different release times in a single machine that minimizes the makespan to meet different due dates and to reduce setup time where setup times are sequence-dependent. In conclusion, The proposed scheduling model is effective, relevant, and can be used for the problem with better efficiency than the baseline scheduling method. This paper's most significant contribution is applying the proposed model to real-world production scenarios. Initially, the proposed scheduling model using constraint programming was proposed to optimally solve the problem. The benefit of using the proposed scheduling model is significant as it can generate optimal solution on small-size instances and best solutions on a larger number of jobs instances. However, the constraint programming model is suitable only for specific instance sizes. On a larger number of jobs instances can provide the best solutions but computation time increases when the number of jobs increases. The constraint programming alone could not handle large-sized instances and improved slower, but it guarantees improvement until an optimal solution is found. When dealing with large-sized instances, It's important to noticed that there is usually a trade-off between solution accuracy and computation time. Depending on the proposed scheduling model, it might be more beneficial to get a near-optimal solution quickly than to spend a significantly longer time aiming for the optimal solutions. Future studies will focus on the development of heuristics to deal with the considered problem to find accurate solutions more quickly. Furthermore, the solution approach is fast enough to be used in practical scenarios where larger instances must be solved within a few minutes and can easily be adapted to be used for different scenarios.

Acknowledgement

This research was supported by the 100th Anniversary Chulalongkorn University Fund for Doctoral Scholarship and the 90th Anniversary Chulalongkorn University Fund (Ratchadaphiseksomphot Endowment Fund).

References

s

[1] J. Jungwattanakit, M. Reodecha, P. Chaovalitwongse, and F. Werner, "A comparison of scheduling algorithms for flexible flow shop problems with unrelated parallel machines, setup times, and dual criteria," *Computers & Operations Research,* vol. 36, no. 2, pp. 358-378, 2009, doi: 10.1016/j.cor.2007.10.004.

[2] D. Lei and J. Cai, "Multi-population meta-heuristics for production scheduling: A survey," *Swarm and Evolutionary Computation,* vol. 58, p. 100739, 2020, doi: 10.1016/j.swevo.2020.100739.

[3] S. H. Choi and K. Wang, "Flexible flow shop scheduling with stochastic processing times: A decomposition-based approach," *Computers & Industrial Engineering,* vol. 63, no. 2, pp. 362-373, 2012, doi: 10.1016/j.cie.2012.04.001.

[4] Y. Zhang and F. Tao, "Real-time information-driven production scheduling system," in *Optimization of Manufacturing Systems Using the Internet of Things*, Y. Zhang and F. Tao, Eds. Academic Press, 2017, ch. 8, pp. 147-163.

[5] P. Priore, A. Gómez, R. Pino, and R. Rosillo, "Dynamic scheduling of manufacturing systems using machine learning: An updated review," *Artificial Intelligence for Engineering Design, Analysis and Manufacturing,* vol. 28, no. 1, pp. 83-97, 2014, doi: 10.1017/S0890060413000516.

[6] R. Martinelli, F. Cristina Martins Queiroz Mariano, and C. Bertini Martins, "Single machine scheduling in make to order environments: A systematic review," *Computers & Industrial Engineering,* vol. 169, p. 108190, 2022, doi: 10.1016/j.cie.2022.108190.

[7] F. Yue, S. Song, Y. Zhang, J. N. D. Gupta, and R. Chiong, "Robust single machine scheduling with uncertain release times for minimising the maximum waiting time," *International Journal of Production Research,* vol. 56, no. 16, pp. 5576-5592, 2018, doi: 10.1080/00207543.2018.1463473.

[8] V. Fernandez-Viagas and A. Costa, "Two novel population based algorithms for the single machine scheduling problem with sequence dependent setup times and release times," *Swarm and Evolutionary Computation,* vol. 63, p. 100869, 2021/06/01/ 2021, doi: 10.1016/j.swevo.2021.100869.

[9] X. Wang, T. Ren, D. Bai, C. Ezeh, H. Zhang, and Z. Dong, "Minimizing the sum of makespan on multi-agent single-machine scheduling with release dates," *Swarm and Evolutionary Computation,* vol. 69, p. 100996, 2022, doi: 10.1016/j.swevo.2021.100996.

[10] R. Patikarnmonthon and M. Lohatepanont, "Production scheduling with capacity-lot size and sequence consideration," *Engineering Journal,* vol. 22, no. 1, pp. 93-107, 2018.

[11] F. Angel-Bello, J. Vallikavungal, and A. Alvarez, "Fast and efficient algorithms to handle the dynamism in a single machine scheduling problem with sequence-dependent setup times," *Computers & Industrial Engineering,* vol. 152, p. 106984, 2021, doi: 10.1016/j.cie.2020.106984.

[12] G. Kirlik and C. Oguz, "A variable neighborhood search for minimizing total weighted tardiness with sequence dependent setup times on a single machine," *Computers & Operations Research,* vol. 39, no. 7, pp. 1506-1520, 2012/07/01/ 2012, doi: 10.1016/j.cor.2011.08.022.

[13] C. Koulamas and G. J. Kyparisis, "A classification of dynamic programming formulations for offline deterministic single-machine scheduling problems," *European Journal of Operational Research,* vol. 305, no. 3, pp. 999-1017, 2023, doi: 10.1016/j.ejor.2022.03.043.

[14] D. Shabtay, "Optimal restricted due date assignment in scheduling," *European Journal of Operational Research,* vol. 252, no. 1, pp. 79-89, 2016, doi: 10.1016/j.ejor.2015.12.043.

[15] C. Low, R.-K. Li, G.-H. Wu, and C.-L. Huang, "Minimizing the sum of absolute deviations under a common due date for a single-machine scheduling problem with availability constraints," *Journal of Industrial and Production Engineering,* vol. 32, no. 3, pp. 204-217, 2015, doi: 10.1080/21681015.2015.1031196.

[16] C. N. Rosyidi, S. N. Hapsari, and W. A. Jauhari, "An integrated optimization model of production plan in a large steel manufacturing company," *Journal of Industrial and Production Engineering,* vol. 38, no. 3, pp. 186-196, 2021, doi: 10.1080/21681015.2021.1882592.

[17] Y. Jaegler, A. Jaegler, F. Z. Mhada, D. Trentesaux, and P. Burlat, "A new methodological support for control and optimization of manufacturing systems in the context of product customization," *Journal of Industrial and Production Engineering,* vol. 38, no. 5, pp. 341-355, 2021, doi: 10.1080/21681015.2021.1891147.

[18] Y.-W. Liu, L.-L. Li, M.-L. Tseng, M. K. Lim, and M. Helmi Ali, "Optimal scheduling of combined cooling, heating, and power microgrid based on a hybrid gray wolf optimizer," *Journal of Industrial and Production Engineering,* vol. 39, no. 4, pp. 277-292, 2022, doi: 10.1080/21681015.2021.1974963.

[19] Y.-C. Huang, Y.-A. Ding, and X.-Y. Chang, "An optimal production scheduling under random selection within multi-objective based on batch movement and asymmetrical setup time," *Journal of Industrial and Production Engineering,* vol. 35, no. 8, pp. 506-525, 2018/11/17 2018, doi: 10.1080/21681015.2018.1520746.

[20] M. D. Toksarı and G. Toğa, "Single batch processing machine scheduling with sequence-dependent setup times and multi-material parts in additive manufacturing," *CIRP Journal of Manufacturing Science and Technology,* vol. 37, pp. 302-311, 2022, doi: 10.1016/j.cirpj.2022.02.007.

[21] F. Yang, M. Davari, W. Wei, B. Hermans, and R. Leus, "Scheduling a single parallel-batching machine with non-identical job sizes and incompatible job families," *European Journal of Operational Research,* vol. 303, no. 2, pp. 602-615, 2022, doi: 10.1016/j.ejor.2022.03.027.

[22] H. Badri, T. Bahreini, and D. Grosu, "A parallel randomized approximation algorithm for non-preemptive single machine scheduling with release dates and delivery times," *Computers & Operations Research,* vol. 130, p. 105238, 2021, doi: 10.1016/j.cor.2021.105238.

[23] E. Molaee, R. Sadeghian, and P. Fattahi, "Minimizing maximum tardiness on a single machine with family setup times and machine disruption," *Computers & Operations Research,* vol. 129, p. 105231, 2021, doi: 10.1016/j.cor.2021.105231.

[24] W. T. Lunardi, E. G. Birgin, P. Laborie, D. P. Ronconi, and H. Voos, "Mixed Integer linear programming and constraint programming models for the online printing shop scheduling problem," *Computers & Operations Research,* vol. 123, p. 105020, 2020, doi: 10.1016/j.cor.2020.105020.

[25] O. Herr and A. Goel, "Minimising total tardiness for a single machine scheduling problem with family setups and resource constraints," *European Journal of Operational Research,* vol. 248, no. 1, pp. 123-135, 2016, doi: 10.1016/j.ejor.2015.07.001.

[26] S. Kır and H. R. Yazgan, "A sequence dependent single machine scheduling problem with fuzzy axiomatic design for the penalty costs," *Computers &*

Industrial Engineering, vol. 92, pp. 95-104, 2016, doi: 10.1016/j.cie.2015.12.012.

[27] A. Karray, M. Benrejeb, and P. Borne, "A hybrid algorithm to solve the single-machine scheduling problem," *International Journal of Innovation and Applied Studies,* vol. 11, p. 623, 2015.

[28] S. Asadamongkol and B. Eua-arporn, "Multistage Transmission Expansion Planning Using Local Branching Method," *Eng. J.,* vol. 14, no. 4, pp. 23-40, 2010.

[29] F. Riane, A. Artiba, and S. E. Elmaghraby, "A hybrid three-stage flowshop problem: Efficient heuristics to minimize makespan," *European Journal of Operational Research,* vol. 109, no. 2, pp. 321-329, 1998, doi: 10.1016/S0377-2217(98)00060-5.

[30] F. Grandoni and G. Italiano, *Algorithms and Constraint Programming.* 2006, pp. 2-14.

[31] Y. Zhang, Z. Zhang, Y. Zeng, and T. Wu, "Constraint programming for multi-line parallel partial disassembly line balancing problem with optional common stations," *Applied Mathematical Modelling,* vol. 122, pp. 435-455, 2023, doi: 10.1016/j.apm.2023.06.009.

[32] V. Heinz, A. Novák, M. Vlk, and Z. Hanzálek, "Constraint Programming and constructive heuristics for parallel machine scheduling with sequence-dependent setups and common servers," *Computers & Industrial Engineering,* p. 108586, 2022, doi: 10.1016/j.cie.2022.108586.

[33] T. Kim, Y.-W. Kim, D. Lee, and M. Kim, "Reinforcement learning approach to scheduling of precast concrete production," *Journal of Cleaner Production,* vol. 336, p. 130419, 2022, doi: 10.1016/j.jclepro.2022.130419.

[34] Z. A. Çil, D. Kizilay, Z. Li, and H. Öztop, "Two-sided disassembly line balancing problem with sequence-dependent setup time: A constraint programming model and artificial bee colony algorithm," *Expert Systems with Applications,* vol. 203, p. 117529, 2022, doi: 10.1016/j.eswa.2022.117529.

Manlika Kiatthadasirikul
Thailand
Diploma (Communication Maintenance), Civil Aviation Training Center
B.Eng. (Logistics Engineering), Mahanakorn University of Technology
M.Eng. (Industrial Engineering), Chulalongkorn University
Currently pursuing a Ph.D. in Industrial Engineering at Chulalongkorn University

Paveena Chaovalitwongse
Thailand
B.Eng. (Industrial Engineering), Chulalongkorn University
M.Eng. (Industrial and Systems Engineering), University of Florida
Ph.D. (Industrial and Systems Engineering), University of Florida

Naragain Phumchusri
Thailand
B.Eng. (Industrial Engineering), Chulalongkorn University
M.Sc. (Industrial Engineering), Georgia Institute of Technology
Ph.D. (Industrial Engineering), Georgia Institute of Technology

DOI:10.4186/ej.2024.28.11.81

Siravit Swangnop
Thailand
B.Eng. (Industrial Engineering), Chulalongkorn University
M.Eng. (Industrial Engineering), Chulalongkorn University
Ph.D. (Industrial Engineering), Chulalongkorn University