Opsi p-ISSN 1693-2102
Vol. 17, No. 2, December 2024 e-ISSN 2686-2352

Improving the job shop scheduling algorithm to minimize total penalty costs considering maintenance activity

Puryani [1], Nurmalia Chalida [2], Apriani Soepardi [1*], Mochammad Chaeron [1], Laila Nafisah [1]
[1]Industrial Engineering Department, Universitas Pembangunan Nasional Veteran Yogyakarta, Babarsari 2 Tambakbayan Yogyakarta 55281 Indonesia
[2]CV Surya Mitra Utama, Jl. Lapangan Sadang 46, Taman, Sidoarjo61257 Jawa Timur

*Corresponding Author: apriani.soepardi@upnyk.ac.id; Tel.: +62274-485268

Article history:
Received: 15 May 2024
Revised: 12 August 2024
Accepted: 30 December 2024
Published: 31 December 2024

Keywords:
Scheduling
Job shop
Earliness
Tardiness
Penalty cost

ABSTRACT
Production scheduling is generally based on the assumption that resources are always available. In reality, these resources, machines, and supporting facilities experience limited availability due to interruptions during the production process. Therefore, to improve these conditions, the maintenance process conducted to reduce the disruption level of the machine needs to be scheduled as part of the available for job processing leading to penalty costs, such as tardiness and earliness. This research aims to develop a new algorithm to solve job shop scheduling problems to minimize the total penalty cost by considering machine unavailability due to scheduled maintenance activities. The proposed model modifies the existing model using a combination of priority rules and a heuristic approach algorithm known as priority dispatching. The result showed that the proposed model produces a greater total cost with a larger flow time than the previous model. Although the flow time is larger, it is more realistic according to real conditions because the proposed model considers machine maintenance activities. Furthermore, the combination of priority rules used also affected the flow time and the total penalty costs incurred, which can be minimized through several alternatives.

DOI:
https://doi.org/10.31315/opsi.v17i2.12291

1. INTRODUCTION

Scheduling is a plan for arranging job sequences and allocating resources, both time and facilities, to each operation that must be completed [1]. The essence of the scheduling objective function is to minimize the total processing time (makespan). In addition, scheduling based on machine environment is divided into several types. These include single and multiple machines, flow, job, and open shops [2–4]. Li [5], Sivrikaya-Serifoglu & Ulusoy [6], and Dorndoft et al. [7] developed a production scheduling optimization algorithm that can handle NP-Hard problems while taking time and resource constraints into account. They employed branch and bound algorithms and genetic algorithms to manage the complexity, aiming to minimize penalty costs and delays. We widely use heuristic and metaheuristic algorithms for scheduling because they can overcome the limitations of mathematical models or exact algorithms that are too slow for large scales [8–10]. Some studies use a combination of exact algorithms and heuristics to solve complex scheduling problems with the aim of optimizing time or cost criteria [11–13].

A job shop is a manufacturing environment where new ones often have different work routes or operations [2–3]. Job shop scheduling is characterized by scheduling *n* jobs on *m* machines, each comprising an unidentical machine sequence or routing. The simple form of this model assumes that each job only passes through a machine once on its route to the process. However, other models allow each job to pass through similar machines more than once on its route.

Production scheduling is generally based on the assumption that resources are always available. However, in reality, these resources, including machines, equipment, and facilities that support the production process, have limited availability of resources [14–16]. For example, when the machine is interrupted during the production process. These conditions are improved by scheduling and accomplishing maintenance activities to reduce the disturbance level on the machine. This causes the job process to take a long time, resulting in penalties such as tardiness and earliness costs. The earliness penalty occurs when the work is completed before the specified time limit, thereby saving costs [17]. Meanwhile, tardiness occurs because the work was not completed within the predetermined time limit, which leads to a penalty [10–12]. Pham & Klinkert [18] and Hsu [19] emphasize makespan reduction as the main priority in scheduling, even though it is applied in different sectors, namely healthcare services and the manufacturing industry.

The issue of scheduling several jobs on a single machine was developed by Li [5] and parallel machine by Sivrikaya-Şerifoğlu & Ulusoy [6] to minimize the sum of earliness and tardiness costs. Chen [16] researched a single machine that addressed the issue of machine unavailability in scheduling. The investigation entailed scheduling problems by several time intervals as a maintenance activity. It is designed for a single machine and focuses on improving the periodic maintenance schedule. Andriani [23] developed a job shop scheduling model to minimize penalty costs without considering disruptions to the production process. Therefore, the proposed model is designed based on this model by considering the maintenance activities schedule.

2. LITERATURE REVIEW

The *earliness* penalty occurs when the work is completed before the specified time limit, thereby saving costs. Meanwhile, *tardiness* occurs because the work was not completed within the predetermined time limit, which leads to a penalty. Baker and Scudder [1] formulated the *earliness* and *tardiness* model as follows.

$$E_i = \max(0, d_i - C_i) = (d_i - C_i)^+ \tag{1}$$

$$T_i = \max(0, C_i - d_i) = (C_i - d_i)^+ \tag{2}$$

$$f(S) = \sum_{i=1}^{n} (\alpha E_i + \beta T_i) \tag{3}$$

where,

E_i	:	*Earliness* on job *i*
d_i	:	*Due date* on job *i*
C_i	:	*Completion time* on job *i*
T_i	:	*Tardiness* on job *i*
α	:	*Earliness* penalty unit cost
β	:	*Tardiness* penalty unit cost
$f(s)$	:	Function of on *S* schedule

Andriani [23] developed a job shop scheduling using the priority dispatching algorithm. The algorithm used a forward and backward-forward scheduling approach to minimize the total earliness and tardiness costs. It is assumed that no other activities can interrupt the production process. The notations used in this model will be employed to develop a proposed algorithm and they are presented below.

St	:	a collection of tasks that are ready to be scheduled at step *t* (iteration step)
P_{st}	:	a partial schedule that contains scheduled operations
C_j	:	completion time of the *j*-th operation
R_{ij}	:	start time of the *i*-th job and the *j*-th operation
R^*	:	the fastest time an operation can be started ($R^* = Cj + t_{ij}$)

m^*	:	machine where R^* can be realized
Noj	:	the next operation of the job
Nom	:	the next operation of the machine
jobStart	:	operation start time which is constrained by the previous operation of the job
mcStart	:	operation start time which is limited to the previous operation of the machine
startTime	:	operation start time of the old schedule
endTime	:	operation finish time of the old schedule
newStart	:	operation start time of the new schedule
newEnd	:	operation finish time of the new schedule
devSt	:	deviation of the start time between the old and new schedules
Aff	:	set of operations affected after rescheduling with new start and finish times
O	:	affected set of operations
I	:	index of *aff*
G	:	index of *O*

The calculation stages of this model realized with the forward approach, are described in the following algorithm.

Step 1: Initiation Stage
Identification of the number and routing of jobs.

Step 2: Set $t = 1$, $C_j = 0$, and $P_{st} = 0$ (P_{st} is a partial schedule containing scheduled operations).
Set S_t (S_t is the set of operations ready to be scheduled) is equal to all operations without predecessors.

Step 3: For each operation in S_t that requires machine m where R_{ij} is performed, select *task ij.*
In model 1: Select task *ij* with the largest remaining processing time, taking into account the effect of t_{ij} on *task ij* (LPTR).
In model 2: Select *task ij* with the largest remaining processing time without considering the effect of t_{ij} on *task ij* (LPTR+1).

Step 4: The selected *task ij* will have $R_{ij} = R^*$ where R^* equals m^*.

a. Is *task ij* > 1?
When selecting whether *task ij* contains S_t and m^* where R^* is greater than one?
If <u>not</u>, then proceed to Step 5.

b. If <u>yes</u>, select the *ij* task with the Short Processing Time (SPT) priority rule, and proceed to Step 5.

c. On Step 4b, is *task ij* > 1?
When selecting Step 4b, is *task ij* in S_t and accomplished in m^* where R^* is greater than one?
If <u>not</u>, then it proceeds to Step 5.

d. If <u>yes</u>, select the *ij task* with the First Come First Serve (FCFS) priority rule, and proceed to Step 5.

Step 5: Proceed to the next step by creating a P_{st+1} partial schedule and refine the data set by:

a. Enter the selected *task ij* on $P_{st+1.}$

b. Eliminate the selected *task ij* from S_t and form S_{t+1} by adding its successor when all of its predecessor tasks have been scheduled.

c. Add *t* to 1.

d. Update the available time for each machine.

e. Fix C_j for all *task ij* in S_{t+1}, namely:

(a) For *task ij*, which is the successor of the selected task,
C_j = max (R^*, available time in machine *k*).

(b) For *task ij* that has not been selected at the previous *Step t*,
C_j = max (C_j at Step *t* before, available time in machine *k*)

Step 6: When there are still unscheduled tasks, proceed to Step 3, otherwise stop.

The following steps describe the calculation stages for the Andriani [23] algorithm, realized with the backwards-forward approach.

Step 1: Initiation Stage
Identification of the number and routing of jobs.

Step 2: Set t = 1 and P_{st} = 0 as well as determine St for each job starting from the most recent operation where R_{ij} on the last *task ij* = d_i.

Step 3: For each operation in S_t that requires machine m where R_{ij} is performed, select *task ij.*
In model 1: Select *task ij* with the largest remaining processing time taking into account t_{ij} in *task ij* (*LPTR*).
In model 2: Select *task ij* with the largest remaining processing time without taking into account t_{ij} in *task ij* (*LPTR*+1).

Step 4: The selected *task ij* will have C_j = C* where C* equals *m**.

a. Is *task ij* > 1?
At the time of selection, is *task ij* in S_t and *m** including C* greater than one?
If not, then proceed to Step 5.
b. If yes, then select the *ij* task with the Short Processing Time (SPT) priority rule, and proceed to Step 5.
c. On Step 4b, is *task ij* >1?
At the time of selecting Step 4b, is *task ij* in S_t and requires *m** where *C** is performed more than once?
If not, then proceed to Step 5.
d. If yes, then select the ij task with the First Come First Serve (FCFS) priority rule. Proceed to Step 5.

Step 5: Proceed to the next step by creating P_{st+1} partial schedule and refine the data set by:

a. Enter the selected *task ij* in $P_{st+1.}$
b. Eliminate the selected *task ij* from S_t and form S_{t+1} by adding its predecessor if all of its successor tasks have been scheduled.
c. Add t with 1.
d. Update the available time for each machine.
e. Fix R_{ij} for all *task ij* in S_{t+1}, namely:
(a) For *task ij*, which is the *predecessor* of the selected task,
R_{ij} = min (C*, available time on machine k).
(b) For *task ij* that has not been selected at the previous Step t,
R_{ij} = min (R_{ij} at Step t before, available time on machine k).

Step 6: If there are still unscheduled tasks, proceed to Step 3, then stop.

Step 7: If the scheduling results are infeasible, advance all infeasible tasks at point t = 0.

Step 8: Set i = 1, g = 1, *completion time* = 0, *devSt* = 0, O = Φ, *jobStart* = *mcStart* = 0 for all operations.

Step 9: $O[g]$ as current operation, *jobStart* = *mcStart* = 0 for infeasible job. Define:
Current newStart = max (*jobStart, mcStart*)
Current newEnd = *Current newStart* + t_{so}

Step 10: If the current job does not match the affected operating group in *aff* go to Step 11 and if not, then reset *aff* (v) to Step 12.

Step 11: Define:
newStart aff (v) = *current newStart* + (C_j the biggest infeasible job x (-1)). Proceed to Step 12.

Step 12: Calculate *newEnd aff* (v) = *newStart aff* (v) + t_{ij}

Step 13: Set *aff*[i] = current operation; i add with 1.

Step 14: Define *noj*, if yes, then:
$O[g]$ = *noj* and *jobStart* dari $O[g]$ = *current newEnd*
Current newEnd = *startTime* + (C_j the biggest infeasible job x (-1)). Proceed to Step 15.

Step 15: Calculate *newEnd aff* (v) = *current newEnd* + t_{ij}

Step 16: Define *nom*, if yes, then:
$O[g]$ = *nom* and *mcStart* from $O[g]$ = *current newEnd*
Current newEnd = *startTime* + (C_j the biggest infeasible tasks x (-1)). Proceed to Step 17.

Step 17: Calculate *newEnd aff* (*aff*) = *current newEnd* + t_{ij}

Step 18: Subtract the current operation from set O and add a new member O from Step 13.

Step 19: If *O* = Ø, hence, *stop*, if not, search new *noj* and *nom* from the current available set *O*.

Chen [16] discusses single-machine scheduling considering limited machine availability due to the periodic maintenance schedule. There is a time interval *T* between two consecutive maintenance activities. A maintenance activity requires an amount of time t for execution (Figure 1).

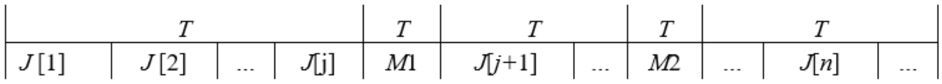

Figure 1. Scheduling with maintenance periods on a single machine [16]

Description:

J[*j*] : *j*-th job

M : maintenance time

T : time interval between two maintenance periods

t : amount of time to perform one maintenance

Furthermore, Chen [16] formulates the problem to minimize the total flow time. The total flow time of all jobs in schedule *S* is modeled as follows.

$$f(S) = \sum_{i=1}^{n} C_i \tag{4}$$

3. MODEL DEVELOPMENT

The differences among the models from Chen [16], Andriani [23], and the proposed one are shown in Table 1. Chen [16] developed a method to minimize total flow time in a scheduling system that considers periodic maintenance and non-resumable jobs. The proposed heuristic algorithm shows results close to optimal with an average error of only 0.57%. This algorithm is much faster compared to the branch and bound algorithm, making it suitable for large-scale problems. Meanwhile, Andriani [23] aimed to develop a job shop scheduling algorithm to minimize the total costs of earliness and tardiness. It modified the algorithm by providing different priority rules.

Table 1. The difference among the models

No.	Description	Chen [16]	Andriani [23]	Proposed Model
1	Machine environment	Single machine	*Job shop*	*Job shop*
2	Assumptions used	There is a schedule of maintenance activities during the production process.	No activity can interrupt the production process or activities	There is a schedule of maintenance activities during the production process

The following assumptions were employed:

1. Scheduling is conducted for processes that can be stopped (manufacturing)
2. Set-up time is not affected by the work order, therefore, it can be considered part of the processing time.
3. Transportation time is negligible
4. Job starts at *t=0* (forward approach)
5. Static job arrival pattern
6. Penalty costs are dissimilar
7. Each job has a different due date
8. The generated process determines the schedule of maintenance activities

9. Scheduled maintenance activities are simultaneously performed on each machine
10. At the time of the scheduled maintenance activity, the machine was not operating.

The proposed model modifies the basic one designed by Andriani [23]. Figure 2 shows that the proposed model is in the form of a job shop flowchart.

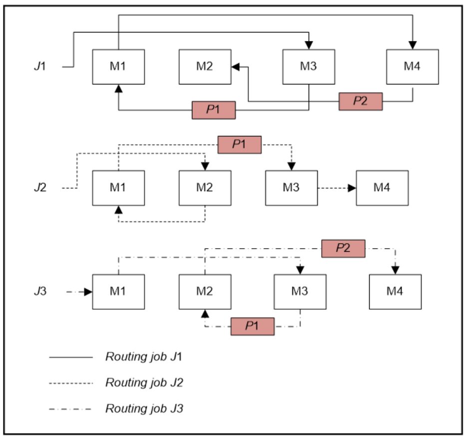

Figure 2. The job shop flowchart

The triplet notation is applied in the study, which notation consists of (*i, j, k*), namely *i* indicates the job number, *j* indicates the operation sequence, and *k* indicates the machine used. The additional notations used in the proposed algorithm are:

t : iteration step
n : number of jobs
t_{ij} : processing time of the *i*-th job and the *j*-th operation
C_{ij} : completion time of the *i*-th job and the *j*-th operation
C^* : the fastest time the operation can be completed
P_x : maintenance activities that have been scheduled by x (maintenance period, x=1, 2, …, x+1)
k : machine number
d_i : *i*-th *due date*
Irc : time interval for job execution that can be scheduled

The calculation stages of the proposed model with a forward approach are described in the following algorithm.

Step 1: Initiation stage
Identify the number and routing of jobs

Step 2: Set $t = 1$, $R_{ij} = 0$, and $P_{st} = 0$ (P_{st} is a partial schedule containing scheduled operations). Set S_t (S_t is the set of operations ready to be scheduled) is equal to all operations without *predecessor*.

Step 3: For every operation in S_t that requires an m machine where C_{ij} is performed, select *task ij*.
In model 1: Select the *task ij* with the largest remaining processing time that considers t_{ij} in the *task ij* (LPTR).
In model 2: Select *task ij* with the largest remaining processing time without considering t_{ij} in the *task ij* (LPTR+1)

Step 4:
a. If there is only one *task ij* selected, *task* ($ij = 1$), then schedule the *task,*
b. If the number of selected *tasks ij* is more than one *task* ($ij > 1$), then select *task ij* with the SPT priority rule (min t_{ij}),
c. If there is still a *task ij* that has (min t_{ij}) at the same price (min $t_{ij} > 1$), then select the *task* with FCFS priority rule.

Step 5:
a. If the *task ij* selected has $R_{ij} + t_{ij} = C_{ij} \leq P_x$ (maintenance activity start schedule), then schedule it before P_x,

b. If the *task ij* selected has $R_{ij} + t_{ij} = C_{ij} > P_x$ (maintenance activity start schedule), then schedule it after P_x,

Task ij selected will have $C_{ij} = C^*$ where C^* will have m^*.

Step 6: Proceed to the next step by creating a partial schedule P_{st+1} and refine the data set as follows:

a. Input the selected *task ij* at P_{st+1}.

b. Remove the selected *task ij* from S_t and form S_{t+1} by adding its *successor* if all its *predecessor* tasks are already scheduled except for the last task.

c. Add *t* with 1

d. Update the *available time* for each machine

e. Fix the R_{ij} for all *task ij* in $S_{t+1,}$ namely:

1. For which is the *successor* of the selected *task ij*,
   R_{ij} = max (C^*, available time at machine *k*).
2. For *task ij* that has not been selected in the previous *step t*,
   R_{ij} = *max* (R_{ij} at step *t* before, available time at machine *k*).

Step 7: If there are still unscheduled tasks, perform Step 3, otherwise stop.

The flowchart of the scheduling algorithm of the proposed forward approach model can be seen in Figure 3.

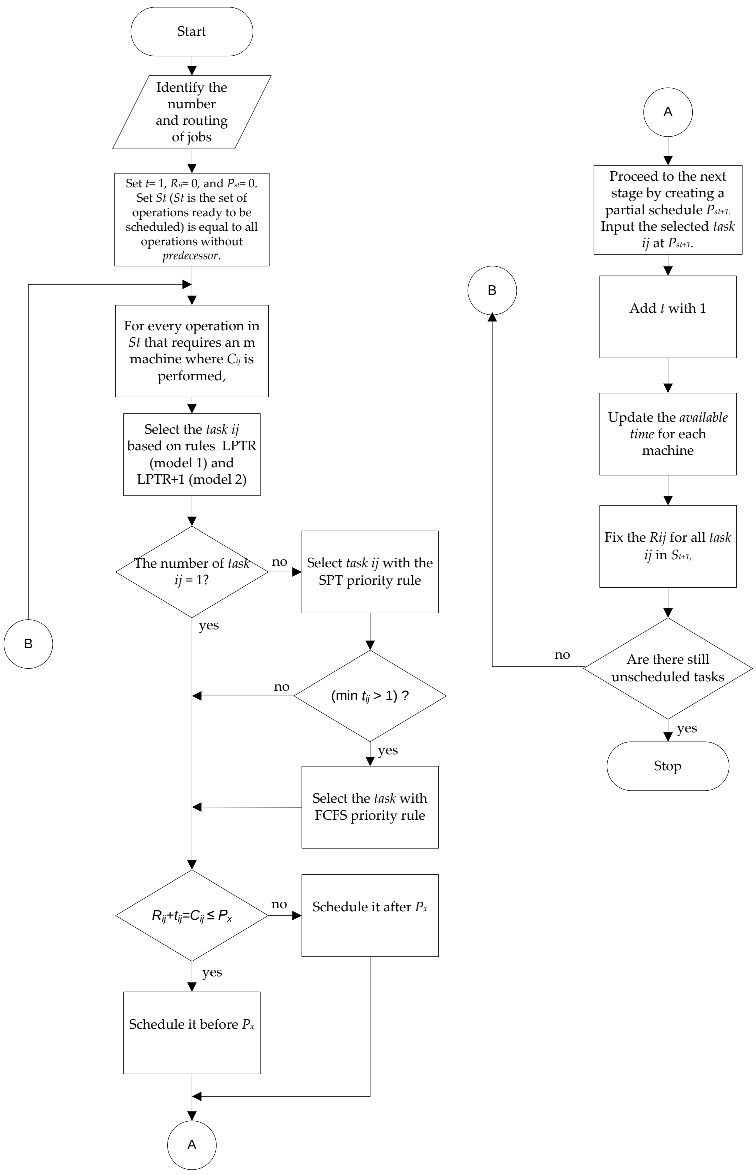

Figure 3. Flowchart of the proposed forward approach model

The calculation stages for the proposed model with the backward-forward approach are described in the following algorithm.

Step 1: Initiation Stage
Identify the number and routing job.

Step 2: Set t = 1 and P_{st} = 0 and determine S_t of each *job* starting from the most recent operation where C_{ij} at last *task ij* = d_i.

Step 3: For each operation in S_t that requires m machine where R_{ij} is performed ($R_{ij} = d_i - t_{ij}$), select *task ij*.
In model 1: Select the *task ij* that has the largest remaining processing time that considers t_{ij} in the *task ij* (LPTR)
In model 2: Select *task ij,* which has the largest remaining processing time without considering t_{ij} in t*ask ij* (LPTR+1).

Step 4:
a. If there is only one *task ij* selected, *task* (*ij* = 1), then schedule the *task*
b. If the number of selected *tasks ij* is more than one *task* (*ij* > 1), then select *task ij* with the SPT priority rule (min t_{ij})
c. If there is still a *task ij* that has (*min* t_{ij}) at the same price (min t_{ij} > 1), then select the *task* with FCFS priority rule.

Step 5: Schedule the selected *ij* task before the possible P_x
The selected *task ij* will have an Irc that has considered the *Px* schedule to have R^*, with m^*.

Step 6: Proceed to the next step by creating a partial schedule P_{st+1} and refine the data set by:
a. Input the selected *task ij* at P_{st+1}.
b. Remove the selected *task ij* from *St* and form S_{t+1} by adding its *successor* if all its *predecessor* tasks are already scheduled except for the last task.
c. Add t with 1
d. Update the *available time* for each machine
e. Fix the R_{ij} for all *task ij* in S_{t+1}, namely:
(a) For which is the *successor* of the selected *task ij*, R_{ij} = max (C^*, available time at machine k)
(b) For *task ij* that has not been selected in the previous step t, R_{ij} = max (R_{ij} at step t before, available time at machine k).

Step 7: If there is still an unscheduled *task*, perform Step 3 and then stop.

Step 8: If the scheduling result is infeasible, advance all infeasible tasks to the point of t = 0.

Step 9: Set i = 1, g = 1, *finish time* = 0, *devSt* = 0, $O = \Phi$, *jobStart* = *mcStart* = 0 for all operations.

Step 10: $O[g]$ as the current operation, *jobStart* = *mcStart* = 0 for infeasible jobs.
Determine:
newStart now= *max* (*jobStart*, *mcStart*)
newEnd now = *newStart now*+ t_{so}

Step 11: If the current job does not match the set of affected operations in the *aff* set, move to Step12, if not then *reset aff* (v) to Step 13.

Step 12: Determine:
newStart aff (v)= *newStart now*+ (C_j job the largest infeasible x (-1))
Proceed to Step 13

Step 13: Calculate *newEnd aff* (v) = *newStart aff* (v) + t_{ij}. Then move to Step14

Step 14: If *newEnd aff*(v) ≤ P_x (maintenance activity start schedule), then schedule before P_x
If *newEnd aff(v)* > P_x (maintenance activity start schedule), then schedule before P_x

Step 15: *Set aff*[i] = current operation; i add with 1.

Step 16: Determine *noj*, if it exists then:
$O[g]$ = *noj* and *jobStart* from $O[g]$ = *newEnd now,*
newEnd now= *startTime* + (C_j *job the largest infeasible* x (-1)),
Proceed to Step 17.

Step 17: Determine *newEnd aff* (v) = *newEnd now* + t_{ij}.
Check the Step 14, Move to the Step 18

Step 18: Determine *nom*, if it exists then:

O[*g*] = *nom* and *mcStart* from *O*[*g*] = *newEnd now*
newEnd now= *startTime* + (C_j *job the largest infeasible x* (-1)).
Proceed to Step 19.
Step 19: Calculate *newEnd aff* (*aff*) = *newEnd now* + t_{ij},
Check the Step 14, Move to the Step 20
Step 20: Subtract the current operation from *set O* and add the new member *O* from Step 15.
Step 21: If *O* = Ø, then *stop*, otherwise find new *noj* and *nom* from the current *set O*.

4. NUMERICAL EXAMPLES

Simulations were conducted for six cases, with details of the total jobs, machines, and penalty costs used (Table 2). The cost of earliness and tardiness is $ 1/unit time and $ 2/unit time, respectively. For each machine, maintenance activity needs a certain amount of time t = 2 units of time. Whereas, there is a time interval of T = 8 units of time between two consecutive maintenance periods [16].

Table 2. The case details for simulations

Cases	Number of jobs	Number of machines	Source
I	3	4	Andriani [23]
II	3	5	Andriani [23]
III	4	3	Generated data
IV	5	3	Generated data
V	4	5	Generated data
VI	3	3	Gaol et al. [24]

The due date of jobs 1, 2, 3, 4, and 5 are 38, 36, 37, 38, and 37 units of time, respectively. The process time data (Table 3) and machine routing (Table 4) used for each case are shown below.

Table 3. The process time data

CASE I	Operation *j*-th			
Job *i*-th	1	2	3	4
1	8	7	8	6
2	5	7	8	7
3	6	5	6	6

CASE II	Operation *j*-th				
Job *i*-th	1	2	3	4	5
1	8	7	8	6	5
2	5	7	8	7	8
3	6	5	6	6	7

CASE III	Operation *j*-th		
Job *i*-th	1	2	3
1	7	6	8
2	6	8	5
3	8	5	7
4	5	7	6

CASE IV	Operation *j*-th		
Job *i*-th	1	2	3
1	8	7	5
2	5	7	8
3	6	5	6
4	5	6	7
5	8	5	6

CASE V	Operation *j*-th				
Job *i*-th	1	2	3	4	5
1	5	7	8	6	5
2	6	5	7	8	7
3	8	7	6	8	6
4	6	5	7	6	8

CASE VI	Operation *j*-th		
Job *i*-th	1	2	3
1	5.65	5.70	3.18
2	11.17	11.41	6.03
3	6.25	3.35	-

Each of the cases ais solved using two different combinations of priority rules, namely *Least Processing Time Remaining* (LPTR) – *Shortest Processing Time* (SPT) – *First Come First Serve* (FCFS) and LPTR+1 – SPT – FCFS, by employing two scheduling approaches, namely forward and backward-forward approach [5,21–22,25].

Table 4. The machine routing data

CASE I	Operation *j*-th			
Job *i*-th	1	2	3	4
1	3	1	4	2
2	2	1	4	3
3	1	3	2	4

CASE II	Operation *j*-th				
Job *i*-th	1	2	3	4	5
1	3	1	4	5	2
2	2	1	5	3	4
3	1	3	2	4	5

CASE III	Operation *j*-th		
Job *i*-th	1	2	3
1	2	1	3
2	1	2	3
3	2	3	1
4	3	1	2

CASE IV	Operation *j*-th		
Job *i*-th	1	2	3
1	3	1	3
2	2	3	1
3	1	3	2
4	1	2	3
5	2	1	2

CASE V	Operation *j*-th				
Job *i*-th	1	2	3	4	5
1	3	4	2	5	1
2	2	1	5	4	3
3	4	5	4	1	3
4	1	3	2	5	4

CASE VI	Operation *j*-th		
Job *i*-th	1	2	3
1	1	2	3
2	2	1	3
3	1	3	-

5. RESULTS AND DISCUSSION

Table 5 shows that each case with various jobs and machines has different penalty costs. However, increasing the number of machines and jobs results in a higher total penalty cost. In cases I (three jobs, four machines) and II (three jobs, five machines), the job will have a longer routing or processing approach. Furthermore, the recapitulation results of all cases in the proposed model are shown below.

Table 5. Recapitulation of total penalty costs

Case	Forward Approach		Backward-forward Approach	
	LPTR – SPT – FCFS	LPTR+1 – SPT – FCFS	LPTR – SPT – FCFS	LPTR+1 – SPT – FCFS
I	18.0	18.0	17.0	17.0
II	64.0	64.0	64.0	64.0
III	29.0	29.0	22.0	22.0
1V	42.0	52.0	35.0	55.0
V	60.0	54.0	82.0	54.0
VI	67.2	30.5	28.8	28.7

When the number of jobs and machines increases due to lower total penalty costs, such as in Cases II (three jobs, five machines) and V (four jobs, five machines), the total penalty cost generated in Case V is lower than in Case II. In contrast to Cases III (four jobs, three machines) and IV (five jobs, three machines), adding the same number of jobs and machines results in a higher total penalty cost. This is because the number of machines is less than the number of jobs.

A combination of different priority rules affects the incidence of penalty costs. This is because the order of the job process is carried out according to their respective priority rules. However, this is similar to the comparison of Cases II and V in the backward-forward approach (LPTR - SPT - FCFS), which resulted in a higher total cost. Cases II and V resulted in $64 and $82, respectively. This shows that the order of work priority affects the resulting penalty costs.

Table 6 shows that the scheduling results of the proposed model are higher than Andriani [23]. This is because the proposed scheduling model has a stipulated time used for the maintenance activities of each machine. With maintenance time, the job is completed within a longer period. Figure 4 shows the difference in the Gantt chart between the results of the proposed and Andriani's [23] models. The differentiated Gantt chart is a backward-forward approach (LPTR+1 - SPT - FCFS) used in Case I to represent the results obtained. The time used for maintenance activities increases the *flow time.*

Table 6. The comparison of results between Andraini [23] and the proposed models

Model	Case	Forward Approach		Backward-forward Approach	
		LPTR – SPT – FCFS	LPTR+1 – SPT – FCFS	LPTR – SPT – FCFS	LPTR+1 – SPT – FCFS
Andriani [23]	I	13.0	13.0	-	-
	II	22.0	22.0	60.0	64.0
Proposed	I	18	18	17	17
	II	64	64	64	96

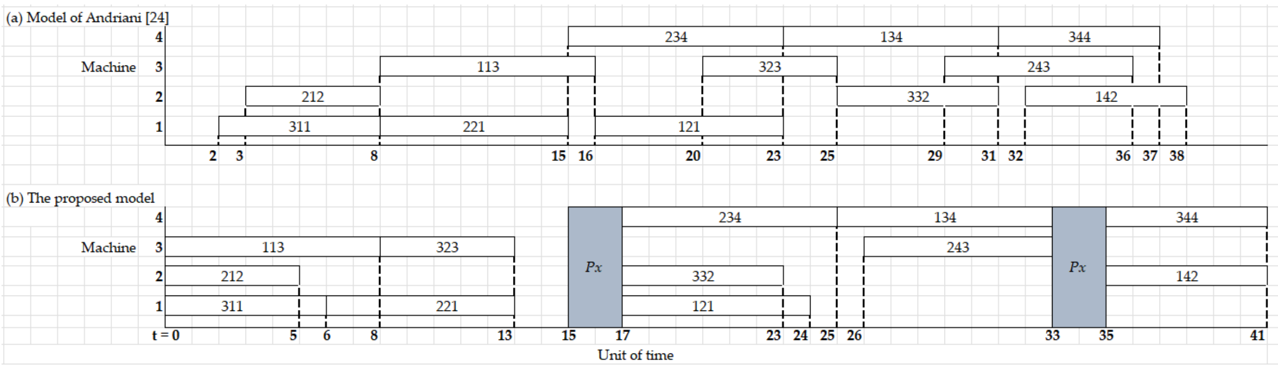

Figure 4. Gantt chart differences in the *backward-forward* approach in Case I

Figure 5 shows scheduling after the *backward-forward* stage has been conducted. Scheduling carried out after the *backward* stage leads to an *infeasible* result. Therefore, it is carried out in the *forward* stage, and the schedule becomes *feasible*. In this Gantt chart, the resulting *flow time* can be suitable because, during the *job* process, none was allocated to the maintenance activities, hence, the machine needs to be available at all times.

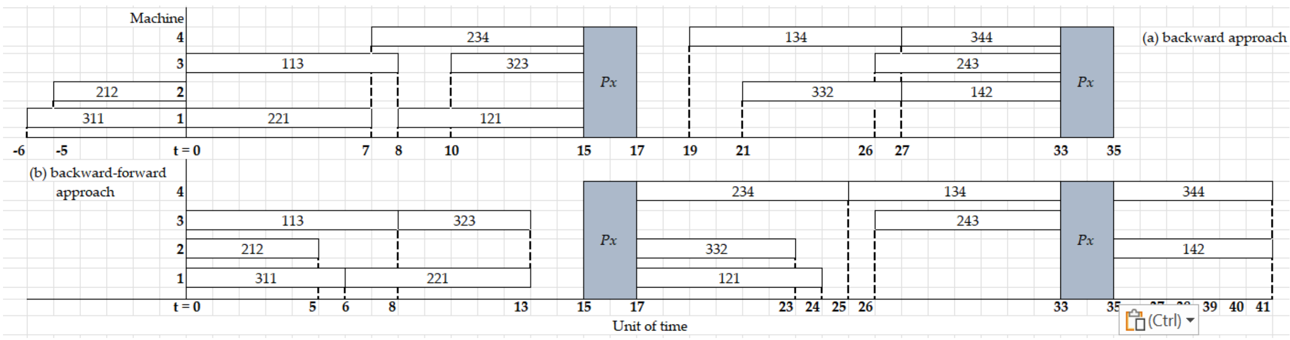

Figure 5. The application of a backward-forward approach to produce a feasible task

The *forward* stage causes a shift, resulting in an *infeasible job* or *infeasible task*. Therefore, the resulting *flow time* is greater than the *due date*, leading to a late schedule and incurring penalty costs. The starting times for *jobs* one, two, and three are not *t* = 0 because it does not result in penalty costs, namely *earliness*. After the *forward* stage was conducted, not all *jobs* were shifted to follow the *infeasible ones*. Instead, it aims to consider the number of penalty costs generated. Task 243 needs to be shifted to *t* = 35 following *task* 311, which is *infeasible* because it was advanced at *t*= 26. Even though the resulting *flow time* for *job* two gets an *earliness* penalty cost of $ 5, compared to the outcome of shifts following an *infeasible job*, the resulting *flow time* obtains a *tardiness* penalty cost of $ 8. This simply means that not all *jobs* have to shift to follow the *infeasible ones* at the *forward* stage. The *job* can shift, advance, or stay fixed, but there is a need to consider whether or not it *crashes* with the others and machines. When there is no *crash*, the *job* can be advanced or fixed at the original time.

6. CONCLUSION

In this study, we discuss a multi-machine problem in job shop scheduling conditions. In the proposed model, a periodic maintenance activity is inspired by the model of Chen [16]. Several numerical examples are applied to validate the proposed algorithm using four different priority rules. A combination of priority rules influences the *flow time* and total penalty costs.

Internally, supposing the maintenance activities are not scheduled, the proposed model scheduling results are the same as Andriani [23]. This shows that the proposed model can be used to complete *job shop* scheduling with or without preventive maintenance schedules. Externally, the proposed model produces a higher *flow time* because time is allocated for maintenance activities.

REFERENCES

[1] K. R. Baker and G. D. Scudder, "Sequencing with earliness and tardiness penalties: A review," *Oper Res*, vol. 38, no. 1, pp. 22–36, 1990, Accessed: Sep. 20, 2024. [Online]. Available: https://about.jstor.org/terms

[2] H. Van and D. Parunak, "Characterizing the manufacturing scheduling problem," *J Manuf Syst*, vol. 10, no. 3, pp. 241–259, 1991, Accessed: Sep. 20, 2024. [Online]. Available: https://doi.org/10.1016/0278-6125(91)90037-3

[3] J. Blazewicz, W. Domschke, and E. Pesch, "The job shop scheduling problem: Conventional and new solution techniques," *Eur J Oper Res*, vol. 93, pp. 1–33, 1996, Accessed: Sep. 20, 2024. [Online]. Available: https://doi.org/10.1016/0377-2217(95)00362-2

[4] A. Arisha, P. Young, and M. El Baradie, "Job Shop Scheduling Problem: An overview," in *International Conference for Flexible Automation and Intelligent Manufacturing*, Dublin, Ireland, 2001, pp. 682–693. [Online]. Available: https://arrow.tudublin.ie/buschmarcon

[5] G. Li, "Single machine earliness and tardiness scheduling," *Eur J Oper Res*, vol. 96, no. 3, pp. 546–558, 1997, Accessed: Sep. 20, 2024. [Online]. Available: https://doi.org/10.1016/S0377-2217(96)00062-8

[6] F. Sivrikaya-S and G. Ulusoy, "Parallel machine scheduling with earliness and tardiness penalties," *Comput Oper Res*, vol. 26, pp. 773–787, 1999, Accessed: Sep. 20, 2024. [Online]. Available: https://doi.org/10.1016/S0305-0548(98)00090-2

[7] U. Dorndorf, E. Pesch, and T. An Phan-Huy, "Solving the open shop scheduling problem," *Journal of Scheduling*, vol. 4, no. 3, pp. 157–174, 2001, https://doi.org/10.1002/jos.73

[8] E. Mokotoff, "Parallel machine scheduling problems: A survey," *Asia-Pacific Journal of Operational Research*, vol. 18, no. 2, p. 193, 2001, Accessed: Sep. 20, 2024. [Online]. Available: https://www.proquest.com/scholarly-journals/parallel-machine-scheduling-problems-survey/docview/204764435/se-2

[9] L. Wang and D.-Z. Zheng, "An elective hybrid optimization strategy for job-shop scheduling problems," *Comput Oper Res*, vol. 28, pp. 585–596, 2001, Accessed: Sep. 20, 2024. [Online]. Available: https://doi.org/10.1016/S0305-0548(99)00137-9

[10] R. Maheswaran and S. G. Ponnambalam, "An investigation on single machine total weighted tardiness scheduling problems," *International Journal of Advanced Manufacturing Technology*, vol. 22, no. 3–4, pp. 243–248, 2003, https://doi.org/10.1007/s00170-002-1466-0

[11] M. Feldmann and D. Biskup, "Single-machine scheduling for minimizing earliness and tardiness penalties by meta-heuristic approaches," *Comput Ind Eng*, vol. 44, no. 2, p. 307, 2003, Accessed: Sep. 20, 2024. [Online]. Available: www.elsevier.com/locate/dsw

[12] C. Koulamas, "The single-machine total tardiness scheduling problem: Review and extensions," *Eur J Oper Res*, vol. 202, no. 1, pp. 1–7, Apr. 2010, https://doi.org/10.1016/j.ejor.2009.04.007

[13] R. Ruiz and J. A. Vázquez-Rodríguez, "The hybrid flow shop scheduling problem," *Eur J Oper Res*, vol. 205, no. 1, pp. 1–18, Aug. 2010, https://doi.org/10.1016/j.ejor.2009.09.024

[14] J. D. Blocher, S. Chand, and K. Sengupta, "Changeover scheduling problem with time and cost considerations: Analytical results and a forward algorithm," *Oper Res*, vol. 47, no. 4, pp. 559–569, 1999, https://doi.org/10.1287/opre.47.4.559

[15] V. Lauff and F. Werner, "Scheduling with common due date, earliness and tardiness penalties for multimachine problems: A survey," *Math Comput Model*, vol. 40, no. 5–6, pp. 637–655, 2004, https://doi.org/10.1016/j.mcm.2003.05.019

[16] W. J. Chen, "Minimizing total flow time in the single-machine scheduling problem with periodic maintenance," *Journal of the Operational Research Society Journal of the Operational Research Society*, vol. 57, no. 4, pp. 410–415, 2006, Accessed: Sep. 20, 2024. [Online]. Available: www.palgrave-journals.com/jors

[17] A. Allahverdi, C. T. Ng, T. C. E. Cheng, and M. Y. Kovalyov, "A survey of scheduling problems with setup times or costs," *Eur J Oper Res*, vol. 187, no. 3, pp. 985–1032, Jun. 2008, https://doi.org/10.1016/j.ejor.2006.06.060

[18] D. N. Pham and A. Klinkert, "Surgical case scheduling as a generalized job shop scheduling problem," *Eur J Oper Res*, vol. 185, no. 3, pp. 1011–1025, Mar. 2008, https://doi.org/10.1016/j.ejor.2006.03.059

[19] C. J. Hsu, C. Low, and C. T. Su, "A single-machine scheduling problem with maintenance activities to minimize makespan," *Appl Math Comput*, vol. 215, no. 11, pp. 3929–3935, Feb. 2010, https://doi.org/10.1016/j.amc.2009.11.040

[20] A. Lova and P. Tormos, "Combining Random Sampling and Backward-Forward heuristics for Resource-Constrained Multi-Project Scheduling," in *the Eight International Workshop on Project Management and Scheduling*, Valencia, Spain, 2002, pp. 244–48. Accessed: Sep. 20, 2024. [Online]. Available: https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=53df782f2a2c766146ac43b1a03552aa7118bef1

[21] H.-M. Wang, P.-C. Chang, and F.-D. Chou, "A hybrid forward/backward approach for single batch scheduling problems with non-identical job sizes," *Journal of the Chinese Institute of Industrial Engineers*, vol. 24, no. 3, pp. 191–199, 2007, Accessed: Sep. 20, 2024. [Online]. Available: https://doi.org/10.1007/978-0-387-74905-1_4

[22] A. Udomsakdigool and V. Kachitvichyanukul, "Multiple-colony ant algorithm with forward-backward scheduling approach for job-shop scheduling problem," in *the Advances in Industrial Engineering and Operations Research*, Boston: Springer, USA, 2008, pp. 39–55. Accessed: Sep. 20, 2024. [Online]. Available: https://doi.org/10.1007/978-0-387-74905-1_4

[23] V.E. Andriani, "Pengembangan Model Penjadwalan Job Shop untuk Meminimumkan Total Biaya Tardiness dan Earliness," Thesis (Unpublished), Industrial Engineering Department, *Universitas Pembangunan Nasional Veteran, Yogyakarta*, 2011 (in Indonesian).

[24] S.I. Gaol, U. Mauidzoh, and. Astuti, "Analisa Penjadwalan Job Shop untuk Meminimalkan Waktu Keseluruhan Produk Menggunakan Pendekatan Heuristik Dispatching Rule," Thesis (Unpublished), *Sekolah Tinggi Teknologi Adisutjipto, Yogyakarta*, 2012 (in Indonesian).

[25] L. Mönch., R. Unbehaun., & Y.I. Choung., "Minimizing earliness–tardiness on a single burn-in oven with a common due date and maximum allowable tardiness constraint". Or Spectrum, 28, 177-198, (2006).