International Journal of Knowledge and Systems Science
Volume 13 • Issue 1

Novel Hybrid Algorithms for a Single Machine Scheduling Problem With an Overtime Constraint

Jakkrit Latthawanichphan, King Mongkut's University of Technology North Bangkok, Thailand
Watcharapan Sukkerd, Rajamangala University of Technology Phra Nakhon, Thailand
Watchara Songserm, Rajamangala University of Technology Phra Nakhon, Thailand
Teeradej Wuttipornpun, King Mongkut's University of Technology North Bangkok, Thailand*

ABSTRACT

In this paper, a single machine scheduling problem with an overtime constraint is studied. The objective is to minimise the total penalty cost defined as the sum of tardy, early, and overtime costs. Three novel hybrid algorithms that hybridise a new heuristic with genetic algorithm, tabu search, and simulated annealing, referred to as GA^H, TS^H, and SA^H, are proposed to solve the problem. In each iteration of the proposed hybrid algorithms, a given metaheuristic is used to determine a sequence of jobs, whereas a new heuristic is used to minimise the total penalty cost of the sequence using a backward-forward scheduling technique and a penalty cost trade-off process. Exhaustive experiments are conducted to evaluate the effectiveness of the proposed hybrid algorithms. For medium-scale and large-scale problems, TS^H with its best common parameter setting referred to as TS^{H2} clearly outperforms the exact algorithm, whereas both algorithms can obtain the optimal solution for small-scale problems. In addition, the computational time of TS^{H2} is in an acceptable range for the planner.

KEYWORDS

Application in Industries, Earliness, Metaheuristics, Overtime, Single Machine, Tardiness

1. INTRODUCTION

In the scheduling context, tardiness is one of the most important penalty time components to be minimised because it always generates a very high penalty cost and also reduces the reputation of the firms. Other penalty time components, for instance, earliness and overtime, are practically assigned to jobs to reduce tardiness. However, they should be carefully assigned since they also generate early (holding) cost and overtime cost. Therefore, it is a challenging problem for planners and researchers to simultaneously minimise either penalty time components or penalty cost components. This paper addresses the problem by proposing three novel hybrid algorithms that hybridise a new heuristic with three metaheuristics to minimise the total penalty (TP) cost, which is defined as the sum of tardy,

DOI: 10.4018/IJKSS.298708 *Corresponding Author

early, and overtime costs. These penalty costs are selected based on the following reasons. First, they are the practical penalty costs that always occur in many industries. Second, using them for making a decision is easier than using their penalty times. Third, simultaneously minimising them is rarely found in the literature.

Three metaheuristics used in this paper are: genetic algorithm (GA), tabu search (TS), and simulated annealing (SA). Based on a set of experiments conducted beforehand, solving large-scale problems using their conventional algorithms requires a very long computational time to obtain steady stage solutions. This is because there are many alternatives to assigning the amount of penalty cost components to the jobs resulting in repeated searches for a given sequence. In addition, it is exceedingly complex to encode a solution to simultaneously represent the sequence of jobs and the amount of penalty cost components assigned to the jobs (Li et al., 2021; Yoda et al., 2014; Zobolas et al., 2008).

To reduce the computational time as well as increase the search ability of the metaheuristics, three novel hybrid algorithms are proposed. Their concept is to hybridise a new heuristic with the three metaheuristics (referred to as GA^H, TS^H, and SA^H). In each iteration of the proposed hybrid algorithms, a given metaheuristic is used to determine a sequence of jobs, whereas a new heuristic is used to minimise the TP cost of the sequence. By the proposed hybrid algorithms, only one solution for a given sequence of jobs is obtained. It can remedy the repeated searches and reduce the computational time. Furthermore, encoding the solution for the problem is much easier since only the sequence of jobs needs to be encoded.

The effectiveness of the proposed hybrid algorithms is evaluated using a single machine production shop. Although it is the simplest shop floor, it is proved to be an NP-hard problem when tardiness and overtime are minimised simultaneously (Jaramillo & Erkoc, 2017; Yang et al., 2004).

This paper makes contribution to scheduling theory and the related industries. For the first contribution, three novel hybrid algorithms capable of minimising the TP cost that is a practical objective and rarely found in the literature, are proposed. For the second contribution, the proposed hybrid algorithms are developed as a software package so that the planner can use it in real practice. It can be downloaded along with explicit instructions by following the link provided in the conclusion section of this paper.

This paper is organised into eight sections as follows: 1) introduction, 2) literature review, 3) overtime policies, 4) problem formulation, 5) details of the proposed heuristic and hybrid algorithms, 6) details of case studies and experiments, 7) results and discussion, and finally 8) conclusion and recommendations for further study.

2. LITERATURE REVIEW

Although, the one-stage or single machine (SM) problem is the simplest combinatorial scheduling problem, it is generally categorised as NP-hard when dealing with large-scale sophisticated problems (Du & Leung, 1990; Jaramillo & Erkoc, 2017; Vakhania, 2018; Yang et al., 2004; Zhu & You, 2017). The objectives always studied in the literature are: tardiness, earliness, lateness, overtime, setup time, flow time, makespan, and their variants. In terms of time, each of them is commonly referred to as "penalty time." And, in terms of cost, it is referred to as "penalty cost."

For large-scale problems, many researchers have attempted to minimise these objectives, either individually or simultaneously through two main solution approaches: heuristics and metaheuristics. Exact algorithms such as integer programming, mixed integer programming, branch-and-bound, and so forth, are rarely selected as the main solution approach since they can obtain an optimal solution within a short or practical computational time only for small-scale problems. In fact, in most of the scheduling studies of large-scale problems, exact algorithms are used as a benchmark to evaluate the efficiency of their proposed algorithms. When the proposed algorithm obtains a solution that is near-to or similar to the optimal solution for small-scale problems, it is regarded as an excellent algorithm for solving medium-scale and large-scale problems as well (Cheng et al., 2005; Ding et al., 2016; Ferrolho & Crisóstomo, 2007; Geiger, 2010).

A literature survey related to the SM scheduling problems that minimised either the penalty times or penalty costs over the past three decades, is summarised in Table 1. It is divided into two main categories: SM with non-overtime (SMNO) and SM with overtime (SMO). Both categories are reviewed in terms of three focused aspects: objectives, solution approaches, and problem characteristics, because they are later used to specify the scope of this paper.

Table 1. The literature survey on the SM problems with minimising either penalty times or costs

Authors (Year)	SM type	Objectives	Solution approaches	Problem characteristics
Melouk et al. (2004)	SMNO	Makespan	SA	Non-identical batch
Nesello et al. (2018)	SMNO	Makespan	Iterative exact algorithm	Periodic maintenance with SDST
Perez-Gonzalez & Framinan (2018)	SMNO	Makespan	Constructive heuristics	Periodic machine availability
Nazif & Lee (2010)	SMNO	Maximum lateness	GA	Job family SDST
Györgyi & Kis (2018)	SMNO	Maximum lateness	Branch-and-cut	Raw material constraints
Sioud et al. (2012)	SMNO	Total tardiness	GA with hybrid crossover	SDST
Süer et al. (2012)	SMNO	Total tardiness	GA	Non-zero ready times and non-pre-emptive
Herr & Goel (2016)	SMNO	Total tardiness	Heuristic	Job family SDST with resource constraints
Molaee et al. (2011)	SMNO	Maximum earliness, and number of tardy jobs	Heuristic	Availability constraints
Rabadi et al. (2002)	SMNO	Total tardiness and earliness	SA	SDST
Li et al. (2015)	SMNO	Total tardiness and earliness	Heuristic, hybrid GA	Common due dates and non-identical batch sizes
Low et al. (2016)	SMNO	Total tardiness and earliness	Dynamic programming	Common due dates and unavailability periods
Yazdani et al. (2017)	SMNO	Total tardiness and earliness	EVNS	Different due dates and unavailability periods
González & Vela (2015)	SMNO	Total weighted tardiness	Memetic algorithm	SDST
Kirlik & Oguz (2012)	SMNO	Total weighted tardiness	VNS	SDST
Suppiah & Omar (2014)	SMNO	Total weighted tardiness	Hybrid TS	Incompatible families and SDST
Khorshidian et al. (2011)	SMNO	Total weighted tardiness and earliness	GA	Pre-emptive JIT
M'Hallah & Alhajraf (2016)	SMNO	Total weighted tardiness and earliness	Hybrid ACO and VNS	JIT
Arroyo et al. (2011)	SMNO	Total weighted tardiness and earliness, and flow time	Multi-objective VNS	SDST
Keshavarz et al. (2015)	SMNO	Total weighted tardiness and earliness costs	Branch-and-bound, Lagrangian relaxation	SDST
Valente & Gonçalves (2009)	SMNO	Linear earliness and quadratic tardiness costs	GA	No machine idle time
Vilà & Pereira (2013)	SMNO	Total weighted quadratic tardiness and earliness costs	Insertion heuristics	No machine idle time
Yang et al. (2004)	SMO	Total weighted tardiness and overtime costs	Pseudo-polynomial time, Local search algorithm	Non-pre-emptive
Jaramillo & Erkoc (2017)	SMO	Total weighted tardiness and overtime costs	Three-stage heuristic	Pre-emptive with the common processing time
Jaramillo & Erkoc (2018)	SMO	Total weighted tardiness and overtime costs	Aggregate pre-emptive scheduling	Pre-emptive with the common processing time

In the SMNO category, many research works were conducted to minimise either a penalty time or a penalty cost. They can be summarised as follows. Makespan was minimised in three problem and solution approach pairs: the non-identical batch processing problem that was solved using SA (Melouk et al., 2004), the problem with periodic maintenance and sequence-dependent setup time (SDST) using an iterative exact algorithm (Nesello et al., 2018), and the periodic machine availability problem using a set of constructive heuristics (Perez-Gonzalez & Framinan, 2018). Maximum lateness was minimised by GA for a job family SDST problem (Nazif & Lee, 2010), and by a branch-and-cut algorithm for a raw material constraint problem (Györgyi & Kis, 2018). Total tardiness was minimised in the following problem and solution approach pairs: the SDST problem that was solved using GA with hybrid crossover operators (Sioud et al., 2012), the problem with non-zero ready times and non-pre-emptive allowance using GA (Süer et al., 2012), and the job family SDST problem with resource constraints using a heuristic (Herr & Goel, 2016).

Besides the single objective studies, some combinations of either penalty times or penalty costs were also considered. Maximum earliness and the number of tardy jobs were simultaneously minimised by a heuristic for a problem with availability constraints (Molaee et al., 2011). Total tardiness and earliness were minimised in the following studies: the SDST problem that was solved using SA (Rabadi et al., 2002), the problem with common due dates and non-identical batch sizes using heuristics and a hybrid GA (Li et al., 2015), the problem with common due dates and unavailability periods using dynamic programming (Low et al., 2016), and the problem with different due dates and multiple unavailability periods using an enhanced variable neighbourhood search referred to as EVNS (Yazdani et al., 2017).

Cases with the variants of penalty times and costs, such as weighted and quadratic terms, were also studied. Total weighted tardiness was minimised in the following problem and solution approach pairs: the SDST problem that was solved using a memetic algorithm (González & Vela, 2015), the SDST problem using VNS (Kirlik & Oguz, 2012), and the problem with incompatible families and SDST using a hybrid TS (Suppiah & Omar, 2014). Total weighted tardiness and earliness were minimised by GA for a pre-emptive just-in-time (JIT) problem (Khorshidian et al., 2011), and by a hybrid between ant colony optimisation (ACO) and VNS for a JIT problem (M'Hallah & Alhajraf, 2016). A multi-objective VNS was proposed to minimise total weighted tardiness and earliness as a first priority, and minimise total flow time as a second priority, for an SDST problem (Arroyo et al., 2011). Total weighted tardy and early costs were minimised by Lagrangian relaxation and a branch-and-bound algorithm for an SDST problem (Keshavarz et al., 2015). Linear early and the quadratic tardy costs were minimised by GA for a non-machine idle time problem (Valente & Gonçalves, 2009). Weighted quadratic tardy and early costs were minimised by insertion heuristics for a non-machine idle time problem (Vilà & Pereira, 2013).

Unlike the SMNO category, only a few works related to the SMO problem are found in the literature. The objectives of the SMO problem were frequently studied in a combination of tardy cost and overtime cost. This is because overtime is normally required when tardiness arises. They can be summarised as follows. Total weighted tardy and overtime costs were minimised in a non-pre-emptive problem and solved by a pseudo-polynomial time and a local search algorithm (Yang et al., 2004). The same objective was minimised in a pre-emptive with the common processing time problem and it was solved by two heuristics: a three-stage heuristic (Jaramillo & Erkoc, 2017) and an aggregate pre-emptive scheduling heuristic (Jaramillo & Erkoc, 2018).

Based on the review, some conclusions can be drawn. First, the SMNO and SMO problems are still of great interest to the scheduling community. It clearly shows that there are a very limited number of works that contribute to the SMO problem. Second, penalty times and costs are still an interesting matter at present. From a practical management point of view, however, penalty costs can be simply translated rather than penalty times. In addition, dealing with penalty costs can avoid the hard work of determining the optimal weights for penalty time components. Third, simultaneously minimising overtime and other penalty costs (especially tardy cost) renders the problem NP-hard

thus making it difficult to determine the optimal solution within a practical computational time by the exact algorithm. This is because many binary and integer variables (particularly overtime) are required to satisfy the constraints. Fourth, minimising the sum of tardy, early, and overtime costs, as performed in this paper, is rarely found in the literature. Fifth, the solution approaches that are frequently used for large-scale problems, are the heuristic and metaheuristic algorithms with their modifications or hybrids. These approaches are applied to other sophisticated combinatorial problems as well (Al-Moadhen et al., 2016; Chiadamrong & Tangchaisuk, 2021; Hewahi, 2015; Rerkjirattikal & Olapiriyakul, 2019; Rerkjirattikal et al., 2020a; Rerkjirattikal et al., 2020b; Srizongkhram et al., 2020; Zhang & Guo, 2011). Finally, for large-scale problems, the effectiveness of the proposed algorithms is evaluated by comparing their solutions with the existing algorithms or the known bounds from the exact algorithm. On the other hand, for small-scale problems, it is evaluated in comparison with the optimal solution from the exact algorithm.

3. OVERTIME POLICIES

In this paper, an overtime decision is made based on three practical overtime policies observed from the selected factories. The unit of all time parameter measures (processing time, due time, regular time, and overtime) is in hours. Each day consists of eight hours of a regular period and four hours of an overtime period.

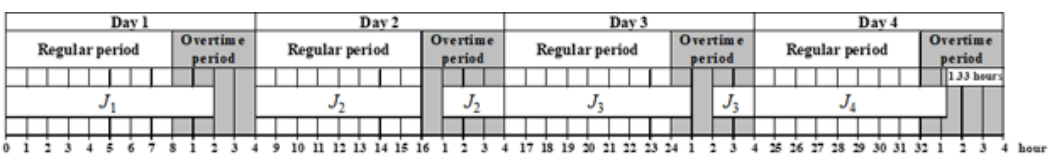
For the first policy, the required overtime period of a given job must start from the earliest available overtime hour on a given day. Figure 1 illustrates two and three hours of two overtime periods required for jobs J_1 on day 1 and J_2 on day 2. Based on this policy, two hours of the overtime period for J_1 are allowed, whereas three hours of the overtime period for J_2 are not allowed. For the second policy, the required overtime period must be occupied by consecutive overtime hours. In Figure 1, there are three hours of the overtime period required for job J_3 on day 3. Based on the second policy, it does not allow three hours for J_3 since they are not occupied by consecutive hours. For the last overtime policy, the required overtime period must be an integer starting from one hour to four hours. Fractional numbers are not allowed since they are hard to handle in terms of time and cost management. For instance, J_4 is not allowed to occupy 1.33 hours of the overtime period on day 4 as shown in Figure 1. These overtime policies are implemented in many factories and the selected factories in this paper.

Note that if the unit of the time parameter measures is changed to different units, such as minute, half-hour, day, and so forth, all time parameters must be converted to integer of the new unit. For example, six hours of processing time must be converted to 360 minutes when the unit of interest is in minutes.

4. PROBLEM FORMULATION

This section presents the formulation of the SMO problem in this study by using the mathematical model. The indices, parameters, variables, objective, and constraints are defined as follows.

Figure 1. The illustration of the practical overtime policies



4.1 Indices

j = Index of job from 1 to JOB
q = Index of position from 1 to POS
d = Index of day from 1 to D
ot = Index of overtime hour from 1 to OT
q' = Member of position q
d' = Member of day d
ot' = Member of overtime ot

4.2 Parameters

JOB = Number of jobs
POS = Number of positions
D = Number of days (in days)
OT = Maximum overtime hours per day (in hours)
R = Maximum regular hours per day (in hours)
M = Large positive number
P_j = Processing time of job j (in hours)
Due_j = Due time of job j (in hours)
t_j^{cost} = Tardy cost per hour of job j (in dollars)
e_j^{cost} = Early cost per hour of job j (in dollars)
ot_j^{cost} = Overtime cost per hour of job j (in dollars)

4.3 Variables

$C^{\max}$ = Maximum completion time (in hours)
T_q = Tardiness of a job at position q (in hours)
$T_{j,q}$ = Tardiness of job j at position q (in hours)
E_q = Earliness of a job at position q (in hours)
$E_{j,q}$ = Earliness of job j at position q (in hours)
OT_q = Overtime of a job at position q (in hours)
$OT_{j,q}$ = Overtime of job j at position q (in hours)
n = Integer number

4.4 Decision Variables

S_q = Start time of a job at position q in a regular period (in hours)

$S_q^{'}$ = Start time of a job at position q in an overtime period (in hours)

C_q = Completion time of a job at position q in a regular period (in hours)

$C_q^{'}$ = Completion time of a job at position q in an overtime period (in hours)

$$x_{j,q} = \begin{cases} 1 & \text{if job } j \text{ is processed at position } q \\ 0 & \text{otherwise} \end{cases}$$

$$g_{d,ot} = \begin{cases} 1 & \text{if the job is processed on day } d \text{ at hour } ot \\ 0 & \text{otherwise} \end{cases}$$

$$y_{q,d,ot} = \begin{cases} 1 & \text{if the job at position } q \text{ is processed on day } d \text{ at hour } ot \\ 0 & \text{otherwise} \end{cases}$$

$$a_{q,d,ot} = \begin{cases} 1 & \text{if overtime is allowed for the job at position } q \text{ after day } d \text{ at hour } ot \\ 0 & \text{otherwise} \end{cases}$$

$$b_{q,d,ot} = \begin{cases} 1 & \text{if overtime is allowed for the job at position } q \text{ before day } d \text{ at hour } ot \\ 0 & \text{otherwise} \end{cases}$$

$$f_{q,d,ot} = \begin{cases} 1 & \text{if overtime is allowed for unscheduled jobs before position } q \text{ on day } d \text{ at hour } ot \\ 0 & \text{otherwise} \end{cases}$$

4.5 Objective Function

$$\text{Minimise } Z = \sum_{j=1}^{JOB}\sum_{q=1}^{POS}(t_j^{\text{cost}}T_{j,q}) + \sum_{j=1}^{JOB}\sum_{q=1}^{POS}(e_j^{\text{cost}}E_{j,q}) + \sum_{j=1}^{JOB}\sum_{q=1}^{POS}(ot_j^{\text{cost}}OT_{j,q}) \tag{1}$$

4.6 Constraints

$$\sum_{q=1}^{POS} x_{j,q} = 1, \forall j \tag{2}$$

$$\sum_{j=1}^{JOB} x_{j,q} = 1, \forall q \tag{3}$$

$$C_q^{'} \geq \sum_{j=1}^{JOB} P_j x_{j,q}, q = 1 \tag{4}$$

$$C_q^{'} \geq C_{q-1}^{'} + \sum_{j=1}^{JOB} P_j x_{j,q}, q = 2,3,\ldots,POS \tag{5}$$

$$Rn - C_q^{'} < M(1 - a_{q,d,ot}), \forall q, d = 1, ot = 1, n = d \tag{6}$$

$$Rn+\sum_{ot'=1}^{ot-1} g_{d,ot'} - C_q^{'} < M(1-a_{q,d,ot}), \forall q,d=1,ot=2,3,...,OT,n=d \tag{7}$$

$$Rn+\sum_{d'=1}^{d-1}\sum_{ot'=1}^{OT} g_{d',ot'} - C_q^{'} < M(1-a_{q,d,ot}), \forall q,d=2,3,...,D,ot=1,n=d \tag{8}$$

$$Rn+\sum_{d'=1}^{d-1}\sum_{ot'=1}^{OT} g_{d',ot'} + \sum_{ot'=1}^{ot-1} g_{d,ot'} - C_q^{'} < M(1-a_{q,d,ot}), \forall q,d=2,3,...,D,ot=2,3,...,OT,n=d \tag{9}$$

$$\sum_{ot'=ot}^{OT} y_{q,d,ot'} + \sum_{d'=d+1}^{D}\sum_{ot'=1}^{OT} y_{q,d',ot'} \le M \times a_{q,d,ot}, \forall q, \forall d, \forall ot \tag{10}$$

$$g_{d,ot} = \sum_{q=1}^{POS} y_{q,d,ot}, \forall d, \forall ot \tag{11}$$

$$g_{d,ot} \ge g_{d,ot+1}, \forall d, ot=1,2,...,OT-1 \tag{12}$$

$$S_q^{'} - Rn < M(1-b_{q,d,ot}), \forall q,d=1,ot=1,n=d \tag{13}$$

$$S_q^{'} - Rn - \sum_{ot'=1}^{ot-1} g_{d,ot'} < M(1-b_{q,d,ot}), \forall q,d=1,ot=2,3,...,OT,n=d \tag{14}$$

$$S_q^{'} - Rn - \sum_{d'=1}^{d-1}\sum_{ot'=1}^{OT} g_{d',ot'} < M(1-b_{q,d,ot}), \forall q,d=2,3,...,D,ot=1,n=d \tag{15}$$

$$S_q^{'} - Rn + \sum_{d'=1}^{d-1}\sum_{ot'=1}^{OT} g_{d',ot'} - \sum_{ot'=1}^{ot-1} g_{d,ot'} < M(1-b_{q,d,ot}), \forall q,d=2,3,...,D,ot=2,3,...,OT,n=d \tag{16}$$

$$\sum_{ot'=1}^{ot-1} y_{q,d,ot'} \le M \times b_{q,d,ot}, \forall q, d = 1, ot = 2,3,...,OT \tag{17}$$

$$\sum_{d'=1}^{d-1} \sum_{ot'=1}^{OT} y_{q,d',ot'} \le M \times b_{q,d,ot}, \forall q, d = 2,3,...,D, ot = 1 \tag{18}$$

$$\sum_{d'=1}^{d-1} \sum_{ot'=1}^{OT} y_{q,d',ot'} + \sum_{ot'=1}^{ot-1} y_{q,d,ot'} \le M \times b_{q,d,ot}, \forall q, d = 2,3,...,D, ot = 2,3,...,OT \tag{19}$$

$$C_q - Rn \le M(1 - f_{q,d,ot}), \forall q, \forall d, \forall ot, n = d \tag{20}$$

$$\sum_{q'=q+1}^{POS} \sum_{d'=1}^{d-1} \sum_{ot'=1}^{OT} y_{q',d',ot'} + \sum_{q'=q+1}^{POS} \sum_{ot'=1}^{ot} y_{q',d,ot'} \le M \times f_{q,d,ot}, \forall q, \forall d, \forall ot \tag{21}$$

$$OT_q = \sum_{d=1}^{D} \sum_{ot=1}^{OT} y_{q,d,ot}, \forall q \tag{22}$$

$$OT_{j,q} \ge OT_q - M(1 - x_{j,q}), \forall j, \forall q \tag{23}$$

$$C_q \ge C'_q - \sum_{q'=1}^{q} \sum_{d=1}^{D} \sum_{ot=1}^{OT} y_{q',d,ot}, \forall q \tag{24}$$

$$C^{\max} \ge C_q, \forall q \tag{25}$$

$$S_q = C_q - \sum_{j=1}^{JOB} P_j x_{j,q} + OT_q, \forall q \tag{26}$$

$$S'_q = C'_q - \sum_{j=1}^{JOB} P_j x_{j,q}, \forall q \tag{27}$$

$$T_q \geq C_q - \sum_{j=1}^{JOB} Due_j x_{j,q}, \forall q \tag{28}$$

$$T_{j,q} \geq T_q - M(1 - x_{j,q}), \forall j, \forall q \tag{29}$$

$$E_q \geq \sum_{j=1}^{JOB} Due_j x_{j,q} - C_q, \forall q \tag{30}$$

$$E_{j,q} \geq E_q - M(1 - x_{j,q}), \forall j, \forall q \tag{31}$$

$$S_q, S_q^{'}, C_q, C_q^{'}, T_q, E_q, OT_q \geq 0, \forall q \tag{32}$$

$$T_{j,q}, E_{j,q}, OT_{j,q} \geq 0, \forall j, \forall q \tag{33}$$

The objective function is to minimise *Z*, which is the TP cost defined as the sum of tardy, early, and overtime costs as shown in equation (1). Constraints (2) and (3) ensure that a job can be scheduled at only one position and the position can process only one job at a time. Note that a position may have more than one hour depending on the processing time of the job at this position. Constraints (4) and (5) ensure that all jobs are processed in a non-pre-emptive non-overlapping manner. Constraints (6)–(10) determine which job should be processed in the overtime period of a selected day. Constraints (11)–(21) ensure when a job requires both regular and overtime periods, these periods comply with the condition that they are consecutive. Constraints (22) and (23) determine the required overtime hours of the jobs. Constraints (24) and (25) determine the completion time of the jobs and the maximum completion time, respectively. Constraints (26) and (27) determine the start time of the jobs in both regular and overtime periods. Constraints (28) and (29) calculate the tardiness of the jobs, whereas constraints (30) and (31) calculate the earliness of the jobs. Finally, constraints (32) and (33) are the non-negativity constraints of the variables.

Since the time parameters in this formulation (processing time, due time, regular time, and overtime) are integers, the time variables (start time, completion time, and others) must be integers as well. This causes the model to require many integer and binary variables, rendering the model NP-hard. As a result, it takes a very long computational time to obtain the optimal solution.

Note that the unit of time parameter measures in this formulation is in integer hours. In case the planner is interested in other units, all time parameter data mentioned above must be converted to integers of the desired units before determining the solution. For instance, if the planner requires a half-hour precision, 10 hours of processing time must be converted to 20-time slots, eight hours of regular time to 16-time slots, four hours of overtime to eight-time slots, a due time at hour 36 to at hour 72, and so on.

Table 2. Data of the numerical example and results after applying the proposed heuristic

Information of jobs						Scheduling results				Penalty costs
Job (j)	P_j	Due_j	t_j^{cost}	ot_j^{cost}	e_j^{cost}	C_j	T_j	OT_j	E_j	
1	6	29	33	18	8	29	0	0	0	0
2	7	8	44	19	16	7	0	0	1	16
3	8	10	43	9	13	11	1	4	0	79
4	4	24	37	17	8	23	0	0	1	8
Average penalty cost (APC)			39.25	15.75	11.25	Total penalty (TP) cost				103

5. DETAILS OF THE PROPOSED HEURISTIC AND HYBRID ALGORITHMS

This section explains the details the proposed heuristic and hybrid algorithms. The proposed heuristic minimises the TP cost using a backward-forward scheduling technique with a penalty cost trade-off process. An explicit numerical example using the information shown in Table 2, illustrates how the proposed heuristic works. The unit of measure of all time parameters and variables shown in Table 2 (P_j, Due_j, C_j, T_j, OT_j, E_j) is in integer hours, whereas the unit of measure of all cost parameters is in dollars.

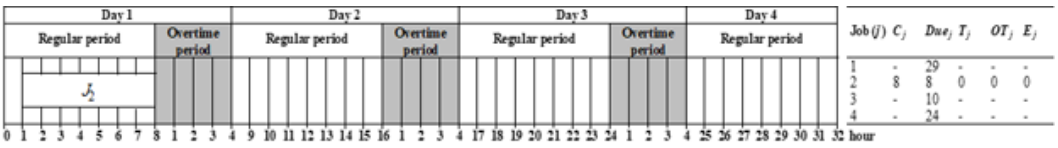
The proposed heuristic has two main steps. The first step is to determine a penalty cost relation by using the average of each penalty cost component (APC). It is calculated from tardy, early, and overtime costs of all jobs. This penalty cost relation is then used in the trade-off process in a way that the highest penalty cost is traded with the lowest penalty cost.

Through the APC values shown in Table 2, the penalty cost relation in a descending order of tardy cost (39.25) → overtime cost (15.75) → early cost (11.25) is obtained. The trade-off process trades tardiness for earliness first. When earliness is not available, overtime is used instead. If a tiebreak of the APC values occurs, for example, when the APC values of overtime and early costs are the same, a desired penalty cost relation must be specified. However, most of the factories, as well as the selected factories in this study, normally have a penalty cost relation similar to the relation in this example.

The benefit of using the penalty cost relation obtained by the APC values is to avoid the complication of the penalty cost trade-off process. This complication occurs when using many penalty cost relations obtained from all jobs. In Table 2, two possible conflicts of the penalty cost components: within the job and between the jobs, can be observed. For instance, the overtime costs of jobs J_1, J_2, and J_4 are higher than the early costs for both within the jobs and between the jobs, whereas the conflict relation occurs within J_3 and between J_2 and J_3. When the cost relations are conflicting, the penalty cost trade-off process becomes complicated, which results in a longer computational time.

The second step is to apply the backward-forward scheduling technique with the penalty cost trade-off process to the jobs in a given sequence (one job at a time). The primary complexity of this step is the situation when a job is being scheduled, the other scheduled jobs are allowed to shift either

Figure 2. Backward scheduling of J_2 from its due time



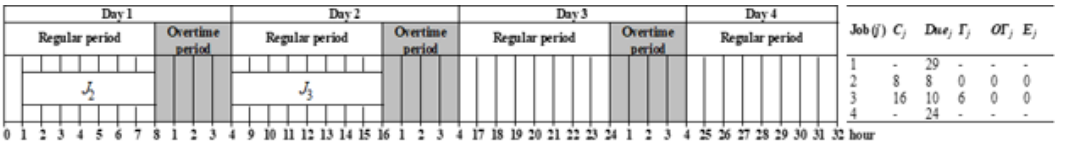
backward or forward to minimise the total penalty cost (the sequence of jobs must be maintained). In Table 2, suppose the sequence of the jobs is $J_2 \rightarrow J_3 \rightarrow J_4 \rightarrow J_1$ and the descending APC relation is tardiness → overtime → earliness. The first job in the sequence, J_2, is selected and scheduled backward from its due time (hour 8), as shown in Figure 2. Because no penalty costs occur, the trade-off process is not required.

The next job in the sequence is J_3. Suppose it is scheduled backward from its due time (hour 10); then it overlaps with J_2 by six hours (the processing time of J_3 is eight hours). Because the overlap of the jobs is not allowed, J_3 is instead scheduled forward to the right side beyond its due time for six hours, as shown in Figure 3. This results in six hours of tardiness since the completion time of J_3 (hour 16) is higher than its due time (hour 10).

Since the tardy cost of J_3 occurs, the trade-off process allows the earliness for J_2 to reduce the tardiness of J_3. Because only one hour is available for J_2 to be completed early, J_2 and J_3 are shifted back to the left side by an hour, as shown in Figure 4. This results in the one hour increase of the earliness of J_2 and the one hour reduction of the tardiness of J_3. The remaining tardiness of J_3 is now five hours.

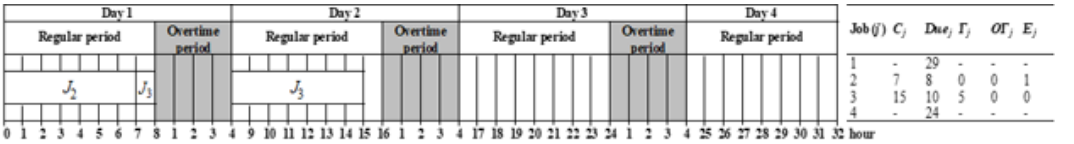
Because there is no more regular time for allowing earliness, overtime is used for the trade-off process. Four overtime hours on day 1 are used for J_3 to reduce the tardiness This results in essentially one hour of tardiness of J_3 as shown in Figure 5. The next job in the sequence is J_4. Because it can be scheduled backward from its due time (hour 24) without overlapping with other scheduled jobs, the trade-off process is not required. The final results of scheduling jobs J_3 and J_4 are shown in this figure.

Figure 3. Forward scheduling of J_3 beyond its due time



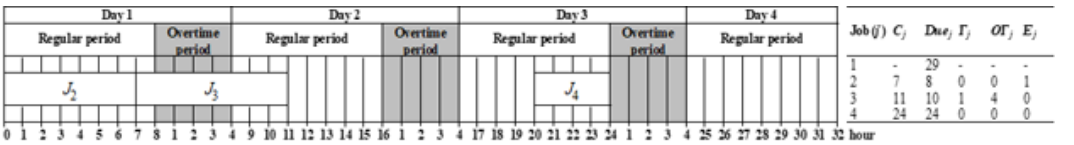
Job (j)	C_j	Due_j	T_j	OT_j	E_j
1	-	29	-	-	-
2	8	8	0	0	0
3	16	10	6	0	0
4	-	24	-	-	-

Figure 4. Backward scheduling of J_2 and J_3



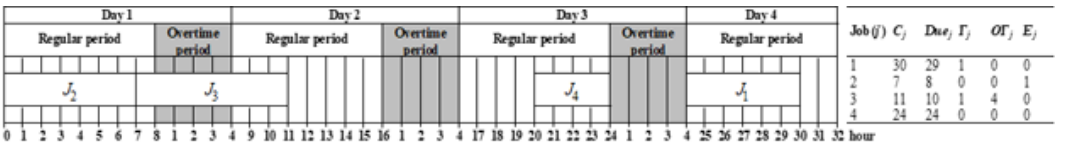
Job (j)	C_j	Due_j	T_j	OT_j	E_j
1	-	29	-	-	-
2	7	8	0	0	1
3	15	10	5	0	0
4	-	24	-	-	-

Figure 5. Backward scheduling of J_3 and J_4



Job (j)	C_j	Due_j	T_j	OT_j	E_j
1	-	29	-	-	-
2	7	8	0	0	1
3	11	10	1	4	0
4	24	24	0	0	0

Figure 6. Forward scheduling of J_1 beyond its due time

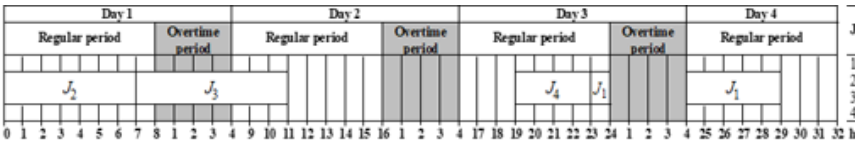


Job (j)	C_j	Due_j	T_j	OT_j	E_j
1	30	29	1	0	0
2	7	8	0	0	1
3	11	10	1	4	0
4	24	24	0	0	0

The last job in the sequence is J_1. Suppose it is scheduled backward from its due time (hour 29); then it overlaps with J_4 by one hour (the processing time of J_1 is six hours). Due to the overlap constraint, J_1 is scheduled forward to the right side beyond its due time by an hour. This results in one hour of tardiness since the completion time of J_1 (hour 30) is higher than its due time (hour 29) as shown in Figure 6.

Since the tardy cost of J_1 occurs, then the trade-off process allows the earliness of J_4 to reduce the tardiness of J_1, as shown in Figure 7. This results in increasing the one hour increase of earliness and one hour reduction of tardiness.

Figure 7. Backward scheduling of J_4 and J_1



Job (j)	C_j	Due_j	T_j	OT_j	E_j
1	29	29	0	0	0
2	7	8	0	0	1
3	11	10	1	4	0
4	23	24	0	0	1

Figure 8. The pseudo-code for the proposed heuristic

Overall procedure: The proposed heuristic.
Inputs: Sequence of jobs, number of positions (POS), processing time of job (P), due time of job (Due), tardy cost, early cost, overtime cost, and APC descending relation (tardiness cost ⇨ overtime cost ⇨ earliness cost).
Indexes: q = index of position.
Variables: MAT = machine available time, S_q = start time of job at position q, C_q = completion time of job at position q, T_q = tardiness of job at position q, OT_q = overtime of job at position q, AE = available earliness, RE = required earliness, AO = available overtime, RO = required overtime.

Begin

Assign index q to the jobs in a given sequence; //q =1, 2,…, POS.
$MAT = 0$; //Set machine available time.
for $q = 1$ to POS //Schedule one job at a time.
{ $S_q = Due_q - P_q$; //Determine start time of job at position q by backward scheduling.
  if ($S_q \geq MAT$) **//On time job case.**
    { $S_q = S_q$; //Update start time of job at position q
      $C_q = S_q + P_q$; //Determine completion time.
      $MAT = C_q$; } //Update machine available time.
  else **//Tardy job case.**
    { $S_q = MAT$; //Update start time of job at position q by forward scheduling.
      $C_q = S_q + P_q$; //Determine completion time of job at position q.
      $T_q = C_q - Due_q$; //Determine tardiness of job at position q.
      $RE = T_q$; //Determine required earliness for trading with tardiness.
      if ($AE \geq RE$) **//Trade tardiness with required earliness only.**
        { $S_q = MAT - RE$; //Update start time of job at position q.
          $C_q = S_q + P_q$; //Update completion time of job at position q.
          $T_q = 0$; } **//Job at position q is completed on time.**
      else **//Trade tardiness with all available earliness.**
        { $S_q = MAT - AE$; //Update start time of job at position q.
          $C_q = S_q + P_q$; //Update completion time of job at position q.
          $T_q = C_q - Due_q$; **//Job at position q is still tardy.**
          $RO = T_q$; //Determine required overtime for trading with tardiness.
          if ($AO \geq RO$) **//Trade tardiness with required overtime only.**
            { $C_q = S_q + (P_q - RO)$; //Update completion time of job at position q.
              $T_q = 0$; } **//Job at position q is completed on time.**
          else **//Trade tardiness with all available overtime.**
            { $C_q = S_q + (P_q - RO)$; //Update completion time of job at position q.
              $T_q = C_q - Due_q$; } **//Job at position q is essential tardy.**
}}}
Outputs: Production schedule with the minimum total penalty cost (sum of tardy, overtime, and early costs).
End

Since all jobs in the sequence are completely scheduled, the proposed heuristic is then stopped. It can be observed in Figure 7 that only the essential tardiness, earliness, and overtime, are required. This results in the minimum TP cost, which is the main benefit of the proposed heuristic. The minimum TP cost for the sequence $J_2 \rightarrow J_3 \rightarrow J_4 \rightarrow J_1$ is \$103, as shown in the last column of Table 2. The overall mechanism of the proposed heuristic is summarised in pseudo-code as shown in Figure 8.

The details of this example are also used as input parameters for the proposed mathematical model. The results show that the solution obtained from the proposed heuristic is similar to the optimal solution determined by the exact algorithm in CPLEX. This suggests that the proposed heuristic is highly efficient. However, without modifications, it cannot be applied to solve the problem in this study since it works well only when the sequence of jobs is known. Accordingly, the proposed hybrid algorithms hybridise the proposed heuristic with the selected metaheuristics in their fitness evaluation steps. In each iteration of the proposed hybrid algorithms (GA^H, TS^H, and SA^H), a given metaheuristic is used to determine a sequence of jobs, whereas the proposed heuristic is used to minimise the TP cost of the sequence. By the concept of the proposed hybrid algorithms, the search ability and the computational time of the metaheuristics can be improved.

6. DETAILS OF CASE STUDIES AND EXPERIMENTS

This section provides the details of case studies and experiments. There are eight selected factories that produce different automotive parts. Their production characteristics can be summarised as follows:

- The shop floor type is SMO with non-pre-emptive and non-overlapping of jobs.
- The job size varies in the range of 5–200 jobs.
- Each day has eight hours of a regular period and four hours of an overtime period.
- The processing time of jobs is in integer hours and it is in the range of 2–10 hours (uniform distribution).
- The due time of the jobs is in integer hours and it is in the range of 5–10 times of the processing time.

Since real data from the factory is confidential, a set of hypothetical data generated based on the mentioned characteristics is used to evaluate the effectiveness of the proposed heuristic and hybrid algorithms. These data are generated by a new instance generator called *i*-IGSP that is developed by King Mongkut's University of Technology North Bangkok (KMUTNB) and Rajamangala University of Technology Phra Nakhon (RMUTP). It is the instance generator capable of generating hypothetical data for various production shop characteristics (Latthawanichphan et al., 2019; Songserm et al., 2021). Note that all the time parameters are uniformly distributed similar to other researches in the literature (Chang et al., 2009; Pan et al., 2017; Vallada & Ruiz, 2009). However, the normal distribution is also available in *i*-IGSP. The practical overtime policies explained earlier are implemented, and the APC relation of the observed factories in a descending order is tardiness → overtime → earliness.

Based on possible job sizes observed from the factories, six problem sizes (5-job, 10-job, 20-job, 50-job, 100-job, and 200-job) are selected. Note that the 5-job and 10-job problems are considered as small-scale problems, the 20-job and 50-job problems as medium-scale problems, and the 100-job and 200-job problems as large-scale problems. In addition, to represent different characteristics of a given job size in terms of processing time and due time data, a set of five instances for each job size problem is further generated by different seed numbers (Latthawanichphan et al., 2019; Songserm et al., 2021) resulting in 30 different problem instances. These instances are used to prove that the proposed hybrid algorithms work well for various problem characteristics.

Three experiments are conducted for different purposes. The first experiment is conducted to evaluate the effectiveness of the proposed heuristic by using 120 sequences of jobs that are randomly selected from all problem sizes. The second experiment is conducted to determine the best common

parameter (BCP) setting for each hybrid algorithm (GA^H, TS^H, and SA^H). The BCP setting is very practical for the planner since it is applicable to all problem characteristics. It is better than using the best parameter setting that may have to change depending on the characteristics. Finally, the last experiment is conducted to evaluate the effectiveness of the proposed hybrid algorithms that are set by their BCP settings. Note that all instance are used for the last two experiments. The available parameter settings of the proposed hybrid algorithms for the last two experiments are shown in Table 3. They are obtained from a set of experiments that are conducted beforehand. The first and second experiments are evaluated through the relative percentage deviation (*RPD*) and its average (*ARPD*) indices, whereas the last experiment is evaluated through the relative percentage improvement (*RPI*) and its average (*ARPI*) indices.

In terms of the experimental runs required for all the experiments, 240 runs are required for the first experiment (120 runs for the proposed heuristic and 120 runs for the mathematical model), whereas 2,190 runs based on the factorial experiments of the details shown in Table 3, are required for the last two experiments. The details of these 2,190 runs are: 960 runs for GA^H (32 settings of 30 instances), 240 runs for TS^H (8 settings of 30 instances), 960 runs for SA^H (32 settings of 30 instances) and 30 runs for the mathematical model.

The computational time of the hybrid algorithms is set to six hours, which is the maximum allowance from the planner, whereas it is allowed to be significantly longer for the exact algorithm (24 hours) because it is challenging to determine the optimal solution or the best bound for a comparison purpose.

The performance indices of each experiment are denoted and calculated by the following equations. The first experiment requires two indices: $RPD_{Seq,Js}$ and $ARPD_{Js}$ as shown in equations (34) and (35). The $RPD_{Seq,Js}$ index is the relative percentage deviation of the solution obtained from the proposed heuristic over the optimal solution or the bound obtained from the exact algorithm. It is used to evaluate the effectiveness for a given sequence and job size. The $ARPD_{Js}$ index is the average of $RPD_{Seq,Js}$. It is used to evaluate the effectiveness for all the sequences of a given job size.

For the second experiment, the $RPD_{Ins,Js}$, $ARPD_{Js}$, and $ARPD_{O}$ indices as shown in equations (36), (37), and (38), are required. The $RPD_{Ins,Js}$ index, the relative percentage deviation of the solution obtained from each setting over the best solution across all settings, is first calculated for each instance and job size. The average of $RPD_{Ins,Js}$ referred to as $ARPD_{Js}$, is then calculated from all the instances of a given job size. Finally, the average of $ARPD_{Js}$ referred to as $ARPD_{O}$, is calculated from all the instances and job sizes. The parameter setting of a given hybrid algorithm that obtains the minimum $ARPD_{O}$, is the BCP setting, which is applicable to all the instances and jobs sizes.

For the last experiment, the relative percentage improvement ($RPI_{Ins,Js}$) and its average ($ARPI_{Js}$) indices as shown in equations (39) and (40), are required. To determine these indices, the worst solution is chosen from the solutions that are obtained by the proposed hybrid algorithms with their

Table 3. Parameters for GA^H, TS^H, and SA^H

Alg	p_s	Crossover	p_c	Mutation	p_m	NBS	NBO	ITemp	NIT	α	TL
GA^H	5, 20	PMX, PBX	0.6, 0.9	SWAP, INSERT	0.4, 0.6	-	-	-	-	-	-
TS^H	-	-	-	-	-	5, 30	SWAP, INSERT	-	-	-	10, 50
SA^H	-	-	-	-	-	5, 30	SWAP, INSERT	100, 300	5, 10	0.8, 0.9	-

Alg: Algorithm, p_s: Population size, p_c: Crossover probability, p_m: Mutation probability, NBS: Neighbourhood sequences, NBO: Neighbourhood operators, ITemp: Initial temperature, NIT: Number of iterations, α: Reduce temperature, TL: Tabu list size.

BCP settings and the exact algorithm. The $RPI_{Ins,Js}$ index is the relative percentage improvement over the worst solution. It is used to evaluate the effectiveness for a given instance and job size. The average of $RPI_{Ins,Js}$ referred to as $ARPI_{Js}$, it is used to evaluate the effectiveness for all instances of a given job size. The proposed hybrid algorithm that obtains the maximum $ARPI_{Js}$ is the recommended algorithm, which is applicable to all the instances of a given job size:

$$RPD_{Seq,Js} = \left[\frac{TPC_{H} - TPC_{Ex}}{TPC_{Ex}}\right]_{Seq,Js} \times 100\% \tag{34}$$

$$ARPD_{Js} = \frac{1}{20}\sum_{Seq=1}^{20} RPD_{Seq,Js} \tag{35}$$

$$RPD_{Ins,Js} = \left[\frac{TPC_{HB} - TPC_{HB}^{Best}}{TPC_{HB}^{Best}}\right]_{Ins,Js} \times 100\% \tag{36}$$

$$ARPD_{Js} = \frac{1}{5}\sum_{Ins=1}^{5} RPD_{Ins,Js} \tag{37}$$

$$ARPD_{O} = \frac{1}{6\times 5}\sum_{Js2=1}^{6}\sum_{Ins=1}^{5} RPD_{Ins,Js} \tag{38}$$

$$RPI_{Ins,Js} = \left|\frac{TPC_{HB/Ex} - TPC_{HB/Ex}^{Worst}}{TPC_{HB/Ex}^{Worst}}\right|_{Ins,Js} \times 100\% \tag{39}$$

$$ARPI_{Js} = \frac{1}{5}\sum_{Ins=1}^{5} RPI_{Ins,Js} \tag{40}$$

The notations in equations (34)–(40) are as follows: TPC = total penalty cost, *Seq* = the sequence index (1 to 20), *H* = the proposed heuristic, *HB* = GA^H, TS^H, and SA^H, *Ex* = the exact algorithm, *Best* = the best solution, *Worst* = the worst solution, *Ins* = the instance index (1 to 5), *Js* = the job size index (1 to 6).

7. RESULTS AND DISCUSSION

The proposed hybrid algorithms are coded and solved by MATLAB and the mathematical model is solved by the exact algorithm in CPLEX. A desktop computer with the Core i7 Processor 7th generation and 8 GB of RAM is used to run the programs. The Analysis of Variance (ANOVA) method and Tukey's multiple comparison test are used to analyse significant differences in the results.

Table 4. $RPD_{Seq,Js}$ and $ARPD_{Js}$ indices to evaluate the effectiveness of the proposed heuristic

Sequences	5-job		10-job		20-job		50-job		100-job		200-job	
	$RPD_{Seq,Js}$	Ctime	$RPD_{Seq,Js}$	Ctime	$RPD_{Seq,Js}$	Ctime	$RPD_{Seq,Js}$	Ctime	$RPD_{Seq,Js}$	Ctime	$RPD_{Seq,Js}$	Ctime
1	0	< 1 minute	0.33	< 1 minute	0	< 1 minute	-1.41*	1,440	-4.60*	1,440 minutes	-12.76*	1,440 minutes
2	0		0		0		0	392	-7.27*		-16.50*	
3	0		0		0		0	149	-11.30*		-1.17*	
4	0		0		0		0	222	-7.86*		-17.81*	
5	0		0		0		0	82	-7.80*		-6.59*	
6	0		0		0		0	48	-12.00*		-8.48*	
7	0		0		0		0	538	-11.51*		-20.73*	
8	0		1.36		0		-0.21*	1,440	-10.25*		-0.30*	
9	0		0		0		-0.50*	1,440	-8.15*		-13.43*	
10	0		0		0		0	61	-22.51*		-21.09*	
11	0		0		0		0	216	-0.27*		-21.06*	
12	0		0		0		0	455	-0.27*		-19.64*	
13	0		1.49		0		0	78	-10.19*		-19.42*	
14	0		0		0		0	89	-7.69*		-12.72*	
15	0		2.75		0		0	300	-19.10*		-11.96*	
16	0		0		0		0	77	-14.24*		-17.17*	
17	0		0		0		0	787	-9.71*		-16.98*	
18	0		0		0		-0.14*	1,440	-8.35*		-15.10*	
19	0		0		0		0	974	-15.58*		-20.62*	
20	0		0		0		0	292	-8.53*		-20.04*	
$ARPD_{Js}$	0		0.30		0		-0.11		-9.86		-14.68	

7.1 Effectiveness of the Proposed Heuristic

In this section, the solutions of 120 random sequences are determined by the proposed heuristic and the exact algorithm; they are used to calculate the deviation indices: $RPD_{Seq,Js}$ and $ARPD_{Js}$ in Table 4. The Ctime in Table 4 means the computational time (in minutes) required by the exact algorithm. When the exact algorithm fails to obtain the optimal solution within 1,440 minutes (denoted by *), the bound at this stage is used to calculate the indices instead. The negative values of $RPD_{Seq,Js}$ and $ARPD_{Js}$ indicate that the proposed heuristic obtains better solutions than the exact algorithm.

For a given sequence and job size evaluated by $RPD_{Seq,Js}$, the results show that the proposed heuristic obtains optimal solutions or near-to optimal solutions for the 5-job, 10-job, 20-job, and 50-job problems ($RPD_{Seq,Js}$ in the range of -1.41% to 2.75%). The proposed heuristic obtains better solutions than the bounds for the 100-job and 200-job problems (negative $RPD_{Seq,Js}$). For all sequences of a given job size evaluated by $ARPD_{Js}$, the same result direction as explained by $RPD_{Seq,Js}$ is obtained. In terms of computational time perspective, the proposed heuristic can obtain solutions within a second, whereas the exact algorithm requires a significantly longer computational time when dealing with larger job sizes (up to 1,440 minutes). It can be concluded that the proposed heuristic is very efficient to minimise the TP cost for a known sequence of jobs. When the metaheuristics are hybridised with the proposed heuristic, their search ability is improved.

Table 5. $RPD_{Ins,Js}$, $ARPD_{Js}$, and $ARPD_{O}$ indices of GA^{H}

Settings	$RPD_{Ins,Js}$		$ARPD_{Js}$						$ARPD_{O}$
	Min-Max	Std	5-job	10-job	20-job	50-job	100-job	200-job	
GA^{H1}	0-2.07	0.49	0.00	0.00	0.21	0.89	0.04	0.01	0.19 (2)
GA^{H2}	0-1.77	0.39	0.00	0.00	0.21	0.50	0.37	0.02	0.18 (2)
GA^{H3}	0-0.57	0.16	0.00	0.00	0.10	0.29	0.03	0.03	0.07 (1)
GA^{H4}	0-0.48	0.12	0.00	0.00	0.12	0.21	0.05	0.03	0.07 (1)
GA^{H5}	0-0.49	0.12	0.00	0.00	0.00	0.24	0.09	0.16	0.08 (1)
GA^{H6}	0-1.02	0.26	0.00	0.00	0.19	0.31	0.14	0.28	0.15 (2)
GA^{H7}	0-1.60	0.34	0.00	0.00	0.19	0.56	0.08	0.18	0.17 (2)
GA^{H8}	0-1.39	0.33	0.00	0.00	0.19	0.61	0.21	0.23	0.21 (2)
GA^{H9}	0-0.51	0.14	0.00	0.00	0.12	0.24	0.03	0.04	0.07 (1)
GA^{H10}	0-1.06	0.25	0.00	0.00	0.19	0.29	0.03	0.05	0.09 (1)
GA^{H11}	0-1.22	0.23	0.00	0.00	0.24	0.11	0.03	0.05	0.07 (1)
GA^{H12}	**0-0.48**	**0.12**	**0.00**	**0.00**	**0.19**	**0.08**	**0.02**	**0.05**	**0.06 (1)**
GA^{H13}	0-1.09	0.28	0.00	0.00	0.10	0.64	0.21	0.11	0.18 (2)
GA^{H14}	0-1.86	0.41	0.00	0.00	0.24	0.61	0.22	0.12	0.20 (2)
GA^{H15}	0-2.02	0.40	0.00	0.00	0.10	0.58	0.29	0.08	0.18 (2)
GA^{H16}	0-1.22	0.23	0.00	0.00	0.24	0.15	0.18	0.09	0.11 (2)
GA^{H17}	0-1.14	0.36	0.00	0.00	0.19	0.68	0.17	0.63	0.28 (3)
GA^{H18}	0-1.25	0.36	0.00	0.00	0.21	0.70	0.15	0.69	0.29 (3)
GA^{H19}	0-1.14	0.38	0.00	0.00	0.21	0.54	0.17	0.83	0.29 (3)
GA^{H20}	0-1.53	0.51	0.00	0.00	0.37	0.70	0.53	1.30	0.49 (3)
GA^{H21}	0-4.07	0.87	0.00	0.00	0.81	0.94	0.58	0.94	0.55 (3)
GA^{H22}	0-2.80	0.67	0.00	0.00	0.23	1.01	0.58	1.34	0.53 (3)
GA^{H23}	0-3.63	1.14	0.00	0.00	0.19	1.06	0.64	3.09	0.83 (4)
GA^{H24}	0-3.54	1.10	0.00	0.00	0.14	1.14	0.73	2.86	0.81 (4)
GA^{H25}	0-1.22	0.25	0.00	0.00	0.24	0.11	0.07	0.35	0.13 (2)
GA^{H26}	0-0.99	0.29	0.00	0.00	0.19	0.40	0.06	0.37	0.17 (2)
GA^{H27}	0-0.93	0.22	0.00	0.00	0.19	0.12	0.04	0.40	0.12 (2)
GA^{H28}	0-1.06	0.28	0.00	0.00	0.02	0.50	0.05	0.39	0.16 (2)
GA^{H29}	0-1.07	0.27	0.00	0.00	0.00	0.52	0.34	0.25	0.18 (2)
GA^{H30}	0-0.93	0.26	0.00	0.00	0.19	0.51	0.24	0.30	0.21 (2)
GA^{H31}	0-1.22	0.27	0.00	0.00	0.24	0.32	0.24	0.22	0.17 (2)
GA^{H32}	0-0.49	0.17	0.00	0.00	0.10	0.39	0.24	0.23	0.16 (2)

7.2 Determine the BCP Settings of GA^{H}, TS^{H}, and SA^{H}

Tables 5, 6, and 7 show $RPD_{Ins,Js}$, $ARPD_{Js}$, and $ARPD_{O}$ indices for each setting of GA^{H} (GA^{H1}- GA^{H32}), TS^{H} (TS^{H1}- TS^{H8}), and SA^{H} (SA^{H1}- SA^{H32}). The numbers in the parentheses indicate the ranks obtained

by Tukey's multiple comparison test. A lower rank indicates a better parameter setting.

Table 6. $RPD_{Ins,Js}$, $ARPD_{Js}$, and $ARPD_O$ indices of TS^H

Settings	$RPD_{Ins,Js}$		$ARPD_{Js}$						$ARPD_O$
	Min-Max	Std	5-job	10-job	20-job	50-job	100-job	200-job	
TS^{H1}	0-15.00	3.60	0.00	0.00	7.05	0.77	0.02	0.01	1.31 (3)
TS^{H2}	**0-1.94**	**0.40**	**0.00**	**0.00**	**0.59**	**0.19**	**0.05**	**0.01**	**0.14 (1)**
TS^{H3}	0-14.34	2.69	0.00	0.00	3.02	0.28	0.25	0.09	0.61 (2)
TS^{H4}	0-13.46	2.89	0.00	0.00	4.49	0.38	0.19	0.09	0.86 (2)
TS^{H5}	0-9.91	1.90	0.00	0.00	2.82	0.80	0.06	0.01	0.62 (2)
TS^{H6}	0-9.91	1.84	0.00	0.00	2.82	0.28	0.04	0.01	0.52 (2)
TS^{H7}	0-8.25	1.95	0.00	0.00	2.90	0.70	0.17	0.10	0.64 (2)
TS^{H8}	0-1.40	0.86	0.00	0.00	0.45	0.41	0.16	0.11	0.19 (1)

From Table 5, GA^{H12} is the BCP setting of GA^H because it obtained the minimum $ARPD_O$ (in bold numbers). For a given instance and job size, GA^{H12} obtains the solution deviated from its best solution with the $RPD_{Ins,Js}$ values in the range of 0–0.48% and the standard deviation (Std) of 0.12%. For all instances of a given job size, in the 50-job problem, for example, GA^{H12} obtains the solutions deviated from their best solutions with the $ARPD_{Js}$ value of 0.08%. For all instances and job sizes, GA^{H12} obtained the solutions deviated from their best solutions with the $ARPD_O$ value of 0.06% ($ARPD_O$). This clearly shows that the use of the BCP setting for GA^H is very efficient since it can provide the solution with a very small deviation from its best solution. By the BCP setting, only one parameter setting of a given hybrid algorithm can be applied to all tested problem characteristics. It is a more practical solution for the planner rather than the best setting for a given instance and job size.

The same explanation extends to TS^H and SA^H by using their $RPD_{Ins,Js}$, $ARPD_{Js}$, and $ARPD_O$ values shown in Tables 6 and 7. The results show that TS^{H2} and SA^{H24} are the BCP settings for TS^H and SA^H, and the same result direction as GA^{H12} is also obtained. The details of the BCP setting for GA^{H12}, TS^{H2}, and SA^{H24}, along with their $ARPD_O$ and standard deviation values, are shown in Table 8. Note that it is not possible to directly compare the $RPD_{Ins,Js}$, $ARPD_{Js}$, and $ARPD_O$ indices of these algorithms since their best solutions are different.

7.3 Effectiveness of the Proposed Hybrid Algorithms With Their BCP Settings

In this section, the solutions of all tested instances and job sizes are determined by the exact algorithm and the proposed hybrid algorithms set by their BCP settings (GA^{H12}, TS^{H2}, and SA^{H24}). The worst solution is chosen among them and used to calculate the improvement indices: $RPI_{Ins,Js}$ and $ARPI_{Js}$. The results are shown in Table 9. The ranks obtained by Tukey's multiple comparison test are shown in parentheses. A lower rank indicates a better improvement level. The same ranking numbers are considered as insignificant improvement levels. Note that when the exact algorithm cannot obtain the optimal solution within 1,440 minutes (denoted by *), the bound at this stage is used to calculate the improvement indices instead.

In terms of the solution quality of a given instance and job size evaluated by $RPI_{Ins,Js}$, the results are as follows. For the 5-job problem, no significant differences are observed in the solutions between the proposed hybrid algorithms and the exact algorithm ($RPI_{Ins,Js} = 0$). For the 10-job problem, the exact algorithm obtains slightly better solutions than the proposed hybrid algorithms with the $RPI_{Ins,Js}$ values of 2.67% and 2.74% for the instance numbers 1 and 4; no significant differences are

Table 7. $RPD_{Ins,Js}$, $ARPD_{Js}$, and $ARPD_O$ indices of SA^H

Settings	$RPD_{Ins,Js}$		$ARPD_{Js}$						$ARPD_O$
	Min-Max	Std	5-job	10-job	20-job	50-job	100-job	200-job	
SA^{H1}	0-7.14	2.21	0.00	0.00	2.17	5.61	3.67	3.59	2.51 (3)
SA^{H2}	0-7.67	2.33	0.00	0.00	2.39	5.95	3.61	3.52	2.58 (3)
SA^{H3}	0-7.39	2.29	0.00	0.00	2.05	5.91	3.43	3.51	2.48 (3)
SA^{H4}	0-8.53	2.52	0.00	0.00	0.96	6.34	3.75	3.63	2.45 (3)
SA^{H5}	0-7.54	2.23	0.00	0.00	1.50	5.81	3.48	3.52	2.38 (2)
SA^{H6}	0-7.71	2.29	0.00	0.00	2.27	5.90	3.45	3.68	2.55 (3)
SA^{H7}	0-7.24	2.27	0.00	0.00	1.11	5.92	3.29	3.48	2.30 (2)
SA^{H8}	0-7.23	2.10	0.00	0.00	2.03	5.51	3.56	3.56	2.44 (3)
SA^{H9}	0-13.90	4.27	0.00	0.00	4.50	10.57	6.60	7.08	4.79 (4)
SA^{H10}	0-11.81	4.19	0.00	0.00	5.77	10.10	6.62	7.58	5.01 (5)
SA^{H11}	0-13.42	4.27	0.00	0.00	5.43	10.16	6.90	7.28	4.96 (5)
SA^{H12}	0-13.77	4.21	0.00	0.00	5.20	10.58	6.63	7.14	4.92 (5)
SA^{H13}	0-14.30	4.09	0.00	0.00	4.85	10.12	6.67	7.32	4.83 (4)
SA^{H14}	0-14.05	4.41	0.00	0.00	4.50	10.76	6.87	7.59	4.95 (5)
SA^{H15}	0-15.36	4.63	0.00	0.00	5.00	10.80	6.50	7.48	4.96 (5)
SA^{H16}	0-13.79	4.19	0.00	0.00	3.99	10.77	6.87	7.27	4.82 (4)
SA^{H17}	0-0.12	0.04	0.00	0.00	0.00	0.05	0.08	0.03	0.03 (1)
SA^{H18}	0-0.15	0.04	0.00	0.00	0.00	0.06	0.07	0.02	0.02 (1)
SA^{H19}	0-0.73	0.14	0.00	0.00	0.00	0.24	0.07	0.03	0.06 (1)
SA^{H20}	0-0.73	0.13	0.00	0.00	0.00	0.19	0.06	0.02	0.05 (1)
SA^{H21}	0-0.74	0.14	0.00	0.00	0.00	0.19	0.06	0.02	0.04 (1)
SA^{H22}	0-0.13	0.04	0.00	0.00	0.00	0.05	0.06	0.04	0.02 (1)
SA^{H23}	0-0.11	0.03	0.00	0.00	0.00	0.06	0.03	0.02	0.02 (1)
SA^{H24}	**0-0.10**	**0.03**	**0.00**	**0.00**	**0.00**	**0.06**	**0.03**	**0.01**	**0.02 (1)**
SA^{H25}	0-0.19	0.07	0.00	0.00	0.00	0.12	0.11	0.15	0.06 (1)
SA^{H26}	0-0.20	0.07	0.00	0.00	0.00	0.11	0.11	0.15	0.06 (1)
SA^{H27}	0-0.22	0.07	0.00	0.00	0.00	0.09	0.09	0.15	0.06 (1)
SA^{H28}	0-0.18	0.07	0.00	0.00	0.00	0.13	0.10	0.14	0.06 (1)
SA^{H29}	0-0.19	0.07	0.00	0.00	0.00	0.13	0.11	0.15	0.06 (1)
SA^{H30}	0-0.22	0.07	0.00	0.00	0.00	0.13	0.09	0.14	0.06 (1)
SA^{H31}	0-0.19	0.07	0.00	0.00	0.00	0.10	0.12	0.15	0.06 (1)
SA^{H32}	0-0.21	0.07	0.00	0.00	0.00	0.14	0.10	0.14	0.06 (1)

Table 8. Details of GA^{H12}, TS^{H2}, and SA^{H24}

Alg	p_s	Crossover	p_c	Mutation	p_m	NBS	NBO	Temp	NIT	α	TL	$ARPD_O$	Std
GA^{H12}	5	PBX	0.9	SWAP	0.6	-	-	-	-	-	-	0.06	0.12
TS^{H2}	-	-	-	-	-	5	SWAP	-	-	-	50	0.14	0.4
SA^{H24}	-	-	-	-	-	30	INSERT	300	10	0.9	-	0.02	0.03

observed in the solutions for the instance numbers 2, 3, and 5. The proposed hybrid algorithms obtain significantly better solutions than bounds from the exact algorithm with the $RPI_{Ins,Js}$ values in the range of 6.29–59.78% for the 20-job problem, 21.46–94.92% for the 50-job problem, 93.41–98.45% for the 100-job problem, and 95.82–96.64% for the 200-job problem.

Table 9. $RPI_{Ins,Js}$ and $ARPI_{Js}$ indices of GA^{H12}, TS^{H2}, SA^{H24}, and the exact algorithm

Algorithms	$RPI_{Ins,Js}$					
GA^{H12}	**5-job**	**10-job**	**20-job**	**50-job**	**100-job**	**200-job**
Instance 1	0.00 (1)	0.00 (2)	17.87 (1)	70.32 (1)	98.11 (1)	95.70 (1)
Instance 2	0.00 (1)	0.00 (1)	36.76 (1)	38.36 (1)	98.40 (1)	95.87 (1)
Instance 3	0.00 (1)	0.00 (1)	7.16 (1)	21.46 (1)	98.45 (1)	96.06 (1)
Instance 4	0.00 (1)	0.00 (2)	58.36 (1)	55.03 (1)	93.43 (1)	96.63 (1)
Instance 5	0.00 (1)	0.00 (1)	59.78 (1)	94.92 (1)	96.08 (1)	95.83 (1)
$ARPI_{Js}$	0.00 (1)	0.00 (2)	35.99 (1)	56.02 (1)	96.89 (1)	96.02 (1)
TS^{H2}	**5-job**	**10-job**	**20-job**	**50-job**	**100-job**	**200-job**
Instance 1	0.00 (1)	0.00 (2)	17.87 (1)	70.24 (1)	98.11 (1)	95.71 (1)
Instance 2	0.00 (1)	0.00 (1)	36.58 (1)	38.32 (1)	98.39 (1)	95.88 (1)
Instance 3	0.00 (1)	0.00 (1)	6.29 (1)	21.60 (1)	98.45 (1)	96.06 (1)
Instance 4	0.00 (1)	0.00 (2)	57.56 (1)	54.85 (1)	93.42 (1)	96.64 (1)
Instance 5	0.00 (1)	0.00 (1)	59.70 (1)	94.88 (1)	96.08 (1)	95.83 (1)
$ARPI_{Js}$	0.00 (1)	0.00 (2)	35.60 (1)	55.98 (1)	96.89 (1)	96.02 (1)
SA^{H24}	**5-job**	**10-job**	**20-job**	**50-job**	**100-job**	**200-job**
Instance 1	0.00 (1)	0.00 (2)	17.87 (1)	70.28 (1)	98.11 (1)	95.70 (1)
Instance 2	0.00 (1)	0.00 (1)	37.03 (1)	38.33 (1)	98.39 (1)	95.87 (1)
Instance 3	0.00 (1)	0.00 (1)	7.16 (1)	21.55 (1)	98.44 (1)	96.05 (1)
Instance 4	0.00 (1)	0.00 (2)	58.56 (1)	55.00 (1)	93.41 (1)	96.63 (1)
Instance 5	0.00 (1)	0.00 (1)	59.78 (1)	94.92 (1)	96.07 (1)	95.82 (1)
$ARPI_{Js}$	0.00 (1)	0.00 (2)	36.08 (1)	56.01 (1)	96.89 (1)	96.01 (1)
Exact algorithm	**5-job**	**10-job**	**20-job**	**50-job**	**100-job**	**200-job**
Instance 1	0.00 (1)	2.67 (1)	0.00* (2)	0.00* (2)	0.00* (2)	0.00* (2)
Instance 2	0.00 (1)	0.00* (1)	0.00* (2)	0.00* (2)	0.00* (2)	0.00* (2)
Instance 3	0.00 (1)	0.00 (1)	0.00* (2)	0.00* (2)	0.00* (2)	0.00* (2)
Instance 4	0.00 (1)	2.74 (1)	0.00* (2)	0.00* (2)	0.00* (2)	0.00* (2)
Instance 5	0.00 (1)	0.00 (1)	0.00* (2)	0.00* (2)	0.00* (2)	0.00* (2)
$ARPI_{Js}$	0.00 (1)	1.08 (1)	0.00 (2)	0.00 (2)	0.00 (2)	0.00 (2)

Based on the $RPI_{Ins,Js}$ index, it is observed that GA^{H12}, TS^{H2}, and SA^{H24} obtain insignificant improvement levels among themselves for all tested instances and job sizes (same ranking numbers). It is concluded that the solutions obtained from GA^{H12}, TS^{H2}, and SA^{H24} are almost the same. In addition, the instance number 3 of the 20-job and 50-job problems could be considered as hard instances since the exact algorithm could not determine any optimal solutions within 1,440 minutes; GA^{H12}, TS^{H2}, and SA^{H24} can improve the bounds from the exact algorithm in the range of 6.29–21.60%.

Regarding all instances of a given job size evaluated by $ARPI_{Js}$, no improvement levels are observed for the 5-job problem ($ARPI_{Js} = 0$). The exact algorithm improves the solutions obtained from the hybrid algorithms by 1.08% for the 10-job problem. On the other hand, the hybrid algorithms improve the bounds from the exact algorithm with the truncated $ARPI_{Js}$ values of 35% for the 20-job problem, 55% for the 50-job problem, and 96% for both 100-job and 200-job problems, respectively. It is concluded that the hybrid algorithms clearly outperform the exact algorithm when applied to medium-scale and large-scale problems (20-job, 50-job, 100-job, and 200-job problems), whereas both of them work well for small-scale problems (5-job and 10-job problems).

The hybrid algorithms are also very efficient in terms of the computational time as shown in Table 10. They converge to stable solutions within the range of 1–120 minutes for all instances and job sizes. It is very fast when compared regardless of the allowable computational time from the planner (360 minutes) or the computational time for the exact algorithm (1,440 minutes). The exact algorithm obtains optimal solutions very fast only for all instances of the 5-job problem, but it cannot determine the optimal solutions within 1,440 minutes for larger job sizes. Comparing the computational times among GA^{H12}, TS^{H2}, and SA^{H24}, it is obvious that TS^{H2} requires the shortest computational time across the entire tested problem characteristics (rank 1).

The most appropriate hybrid algorithm for minimising the TP cost of the SMO problem characteristics in this study should be considered from three perspectives: the solution quality, the computational time, and the complexity of the proposed metaheuristics. Based on the three perspectives, TS^{H2} is recommended since it obtains rank 1 for the first two aspects, and it is the simplest algorithm compared to GA^{H12} and SA^{H24}.

Table 10. Computational time of GA^{H12}, TS^{H2}, SA^{H24}, and the exact algorithm

Algorithms	Computational time to a stable solution (minutes)					
	5-job	10-job	20-job	50-job	100-job	200-job
GA^{H12}	< 1 (1)	< 1 (1)	< 1 (1)	3 (1)	22 (2)	120 (2)
TS^{H2}	< 1 (1)	< 1 (1)	< 1 (1)	< 1 (1)	6 (1)	97 (1)
SA^{H24}	< 1 (1)	< 1 (1)	< 1 (1)	25 (2)	59 (3)	93 (1)
Exact algorithm	< 1 (1)	320 (2)	1,440 (2)	1,440 (3)	1,440 (4)	1,440 (3)

8. CONCLUSION

In this study, three novel hybrid algorithms are developed for solving a single machine scheduling problem with an overtime constraint. The objective is to minimise the total penalty (TP) cost defined as the sum of tardy, early, and overtime costs. The concept of the proposed hybrid algorithms is to hybridise a new heuristic that is capable of minimising the TP cost for a known sequence, with genetic algorithm, tabu search, and simulated annealing, referred to as GA^{H}, TS^{H}, and SA^{H}, to increase their search ability.

The mechanism of the proposed heuristic to minimise the TP cost is a backward-forward scheduling technique with a penalty cost trade-off process. The effectiveness of the proposed heuristic is evaluated by using a set of 120 sequences (randomly selected with various job sizes). The results show that the proposed heuristic is very efficient since it obtains the solutions that have very small deviations from the optimal solutions for small-scale problems, and it obtains significantly better solutions than the bounds from the exact algorithm for medium-scale and large-scale problems. In addition, the computational time of the proposed heuristic is less than a second while the exact algorithm requires a significantly longer.

Since there are many parameters in GA^{H}, TS^{H}, and SA^{H}, the best common parameter (BCP) setting of each hybrid algorithm applicable to all instances and job sizes are preferred rather than the best setting for an individual instance and job size. The results show that GA^{H12}, TS^{H2}, and SA^{H24}, are the BCP setting of GA^{H}, TS^{H}, and SA^{H}. They provide very excellent solutions that slightly deviate from their best solutions.

The effectiveness of GA^{H12}, TS^{H2}, and SA^{H24}, is evaluated by using the SMO problem characteristics possibly found in the selected factories. The evaluation is performed based on the average relative percentage improvement over the worst solution. The results show that GA^{H12}, TS^{H2}, and SA^{H24} significantly improve the worst solution for medium-scale and large-scale problems while they all obtain the optimal solutions for small-scale problems. It is also observed that GA^{H12}, TS^{H2}, and SA^{H24} obtain insignificant improvement levels.

The computational times of GA^{H12}, TS^{H2}, and SA^{H24}, are in the range of 1-120 minutes, which is significantly shorter than the practical limit specified by the planner (360 minutes) and the computational time required by the exact algorithm (1,440 minutes). It is observed that TS^{H2} requires the shortest computational time compared to GA^{H12} and SA^{H24}.

From the solution quality, computational time, and complexity of the algorithm perspectives, TS^{H2} is the most recommended for the SMO problem in this study since it is the best in all perspectives. The proposed hybrid algorithm is developed as a software package freely downloadable from https://sites.google.com/view/scheduling/smo.

There are some suggestions for further study. The proposed heuristic should be further developed and applied to other production shops (such as flow shop, job shop, and their variants) and other production characteristics (SDST, pre-emptive, no-wait scheduling, and so on). Furthermore, it is suggested to hybrid with other metaheuristics such as iterated local search (ILS), ant colony optimisation (ACO), particle swarm optimisation (PSO), and so on, to investigate their search ability. Therefore, further research is expected to be conducted in order to address these suggestions.

ACKNOWLEDGMENT

The authors express their sincere gratitude to anonymous reviewers for their valuable comments. The gratitude is also given to King Mongkut's University of Technology North Bangkok (KMUTNB) for the funding (KMUTNB-64-DRIVE-18) and to Rajamangala University of Technology Phra Nakhon (RMUTP) for all of their support.

REFERENCES

Al-Moadhen, A. A., Packianather, M., Setchi, R., & Qiu, R. (2016). Robot task planning in deterministic and probabilistic conditions using semantic knowledge base. *International Journal of Knowledge and Systems Science*, *7*(1), 56–77. doi:10.4018/IJKSS.2016010104

Arroyo, J. E. C., Ottoni, R. D. S., & Oliveira, A. D. P. (2011). Multi-objective variable neighbourhood search algorithms for a single machine scheduling problem with distinct due windows. *Electronic Notes in Theoretical Computer Science*, *281*, 5–19. doi:10.1016/j.entcs.2011.11.022

Chang, P. C., Chen, S. H., & Mani, V. (2009). A hybrid genetic algorithm with dominance properties for single machine scheduling with dependent penalties. *Applied Mathematical Modelling*, *33*(1), 579–596. doi:10.1016/j.apm.2008.01.006

Cheng, T. C. E., Kang, L. Y., & Ng, C. T. (2005). Single machine due-date scheduling of jobs with decreasing start-time dependent on the processing times. *International Transactions in Operational Research*, *12*(3), 355–366. doi:10.1111/j.1475-3995.2005.501_1.x

Chiadamrong, N., & Tangchaisuk, C. (2021). Hybrid simulation-based optimization for production planning of a dedicated remanufacturing system. *International Journal of Knowledge and Systems Science*, *12*(3), 53–79. doi:10.4018/IJKSS.2021070103

Ding, J., Lü, Z., Cheng, T. C. E., & Xu, L. (2016). Breakout dynasearch for the single-machine total weighted tardiness problem. *Computers & Industrial Engineering*, *98*, 1–10. doi:10.1016/j.cie.2016.04.022

Du, J., & Leung, J. Y.-T. (1990). Minimizing total tardiness on one machine is NP-Hard. *Mathematics of Operations Research*, *15*(3), 483–495. doi:10.1287/moor.15.3.483

Ferrolho, A., & Crisóstomo, M. (2007). Single machine total weighted tardiness problem with genetic algorithms. In *IEEE/ACS International Conference on Computer Systems and Applications* (pp. 1-8). doi:10.1109/AICCSA.2007.370857

Geiger, M. J. (2010). On the heuristic search for the single machine total weighted tardiness problem - some theoretical insights and their empirical verification. *European Journal of Operational Research*, *207*(3), 1235–1243. doi:10.1016/j.ejor.2010.06.031

González, M. A., & Vela, C. R. (2015). An efficient memetic algorithm for total weighted tardiness minimization in a single machine with setups. *Applied Soft Computing*, *37*, 506–518. doi:10.1016/j.asoc.2015.07.050

Györgyi, P., & Kis, T. (2018). Minimizing the maximum lateness on a single machine with raw material constraints by branch-and-cut. *Computers & Industrial Engineering*, *115*, 220–225. doi:10.1016/j.cie.2017.11.016

Herr, O., & Goel, A. (2016). Minimising total tardiness for a single machine scheduling problem with family setups and resource constraints. *European Journal of Operational Research*, *248*(1), 123–135. doi:10.1016/j.ejor.2015.07.001

Hewahi, N. M. (2015). Particle swarm optimization for Hidden Markov Model. *International Journal of Knowledge and Systems Science*, *6*(2), 1–15. doi:10.4018/IJKSS.2015040101

Jaramillo, F., & Erkoc, M. (2017). Minimizing total weighted tardiness and overtime costs for single machine preemptive scheduling. *Computers & Industrial Engineering*, *107*, 109–119. doi:10.1016/j.cie.2017.03.012

Jaramillo, F., & Erkoc, M. (2018). Modeling single machine preemptive scheduling to minimize the cost of tardiness and overtime. SSRN *Electronic Journal*. https://ssrn.com/abstract=3197262 10.2139/ssrn.3197262

Keshavarz, T., Savelsbergh, M., & Salmasi, N. (2015). A branch-and-bound algorithm for the single machine sequence-dependent group scheduling problem with earliness and tardiness penalties. *Applied Mathematical Modelling*, *39*(20), 6410–6424. doi:10.1016/j.apm.2015.01.069

Khorshidian, H., Javadian, N., Zandieh, M., Rezaeian, J., & Rahmani, K. (2011). A genetic algorithm for JIT single machine scheduling with preemption and machine idle time. *Expert Systems with Applications*, *38*(7), 7911–7918. doi:10.1016/j.eswa.2010.10.066

Kirlik, G., & Oguz, C. (2012). A variable neighbourhood search for minimizing total weighted tardiness with sequence dependent setup times on a single machine. *Computers & Operations Research*, *39*(7), 1506–1520. doi:10.1016/j.cor.2011.08.022

Latthawanichphan, J., Songserm, W., & Wuttipornpun, T. (2019). An instance generator for scheduling problems featuring options for unequal stages and unequal parallel machines. *International Journal of Technology and Engineering Studies*, *5*(4), 106–112. doi:10.20469/ijtes.5.10001-4

Li, X., Ventura, J. A., & Bunn, K. A. (2021). A joint order acceptance and scheduling problem with earliness and tardiness penalties considering overtime. *Journal of Scheduling*, *24*(1), 49–68. doi:10.1007/s10951-020-00672-5

Li, Z., Chen, H., Xu, R., & Li, X. (2015). Earliness–tardiness minimization on scheduling a batch processing machine with non-identical job sizes. *Computers & Industrial Engineering*, *87*, 590–599. doi:10.1016/j.cie.2015.06.008

Low, C., Li, R. K., & Wu, G. H. (2016). Minimizing total earliness and tardiness for common due date single-machine scheduling with an unavailability interval. *Mathematical Problems in Engineering*, *2016*, 1–12. doi:10.1155/2016/6124734

M'Hallah, R., & Alhajraf, A. (2016). Ant colony systems for the single-machine total weighted earliness tardiness scheduling problem. *Journal of Scheduling*, *19*(2), 191–205. doi:10.1007/s10951-015-0429-x

Melouk, S., Damodaran, P., & Chang, P. Y. (2004). Minimizing makespan for single machine batch processing with non-identical job sizes using simulated annealing. *International Journal of Production Economics*, *87*(2), 141–147. doi:10.1016/S0925-5273(03)00092-6

Molaee, E., Moslehi, G., & Reisi, M. (2011). Minimizing maximum earliness and number of tardy jobs in the single machine scheduling problem with availability constraint. *Computers & Mathematics with Applications (Oxford, England)*, *62*(9), 3622–3641. doi:10.1016/j.camwa.2011.09.016

Nazif, H., & Lee, L. S. (2010). Solving single machine scheduling problem with maximum lateness using a genetic algorithm. *Journal of Mathematics Research*, *2*(3), 57–62. doi:10.5539/jmr.v2n3p57

Nesello, V., Subramanian, A., Battarra, M., & Laporte, G. (2018). Exact solution of the single-machine scheduling problem with periodic maintenances and sequence-dependent setup times. *European Journal of Operational Research*, *266*(2), 498–507. doi:10.1016/j.ejor.2017.10.020

Pan, Q., Ruiz, R., & Alfaro-Fernández, P. (2017). Iterated search methods for earliness and tardiness minimization in hybrid flowshops with due windows. *Computers & Operations Research*, *80*, 50–60. doi:10.1016/j.cor.2016.11.022

Perez-Gonzalez, P., & Framinan, J. M. (2018). Single machine scheduling with periodic machine availability. *Computers & Industrial Engineering*, *123*, 180–188. doi:10.1016/j.cie.2018.06.025

Rabadi, G., Anagnostopoulos, G., & Mollaghasemi, M. (2002). A simulated annealing algorithm for a scheduling problem with setup times. In *IIE Annual Conference. Proceedings*. Institute of Industrial Engineers-Publisher.

Rerkjirattikal, P., Huynh, V., Olapiriyakul, S., & Supnithi, T. (2020a). A goal programming approach to nurse scheduling with individual preference satisfaction. *Mathematical Problems in Engineering*, *2020*, 2379091. doi:10.1155/2020/2379091

Rerkjirattikal, P., & Olapiriyakul, S. (2019). Overtime assignment and job satisfaction in noise-safe job rotation scheduling. In *International Symposium on Integrated Uncertainty in Knowledge Modelling and Decision Making (IUKM 2019)* (pp. 26-37). Springer. doi:10.1007/978-3-030-14815-7_3

Rerkjirattikal, P., Wanwarn, T., Starita, S., Huynh, V., Supnithi, T., & Olapiriyakul, S. (2020b). Heuristics for noise-safe job-rotation problems considering learning-forgetting and boredom-induced job dissatisfaction effects. *Environmental Engineering and Management Journal*, *19*(8), 1325–1337. doi:10.30638/eemj.2020.126

Sioud, A., Gravel, M., & Gagné, C. (2012). A hybrid genetic algorithm for the single machine scheduling problem with sequence-dependent setup times. *Computers & Operations Research*, *39*(10), 2415–2424. doi:10.1016/j.cor.2011.12.017

Songserm, W., Latthawanichphan, J., Wuttipornpun, T., & Sukkerd, W. (2021). An improvement of instance generator featuring assembly operations with parallel machines for multi-level and multi-operation scheduling problems. In 2021 Research, Invention, and Innovation Congress (RI2C) (pp. 347-356). IEEE. doi:10.1109/RI2C51727.2021.9559812

Srizongkhram, S., Manitayakul, K., Suthamanondh, P., & Chiadamrong, N. (2020). Fuzzy chance-constrained integer programming models for portfolio investment selection and optimization under uncertainty. *International Journal of Knowledge and Systems Science*, *11*(3), 33–58. doi:10.4018/IJKSS.2020070103

Süer, G. A., Yang, X., Alhawari, O. I., Santos, J., & Vazquez, R. (2012). A genetic algorithm approach for minimizing total tardiness in single machine scheduling. *International Journal of Industrial Engineering and Management*, *3*(3), 163–171.

Suppiah, Y., & Omar, M. K. (2014). A hybrid tabu search for batching and sequencing decisions in a single machine environment. *Computers & Industrial Engineering*, *78*, 135–147. doi:10.1016/j.cie.2014.10.010

Vakhania, N. (2018). Scheduling a single machine with primary and secondary objectives. *Algorithms*, *11*(6), 1–16. doi:10.3390/a11060080

Valente, J. M. S., & Gonçalves, J. F. (2009). A genetic algorithm approach for the single machine scheduling problem with linear earliness and quadratic tardiness penalties. *Computers & Operations Research*, *36*(10), 2707–2715. doi:10.1016/j.cor.2008.11.016

Vallada, E., & Ruiz, R. (2009). Cooperative metaheuristics for the permutation flowshop scheduling problem. *European Journal of Operational Research*, *193*(2), 365–376. doi:10.1016/j.ejor.2007.11.049

Vilà, M., & Pereira, J. (2013). Exact and heuristic procedures for single machine scheduling with quadratic earliness and tardiness penalties. *Computers & Operations Research*, *40*(7), 1819–1828. doi:10.1016/j.cor.2013.01.019

Yang, B., Geunes, J., & O'Brien, W. J. (2004). A heuristic approach for minimizing weighted tardiness and overtime costs in single resource scheduling. *Computers & Operations Research*, *31*(8), 1273–1301. doi:10.1016/S0305-0548(03)00080-7

Yazdani, M., Khalili, S. M., Babagolzadeh, M., & Jolai, F. (2017). A single-machine scheduling problem with multiple unavailability constraints: A mathematical model and an enhanced variable neighborhood search approach. *Journal of Computational Design and Engineering*, *4*(1), 46–59. doi:10.1016/j.jcde.2016.08.001

Yoda, M., Eguchi, T., & Murayama, T. (2014). Job shop scheduling for meeting due dates and minimizing overtime using genetic algorithm incorporating new priority rules. *Journal of Advanced Mechanical Design, Systems and Manufacturing*, *8*(5), 1–10. doi:10.1299/jamdsm.2014jamdsm0071

Zhang, Z., & Guo, C. (2011). Association rules evaluation by a hybrid multiple criteria decision method. *International Journal of Knowledge and Systems Science*, *2*(3), 14–25. doi:10.4018/jkss.2011070102

Zhu, H., & You, Y. (2017). Note on single machine scheduling problems with resource dependent release times. In *2017 3rd International Conference on Information Management* (pp. 190-194). doi:10.1109/INFOMAN.2017.7950373

Zobolas, G. I., Tarantilis, C. D., & Ioannou, G. (2008). Extending capacity planning by positive lead times and optional overtime, earliness and tardiness for effective master production scheduling. *International Journal of Production Research*, *46*(12), 3359–3386. doi:10.1080/00207540601008374