Hindawi
Modelling and Simulation in Engineering
Volume 2021, Article ID 6663375, 19 pages
https://doi.org/10.1155/2021/6663375

Research Article

Biobjective Scheduling for Joint Parallel Machines with Sequence-Dependent Setup by Taking Pareto-Based Approach

Wichai Srisuruk,[1] **Kanchala Sudtachat**,[1] **and Paramate Horkaew**[2]

[1]*School of Manufacturing Engineering, Suranaree University of Technology, Nakhon Ratchasima, Thailand*
[2]*School of Computer Engineering, Suranaree University of Technology, Nakhon Ratchasima, Thailand*

Correspondence should be addressed to Wichai Srisuruk; wichai@sut.ac.th

Received 27 November 2020; Revised 5 August 2021; Accepted 17 August 2021; Published 30 August 2021

Academic Editor: Chen Lin Soo

Modern factories have been moving toward just-in-time manufacturing paradigm. Optimal resource scheduling is therefore essential to minimize manufacturing cost and product delivery delay. This paper therefore focuses on scheduling multiple unrelated parallel machines, via Pareto approach. With the proposed strategy, additional realistic concerns are addressed. Particularly, contingencies regarding product dependencies as well as machine capacity and its eligibility are also considered. Provided a jobs list, each with a distinct resource work hour capacity, this novel scheduling is aimed at minimizing manufacturing costs, while maintaining the balance of machine utilization. To this end, different computational intelligence algorithms, i.e., adaptive nearest neighbour search and modified tabu search, are employed in turn and then benchmarked and validated against combinatorial mathematical baseline, on both small and large problem sets. The experiments reported herein were made on MATLAB™ software. The resultant manufacturing plans obtained by these algorithms are thoroughly assessed and discussed.

1. Introduction

With the recent advances in modern intelligent manufacturing, most industrial works have increasingly been adopting just-in-time (JIT) strategy [1, 2]. With this strategy, manufacturing cost and delivery delay are optimized by means of meticulous production planning. Among prevailing machining approaches presently taken by modern factories, unrelated parallel machine (UPM) [3–6] system is investigated in this study. In the UPM system, a factory consists of several machines operating the same task but taking different time durations. Examples of these factories are lathes and sawmills. Upon commencing any task or restarting an alternate one, an operator often has to prepare the machine by making appropriate configurations and settings. They include fitting new moulds, replacing equipped tools, and cleaning contaminated parts. These activities generally incur additional cost, also known as setting up cost. They are expressed in terms of spent time (and/or money), whose values may be constant or varied as the preceding task. More specifically, a sequence-independent setup (SIS) [7, 8] remains constant regardless of the previous tasks operated on the same machine, whereas a sequence-dependent setup (SDS) does not [4, 5, 9, 10]. In addition, while some tasks can well be performed on one machine, they may be prohibitive on others. For instance, coarse milling may be performed on all available machines, while fine milling is only possible on specific ones. Another concern faced in typical industrial practices is unplanned maintenance (UM) [11] due to faulty resources, especially after a schedule has been issued.

Provided a list of required productions, i.e., jobs, each with associated resource work hour capacity, this paper is aimed at devising an appropriate JIT manufacturing plan. Without loss of generalization ability, UPMs involved in this study were assumed to be of SDS type, where all setups could be made equal, otherwise. It took into account production variables, typically found in practices, e.g., time and money required to make an initial machine configuration, given precedent tasks, machine standard time, and storage costs

for products completely in order to meet requested demand each period (all of which were in integers). The optimal machine scheduling was determined such that the resultant plan incurred minimum manufacturing cost, while maintaining machine utilization balance. In this study, it was further assumed that once started, a task could not be overridden or cancelled, until it was fully completed. To this end, the state-of-the-art computational intelligence algorithms, that is, adaptive nearest neighbour search (ANNS) [12, 13] and modified tabu search (MTS) [14–17], were employed and assessed in turns. The resultant scheduling for small and large problems was subsequently validated against an exhaustive baseline model.

This paper focuses on biobjective unrelated parallel machine scheduling. Its emphasis is placed on minimizing manufacturing costs, while maintaining the balance of machine utilization, based on Pareto optimality. Its main contribution is to remedy prohibitively high complexity of theoretical models by employing heuristic and metaheuristic approaches, namely, ANNS and MTS. To elucidate its merit, realistic instances of JIT manufacturing environments with practical conditions were explored in simulations.

The remaining of this paper is organized as follows. Section 2 reviews the literature and the related works. It provides detailed accounts and critical discussion on machine scheduling problems and state-of-the-art solutions. Section 3 describes the proposed scheme by first outlining its key processes, followed by their description and assumptions made herein. Section 4 describes the experiments on the abovementioned algorithms, including the characteristics of data involved and relevant assessments. Subsequently, this section also demonstrates the merits of the proposed scheme by objectively reporting and discussing the resultant scheduling, based on designated performance metrics. Finally, Section 5 makes the concluding remarks on the contribution of this study and its prospects.

2. Literature Review

This section focuses on recent research on conditional and constrained scheduling. In order to devise an optimal manufacturing plan that minimizes its overall costs, including those incurred by configuring the machines during the production process and by inventory storage, various approaches have been taken in the literature. They can be categorized into those based on solving for an exact solution of some mathematical model and on approximating one by computational intelligent techniques. The following subsections start from background on scheduling theory. After that, the definitions of parallel machine scheduling and its mathematical model are described. On solving such a model, various optimization strategies, both with single and multiple objectives, are subsequently reviewed. Finally, the recent and closely related research works to the proposed scheme are discussed.

2.1. Background on Scheduling. Scheduling is one of the management schemes that attempts to allocate limited resources for completing a mission within given timeframe, possibly under some constraints [18]. In an industrial context, a preferable solution to this problem is determined by optimizing resources' utilization (i.e., manpower and machines) with respect to predefined objectives [6, 19, 20]. This paper proposed a novel solution to a scheduling problem of a manufacturing system, which was characterized as follows.

The system consists of a set of n jobs and m machines, denoted by $I = \{i \mid i \in [1, n]\}$, $J = \{j \mid j \in [1, n]\}$, and $M = \{m \mid m \in [1, MC]\}$, where i and j are the indices of job and m is the indices of machine, respectively. The notations of key parameters involved in the analyses are listed in Table 1.

Scheduling strategies can be divided into flow shop and job shop scheduling [21]. The former consists of consecutive machines or job stations, and each performs different operations. All the jobs are processed by each machine in an exact same sequence, called flow, whereas in the latter group, scheduling is made for each job, whose flow consists of a unique sequence of operations. On solving a scheduling problem, there exist three techniques normally adopted. First, a mathematical model can be used to find an optimal scheduling by means of, for instance, integer, mixed integer, or dynamic programming [22–24]. Objectives typically considered in the optimization are flow time, makespan, lateness, number of tardy jobs, tardiness, etc. [25–27]. As the problem size increases, however, they suffer from excessive complexity. Second, dispatching rules or heuristic techniques are especially designed to reduce complexity while offering acceptable results within reasonable time. Given a set of jobs, these techniques imposed one or more criteria, e.g., first come first served (FCFS), shortest processing time (SPT), longest processing time (LPT), and earliest due date (EDD) [25, 28]. Techniques in this group also consider similar objectives that do those in the former one. Third, neighbourhood search finds an optimal solution more efficiently, especially for larger problems, by incrementally updating the current best solution, taking into account information from its neighbours. They include tabu search, simulated annealing, and genetic algorithms [14, 29, 30].

Furthermore, scheduling also differs by machine layouts. Single machine scheduling is trivial and usually adopted in decomposition of a more complex manufacturing. On the other hand, for the parallel machines, single-process jobs are released to a pool of machines working in parallel. The scheduling problem is now reduced to making two decisions, i.e., job allocation and its sequence. This type of scheduling can be further divided by the functions of associated machines, which are identical (IPM), nonidentical (NPM), and unrelated parallel machine (UPM) [3–6]. This research focuses on UPM, where each machine can process a given job at a different speed. Since a machine can process different jobs, when starting a new one, a setting up is usually needed [7–10]. Therefore, time and cost used in this operation depend on involved jobs and their sequence.

2.2. Mathematical Models of Parallel Machine Scheduling. Provided a problem of scheduling a set of n jobs (I and J) onto m parallel machines (M), as described in Section 2.1, a mathematical model attempts to plan an optimal sequence

Table 1: Nomenclature of generic scheduling parameters.

Parameter	Notation
Release time	R_i
Starting time	St_i
Waiting time	W_i
Setup time	S_{ij}
Processing time	P_{im}
Completion time	C_i
Idle time	I_m
Flow time	F_i
Makespan	$C_{\max}$
Weight	w_i
Due date	d_i
Lateness	L_i
Earliness and tardiness	E_i and T_i

Note: the subscripts i, j denote job and m denote machine indices, respectively.

of job assignments with respect to some criteria. In the present context, two conditions are imposed on the system. That is, some jobs are not assignable to some machines and each job must be assignable to at least one machine. In addition, since a sequence-dependent setup was assumed, a setup time $S(i, j) \geq 0$, used in preparing for a job j after having completed a job i, was considered. Let, at the beginning for a job i, there be no setup involved, that is, $S(0, i) = 0$. Then, an optimal scheduling that minimizes the overall makespan may be posed as an integer programming problem. Following a logistics model proposed by Tong et al. [31], we adopted an objective that minimizes the total processing, storage, and setup costs for all jobs and periods, i.e.,

$$Z = \sum_{j=1}^{n}\sum_{t=1}^{T} g_{jt}C_{jt} + \sum_{j=1}^{n}\sum_{t=1}^{T} o_{jt}H_{jt} + \sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{m=1}^{M}\sum_{t=1}^{T} s_{ij}X_{ijmt}, \quad (1)$$

where g_{jt} and C_{jt} are the unit processing cost and counts of job j at period t, o_{jt} and H_{jt} are unit inventory cost and stored job j at period t, s_{ij} is a cost of setting up a machine for job j after i, and X_{ijmt} is 1 when a job j is processed right after a job i on machine m, at period t or 0 otherwise. Detailed descriptions of problem-specific constraints will be given in Section 3.1.

2.3. Optimization Strategies. With the emergence of modern computer and information technology, optimization has been incorporated into an extensive range of applications, from engineering [32, 33], medicine [34], geographical science [35], to finance [36]. In industrial engineering (IE), an optimization strategy is normally employed in production and logistics planning [18]. Enumerative search, for example, is found applied to solve an exact solution to scheduling or assignment problems. The most prevailing methods in this group include dynamic, linear, and quadratic integer programming [22–24]. Thus far, these methods are only suitable for small problem, because their complexity is considered *NP*-Hard [37] and thus exponentially increases as the problem size. To remedy complexity issue, approximation-optimization approach is proposed and normally called metaheuristics. Methods in this group approximate an *NP*-Hard solution by imitating adaptation of some entities found in nature. They are simulated annealing (SA) [29], tabu search (TS) [14–17], neural network (NN) [38], genetic algorithm (GA) [31, 39], and ant-colony optimization (ACO) [40]. This approach, however, is considered a black-box and thus fails to give any insights into the problem at hand and its behaviour. Experiences on the problem can be incorporated into optimization. This approach, called heuristic optimization, can be divided into 2 groups. With incremental generation, solutions are gradually formed into a final solution, while the neighbourhood search strategy a solution that is not contradicting with predefined conditions is created in turn and, if it gives better results than a previous one, is maintained. Some methods in this group worth mentioning here are the nearest insertion, Petal, sweep, cluster first route second, and route first cluster second [41–43]. With the heuristic optimization, a solution is obtained via a process of learning by investigation, assessing the feedback and making a decision accordingly. Thus far, when experiences are imposed on a problem, the resultant strategy may not be suitable or even applicable to unseen ones. Although they do not guarantee the best solution, their key advantages are much simple simulation, while offering a "good enough" solution to a problem that may not be well structured.

Thanks to those preferable characteristics, the heuristic approach has been taken in many multidisciplinary engineering applications. A widely employed method, called variable neighbourhood search (VNS), was first proposed by Hansen and Mladenović [44] and applied to several complex problems [45, 46]. Its main strategy is avoiding local traps by adjusting neighbourhood structures (NS), if a newly found solution is worse than an existing one or trapped in local minima. It was shown therein that the adjustment increases the likelihood of finding a better solution than local updates without one. Moreover, its implementation is trivial, and it requires only a few parameters. More specifically, given a starting point, Q, VNS defines a searching domain of order $n \in [1, n_{\max}]$, within its neighbourhood, $\mathrm{NS}(Q)$. It then randomly finds an improved solution (shaking), Q', within that domain. Subsequently, it searches for a local optimum Q'', around Q'. This can be done by, for example, perturbation, basic (BLS), or iterated local search (ILS). If this gives a better result, then the optimal solution is updated $Q \longleftarrow Q''$. Otherwise, the domain order is incremented, $n \longleftarrow n + 1$. Since it was proposed, VNS has been applied to both linear and nonlinear integer and mixed-integer programming [47].

Tabu search is a hybrid discrete optimization strategy that relies on neighbourhood search and tabu list (TL), used to store previous solutions. TL can prevent getting locked in local trap by aspiration criteria (AC). Its search strategy is

deterministic and specified by recency and frequency conditions. In addition, the performance can be enhanced by two mechanisms, i.e., intensification and diversification. Provided a search space and radius (R), counter, a TL, and termination criteria (TC), tabu search algorithm starts by randomly picking an initial solution, $S0$ within the search space. It then randomly gathers N neighbours within radius R around $S0$ and stores in a set $S1$ (R). An objective function is evaluated in turn for each point in this set, and the one that gave the best (minimum cost) solution is marked as $S1$. If $S1 \leq S0$, the previous solution, $S0$ is stored in the TL and the solution is updated, $S0 \longleftarrow S1$. Otherwise, $S1$ is stored in the TL. The process is repeated until TC are met, and the optimal solution is the current $S0$. Basic TS is rather slow and can get trapped in local minima. Adaptive tabu search (ATS) that incorporates backtracking and adaptive radius mechanisms was proposed to elevate the issues [48]. With ATS, after a new solution is updated, backtracking is invoked when it is locked by local solution. Search radius is also adapted as it reaches convergence.

On solving biobjective scheduling problems, this paper used both heuristic and metaheuristic methods, respectively, called adaptive nearest neighbour search (ANNS) and modified tabu search (MTS). Pareto optimality of the final production plans and their performance metrics were then validated against enumerative search.

2.4. Single- versus Multiobjective Optimization. Single-objective optimization is aimed at finding the best solution to a problem, under specified constraints. Generally, it consists of three components, i.e., vector of decision variables, constraints, and an objective function. Unlike its single counterpart, multiobjective optimization finds, within feasible regions, the best solutions with respect to more than one objective functions, whose values may be maximized and/or minimized. As described in Section 2.1, a scheduling is an *NP*-Hard problem, whose best solution is not always feasible with a typical algorithm. Much research in this area thus opts for its approximation set instead. This approximation usually involves 2 processes [49, 50], i.e., fitness assignment and population diversification. This is to ensure that the approximated solution is close to the exact one and is uniformly distributed, from one end of the domains to another. Several approaches were taken to assign a fitness, e.g., goal programming, vector evaluation, Goldberg or nondominated sorting, Fonseca and Fleming sorting, and accumulate ranking density strategy (AARS) [51–53].

In the proposed biobjective strategy, for instance, our primary objective function was to minimize all costs incurred by production. To ensure the proper distribution of decision variables, a secondary objective that aims at balancing machine utilization was incorporated.

2.5. Related Works. Scheduling on parallel machines has attracted much interest in the past decades. A range of strategist and objectives have been proposed in the literature. Ruiz-Torres et al. [54], for instance, scheduled different parallel unrelated resources, aiming at reducing the number of late jobs. In that work, both machines with different process times and varying numbers of line staffs were allocated at a given period. They divided the problems into two scenarios, which were parallel machine flexible resource scheduling (PMFRS) and unspecified parallel machine flexible resource one (UMFRS). The problems were solved by a computer program. On scheduling unrelated parallel machines, Kim et al. [55] employed different objectives, taking into account both setup time and total weighted tardiness. They took a heuristic approach with two objectives, which were earliest weighted due date (EWDD) and shortest weighted processing time (SWPT), and two optimizers, namely, two-level batch scheduling (TLBS) and simulated annealing (SA). Edis and Oguz [56] studied unrelated parallel flexible machines and proposed two mathematical models, called PMFRS and UPMFRS, but aiming at minimizing the completion time. Much recently, focus has been moving onto optimization strategies. Polyakovskiy and Hallah [57] studied multistage scheduling by considering earliest weighted late jobs of parallel machines. It was observed that each job required a different process time and was to be delivered at a different due date. Therein, the mixed integer programming (MIP) method, called MASH, was employed to solve the bottleneck problem in multistage scheduling. A just-in-time (JIT) scheduling approach was taken by Kayvanfar et al. [1, 58] to minimize total tardiness and the number of early completed jobs. The overall cost thus depended on whether jobs were completed earlier or later than specified due dates. Similarly, an MIP method was employed. It was assumed that there was no job insertion and the unrelated parallel machines had different processing rates. Nonetheless, their method suffered from the problem size and, without hybrid integration, is suitable for only small ones. With similar machine condition, Zhang et al. [59] minimized weighted average tardiness by means of reinforcement learning (RL). In their experiments, release time and due date were randomly specified, and the resulted scheduling was found to outperform all the methods being benchmarked.

In addition, there have been a number of most recent studies, focusing on parallel machines, JIT approaches, multiobjective strategies, and the combinations of these areas, and hence worth explored here. Majority of early works assumed single objective strategy [60–65]. In 2014, Kayvanfar and Teymourian [60] proposed an intelligent water drop (IWD) algorithm to schedule unrelated parallel machines. It was validated on five machines, with small and large numbers of jobs. However, it did not consider setup time. This work was later extended to account for not only identical [61] but also unidentical [63] machines. The former still followed previous optimization strategy, whereas the latter proposed a parallel net benefit compression–net benefit expansion (PNBC-NBE) algorithm. These methods differed from their precedence in that the former considered JIT manufacturing with controllable process time, while the latter focused on sequence-related setup times. Another similar work was proposed by Lin and Ying [62]. They focused on a new optimization algorithm, called hybrid artificial bee colony (HABC), which was compared against TS, nature inspired, and RSA optimizations. It was validated on bigger problems. Since then, a number of works explored various

metaheuristic optimization strategies [64, 65] on similar problems. Thus far, these methods did not consider cycle time. Biobjective scheduling for batch processing with no setup time was considered in subsequent attempts [66, 67]. Similar to the previous works, they also used metaheuristic optimization. Taking into account sequence-dependent setup time, Yepes-Borrero et al. [68] proposed minimizing both makespan and number of resources, by using greedy algorithm. On solving multiobjective scheduling problems, Kayvanfar et al. [69, 70] used SA and GA as optimizers, for unrelated and identical parallel machines, respectively. Neither setup time nor cycle time was considered in those works, but the latter considered JIT approach. At least one or more limitations are shared by the abovementioned works and different from ours. They include the omissions of sequence-dependent setup time, process time, and cycle time, as well as smaller numbers of objectives.

3. Proposed Method

This paper focuses on scheduling and its analyses on unrelated parallel machines, whose setup times were sequence dependent. Furthermore, it was assumed that assigning a job to any machine is subject to its predefined eligibility and that deliveries were made at production intervals. This total cost minimization problem was solved by using both mathematical model and computational intelligence methods. Later, Pareto strategy [71] for biobjective problem was considered, by integrating balanced machine utilization into the cost functions. Likewise, the scheduling results obtained by the proposed computational intelligence methods were compared against those by the mathematical model baseline.

3.1. Mathematical Model for Scheduling Problems. This section describes a mathematical model referred to as a baseline in benchmarking. With this model, it was assumed that (1) machines were unrelated and parallel, (2) their setup times depended on production sequence, and (3) their eligibility for given jobs was prespecified. The objective of their scheduling was to minimize the makespan by using an integer programming method.

The input variables were scheduling or assignment table (X) and quantity units of processed jobs (N) at each period. Given these variables, completed (C) and quantity inventory units stored at the end of period (H) would be then determined. Finally, the cost function (Z) would be evaluated, given system parameters, from the numbers of assigned, produced, and stored jobs, as expressed in Equation (1). Specifically, the system parameters were (sequence dependent) setup times (S), unit production (G), and storage (O) costs, respectively.

To ensure realistic scheduling, there are some constraints worth considered and described as follows.

In each production round, the quantity units of processed job j must meet delivery demand, while the total production time must fall within specified working hours, as expressed in

$$C_{jt} \leq \text{WorkHour}, \quad \forall_{j,t}. \tag{2}$$

In addition, the quantity units of processed job j produced on those machines in total (all periods, t) must be a positive integer and was no less than the lower lot size. Finally, a job assigned to a machine must be within its capacity. These constraints are given in Equations (3) and (4), respectively.

$$\sum_{m=1}^{MC} N_{jmt} \geq \text{LowerLotSize}, \quad \forall_{j,t}\, j \neq 1, \tag{3}$$

$$N_{jmt} \leq \text{Big}M \cdot Z_{jmt}, \quad \forall_{j,m,t}\, j \neq 1. \tag{4}$$

With this scenario, provided the problem parameters, i.e., jobs, working period and hours, and machines, as well as resources' data, i.e., production and release time, demand, machine eligibility, production and storage unit costs, and setup cost, scheduling gives an optimal binary production table (X_{ijmt}), as well as corresponding quantity units of each processed job on each machine (N_{jm}) total complete time (C_j), at each period. To maintain valid production table, all elements were asserted by auxiliary variables (A_{jmt}, B_{jmt}) as defined in Equations (5) and (6). This ensures, for instance, that a machine must be assigned with at least one job, as given in Equation (7).

$$\sum_{i=1, i\neq j}^{n} X_{ijmt} = 1 - A_{jmt}, \quad \forall_{j,m,t}\, j \neq 1, \tag{5}$$

$$B_{jmt} + A_{jmt} = 1, \quad \forall_{j,m,t}\, j \neq 1, \tag{6}$$

$$\sum_{m=1}^{MC} A_{jmt} = 1, \quad \forall_{j,t}\, j \neq 1. \tag{7}$$

During the integer programming, plausible scheduling was subject to specific constraints, as follows. The quantity units of stored job j at period t, H_{jt}, were equal to the job j previously stored (H_{jt-1}) and currently processed on all machines but subtracted by the amount required. That is,

$$H_{jt} = \sum_{m=1}^{M} N_{jmt} - d_{jt}, \quad \forall_{j,t}\, t = 1, \tag{8}$$

$$H_{jt} = H_{j,t-1} + \sum_{m=1}^{M} N_{jmt} - d_{jt}, \quad \forall_{j,t}\, t \neq 1. \tag{9}$$

Provided that a machine was able to process both jobs i and j and that job j was processed right after job i, then the time that job j (C_{jt}) was completed was equal to the time when a previous job (C_{it}) was completed, plus sequence dependent setup time (S_{ij}) and time used to process the specific amount of that job ($P_{jm} \cdot N_{jmt}$). The resulted complete time (C_{jt}) must be within working hours of that period. Furthermore, when subtracted by setup and process time, it should be later than its release time (R_j). That is, $\forall_{i,j,m,t}$ and $i \neq j, j \neq 1$,

$$C_{jt} - C_{it} + \text{Big}M \cdot \left(1 - \left(X_{ijmt} \cdot E_{im} \cdot E_{jm}\right)\right) \geq P_j N_{jmt} + S_{ij} \cdot \left(X_{ijmt} \cdot E_{im} \cdot E_{jm}\right), \quad (10)$$

$$C_{jt} - P_j \cdot N_{jmt} - \left(S_{ij} \cdot \left(X_{ijmt} \cdot E_{im} \cdot E_{jm}\right)\right) \geq R_j. \quad (11)$$

Regarding a job sequence, only one job j could succeed another job i, and it should be processed by only one machine, and vice versa. In other words, at a given period, a job might not be distributed to different machines. Furthermore, preceding and succeeding jobs must be different, or a job could not be processed if it had just been completed on that machine. These constraints are realized by Equations (12)–(15), respectively.

$$\sum_{i=1}^{n} \sum_{m=1}^{M} X_{ijmt} \cdot E_{im} \cdot E_{jm} = 1, \quad \forall_{j,t}\, j > 2, \quad (12)$$

$$\sum_{j=1}^{n} \sum_{m=1}^{M} X_{ijmt} \cdot E_{im} \cdot E_{jm} = 1, \quad \forall_{i,t}\, i > 2, \quad (13)$$

$$\sum_{i=1}^{n} X_{ijmt} - \sum_{w=1}^{n} X_{jwmt} = 0, \quad \forall_{i,j,m,t}\, i \neq j, j \neq w, \quad (14)$$

$$X_{jjmt} = 0, \quad \forall_{j,m,t}. \quad (15)$$

To ensure utilization constraints, a machine should have at least one first job and one last, maybe different, job, as expressed in Equations (16) and (17), respectively. Likewise, a job should be assigned first and last, each time to at least one, maybe different, machine, as expressed in Equations (18) and (19), respectively. Lastly, any given job must be assigned to only one machine, as expressed in Equation (20).

$$\sum_{j=2}^{n} X_{1jmt} \cdot E_{jm} = 1, \quad \forall_{m,t}, \quad (16)$$

$$\sum_{i=2}^{n} X_{i1mt} \cdot E_{im} = 1, \quad \forall_{m,t}, \quad (17)$$

$$\sum_{j=1}^{n} \sum_{m=1}^{MC} X_{1jmt} \cdot E_{jm} = MC, \quad \forall_{t}\, j \neq 1, \quad (18)$$

$$\sum_{i=1}^{n} \sum_{m=1}^{MC} X_{i1mt} \cdot E_{im} = MC, \quad \forall_{t}\, i \neq 1, \quad (19)$$

$$\sum_{m=1}^{M} \left(X_{ijmt} + X_{jimt}\right) \cdot E_{im} \cdot E_{jm} \leq 1, \quad \forall_{i,j,t}\, i \neq j. \quad (20)$$

To maintain integer computability, values in scheduling tables were only binary, i.e., 1 or 0, depending on whether any assignment of a job after completion of another one was eligible for that machine at that period or otherwise, respectively. Finally, numbers of processed and stored jobs and that of process time per job must be positive integers. Since these constraints are self-explanative, their detailed expressions are hence omitted.

The outputs of scheduling process were, for a given job i and at period t, (1) number of stored job (H_{it}), (2) quantity units of processed jobs at a machine m (N_{imt}), and (3) time spent on processing the job (C_{it}).

3.2. Computational Intelligence Methods. This section describes the proposed heuristic and metaheuristic methods for solving unrelated parallel scheduling problem. Herein, we employed adaptive nearest neighbour search (ANNS) and modified tabu search (MTS). Their processes and formulations in the present context are provided in the following subsections.

3.2.1. Adaptive Nearest Neighbour Search (ANNS). The ANNS starts by specifying an initial solution and a best solution table. Upon entering the updating loop, neighbouring solutions with respect to job sequence (**S**) and processing demand (**D**) were created, adaptively. More specifically, the neighbours were defined by offsetting a current solution by distances of a 2^0–2^2, multiplied by adaptive step size, i.e.,

$$\mathbf{S}(i_s) = \{-4,-2,-1, 0, 1, 2, 4\} \times W_s \times i_s, \quad (21)$$

$$\mathbf{D}(i_d) = \{-4,-2,-1, 0, 1, 2, 4\} \times W_d \times i_d, \quad (22)$$

where i_s and i_d were the indices to job sequence pair and processing demand tables, respectively, whose members were prepopulated by all possible combinations of the respective variables. Specifically, for a problem with four jobs, there were $4! = 24$ possible sequences. The processing demand was computed for each period and product from its actual demand subtracted by that already processed and stored in an inventory. The values were lower bounded by economic order quantities (EOQ) and lower lot size. In addition, **S** and **D** were created neighbours and W_s and W_d were adaptive steps, of the job sequence and the processing demand indices, respectively.

Subsequently, objective functions were evaluated, given each neighbour in turns, and the best solution would be appended to the best solution table. At each iteration, solutions in this table would be sorted by their objective functions. If the size of this table was greater than a predefined value, the worst solutions so far would then be discarded. Later, the remaining ones were assessed, and if there were excessive numbers of duplicated solutions, then the adaptive steps ($W_{s,d}$) were adjusted. The best solution would be chosen in the next iteration. This ANNS process was iterated until convergence or the number of rounds reached a specified limit.

3.2.2. Modified Tabu Search (MTS). Similarly, MTS also started by specifying an initial solution and a best solution table, but it also created a tabu table. Adaptive table indexing of neighbours was evaluated following that of ANNS. Evaluation of objective functions and updating of the best solution table were also the same as before. However, once the best solution table had been updated, the tabu list was modified. If the best solution in the current round was no better than the previous ones, it would be added to the tabu list. This would effectively enable backtracking solutions to existing

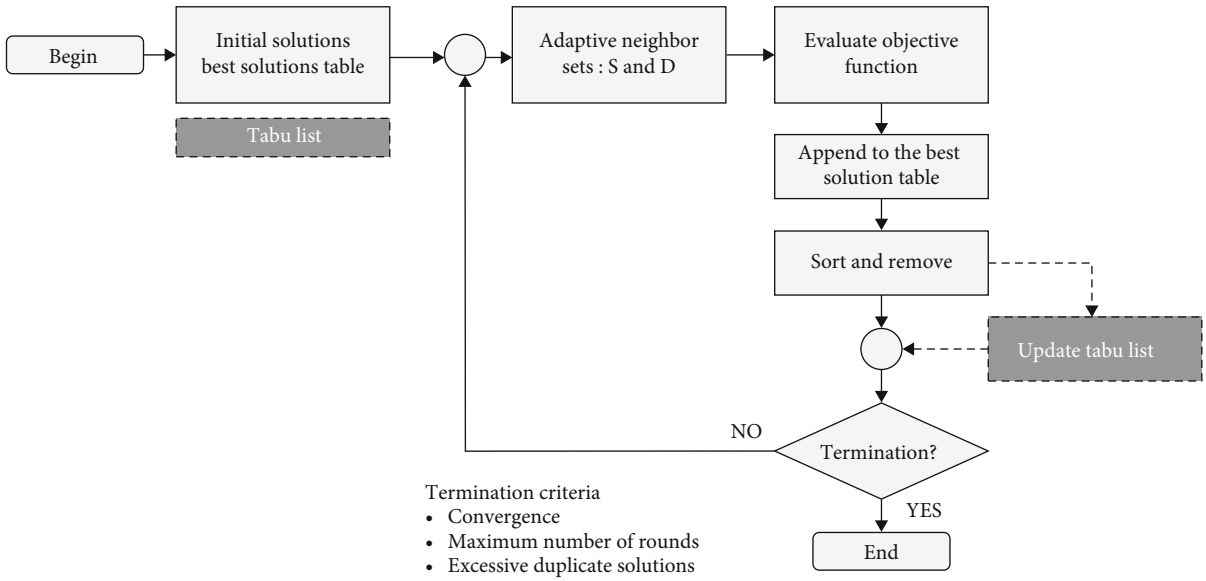


Figure 1: Diagram of ANNS and MTS methods. Light boxes with solid lines and dark boxes with dotted lines indicate common steps and those required only by MTS, respectively.

tabu members, should it be locked in local minima. Likewise, this process was iterated until convergence or the number of rounds reached a specified limit.

The diagram of ANNS and MTS processes is illustrated in Figure 1.

3.3. Pareto-Based Approach toward Biobjective Scheduling. It will be later demonstrated that, with single objective function, although production cost (Z), expressed in Equation (1), was optimized, the machine utilization was not. The machine utilization was defined as a ratio between its operational and total working hours. Therefore, in the present context, another objective was to maximize the minimum makespans (U), normalized over all involving machines, that is,

$$U = \min_{1 \le j \le n} \frac{\sum_{t=1}^{T} C_{jt}}{\sum_{j=1}^{n} \sum_{t=1}^{T} C_{jt}}. \tag{23}$$

To balance between production cost and resource utilization, the main contributions of this paper are to devise a biobjective model of an unrelated parallel machine scheduling problem and then to obtain their Pareto solutions. Consider a 2-objective minimization problem:

$$\min \; F(X)\; F(X) = \{Z(X), U(X)\}. \tag{24}$$

A solution X is said to dominate a solution Y if $Z(X) \le Z(Y)$ and $U(X) \le U(Y)$, and there is at least one of these functions, where X yields strictly lower value than does Y. Solution X is called Pareto optimal if it is not dominated by any other feasible solutions. To this end, three Pareto strategies are proposed.

(a) *Pseudo-Pareto Optimal.* A set of some optimal production plans with respect to the primary objective, i.e., production cost, was collected. Among these plans, the one with optimal utilization was selected.

(b) *Parallel Pareto Optimal.* Both primary and machine utilization objective functions were evaluated, and a set of optimal scheduling with respect to each of these objectives was collected, separately. Upon convergence, the optimal one from both groups was selected.

(c) *Serial Pareto Optimal.* The production objective function was first evaluated, but an optimal one would be admissible in the candidate list, only if its utilization was also within an acceptable range. Again, upon convergence, the optimal one was finally selected.

3.4. Experiments. Herein, three types of scheduling problems, which were reference, and those with small and large sizes, were considered. For each problem, the input parameters were number of jobs (nJob), number of periods (nPeriod), number of machines (nMachine), working hours (WorkHour), and order or lot size (LotSize). The conditional parameters were production and release time, demand, machine eligibility, and the unit costs for production, storage, and sequence-dependent setup. Following recommendations made by Afzalirad and Rezaeian [72], the parameters were specified as listed in Table 2.

For each problem, their overall cost and scheduling times were compared, among integer programming of the mathematical model (3.1), ANNS (3.2.1), and MTS (3.2.2).

To obtain the results reported as follows, the mathematical models were solved by LINGO 11.0 software, while the ANNS and MTS were implemented on MATLAB™. Both programs were run on a personal computer (PC), installed

Table 2: Specification of input and condition parameters. $\mathbf{U}(A, B)$ means the values being drawn from a uniform distribution of a random variable (RV) between A and B.

Parameter	References	Small problems	Large problems
nJob	{6, 9}	{5, 6, 7, 8}	{10, 15, 20, 25, 30}
nPeriod	{3, 6}	{6, 9, 12}	{6, 9}
nMachine	{2}	{2, 3}	{4, 6, 8, 10}
Processing time: $P(J)$	Fixed	**U** (10, 20)	**U** (20, 50)
Release time: $R(J)$	Fixed	**U** (5, 30)	**U** (1, 200)
Setup time: $S(I, J)$	Fixed	**U** (5, 20)	**U** (4, 10)
Demand: $D(J, P)$	Fixed	**U** (10, 40)	**U** (10, 200)
Machine eligibility: $E(J, M)$	Fixed	**U** (0, 1)	**U** (0, 1)
Production cost: $G(J, P)$	Fixed	**U** (10, 50)	**U** (50, 100)
Storage cost: $O(J, P)$	Fixed	**U** (10, 50)	**U** (50, 100)

with Windows 10 64-bit operating system. The PC was equipped with an Intel I5–6200U processor, clocked at 2.30 GHz and 4 GB main memory.

4. Results and Discussion

This section reports experiment results and relevant discussion on three scheduling methods, namely, the mathematical model, ANNS, and MTS. They were evaluated by reference, small, and large problems.

4.1. Reference Problems. As preliminary model assessments, a set of reference problems was first solved using the mathematical model. Specifically, scheduling 6 and 9 jobs (J), on 2 machines (M), and within 3 and 6 periods (P) were assessed. Their objective functions as expressed in Equation (1) and corresponding computing time are listed in Table 3. Note that, with 09J02M06P case, processing steps and time taken were extremely long and consumed excessive resources on our system. Its details were hence not included in the table (case 4∗). The values in objectives column are global optimum found in each case.

It is evident from Table 3 that as the problems got slightly more complex, for instance, from 6 to 9 jobs, the solver steps and hence scheduling time exponentially increased. In fact, with our setting, the integer programming process took about 4 days already to complete. This method is therefore not suitable for larger scheduling problems, and alternatives would be required, especially for actual JIT manufacturing.

Figure 2 depicts an example of scheduling for the 09J02M03P, while Figure 3 provides, at each period, the number of jobs being ordered (demanded) and stored in the inventory and those being processed on each of the two machines. Note from both figures that only eight real jobs appear. This is due to the first job being exploited as a dummy to satisfy the first and last job constraints, as described in Section 3.1.

In the subsequent experiments, ANNS and MTS were benchmarked against the mathematical model on the same problems. However, because both ANNS and MTS involved uniformly random initializations, they were each executed for six runs. The objective functions and scheduling time for all those runs and the respective averages as well as the differences than those obtained by the mathematical model are listed in Table 4.

Table 3: Number of solver steps, objective functions, and time required by mathematical model-based scheduling of reference problems.

Case	Problems	Solver steps	Objectives	Scheduling times (H:M:S)
1	06J02M03P	14,199	16,981	00:00:38
2	06J02M06P	187,560,294	257,251	95:23:59
3	09J02M03P	183,297,581	74,536	101:23:12
4∗	09J02M06P	N/A	N/A	N/A

Evidently, with small problems (06J02M03P), ANNS and MTS took similar processing time to integer programming on the mathematical model. Even when these problems getting much complicated and hence the mathematical model failed to schedule within reasonable amount of time, computing time required by ANNS and MTS remained roughly unchanged, while giving similar objective values to the mathematical model. Furthermore, closer inspection on the deviations of their objective values from those obtained by the mathematical model was performed. The results are plotted in Figure 4. It is observed that ANNS results were much consistent and closer to the baseline in cases of fewer jobs. The opposite is true, however, for bigger problems.

4.2. Small Problems. Experiments on small problems are those that could be completed within 100 hours. Ten problems were created following the prescriptions in Table 2. They were, for example, 05J02M06P, 07J02M06P, and 08J03M09P. For each case, six production cases were generated, resulting in sixty test cases in total. Figure 5 compares the results obtained by three methods. The graph on the left plots the objective values optimized by mathematical model, ANNS, and MTS, respectively. On the right-hand side,

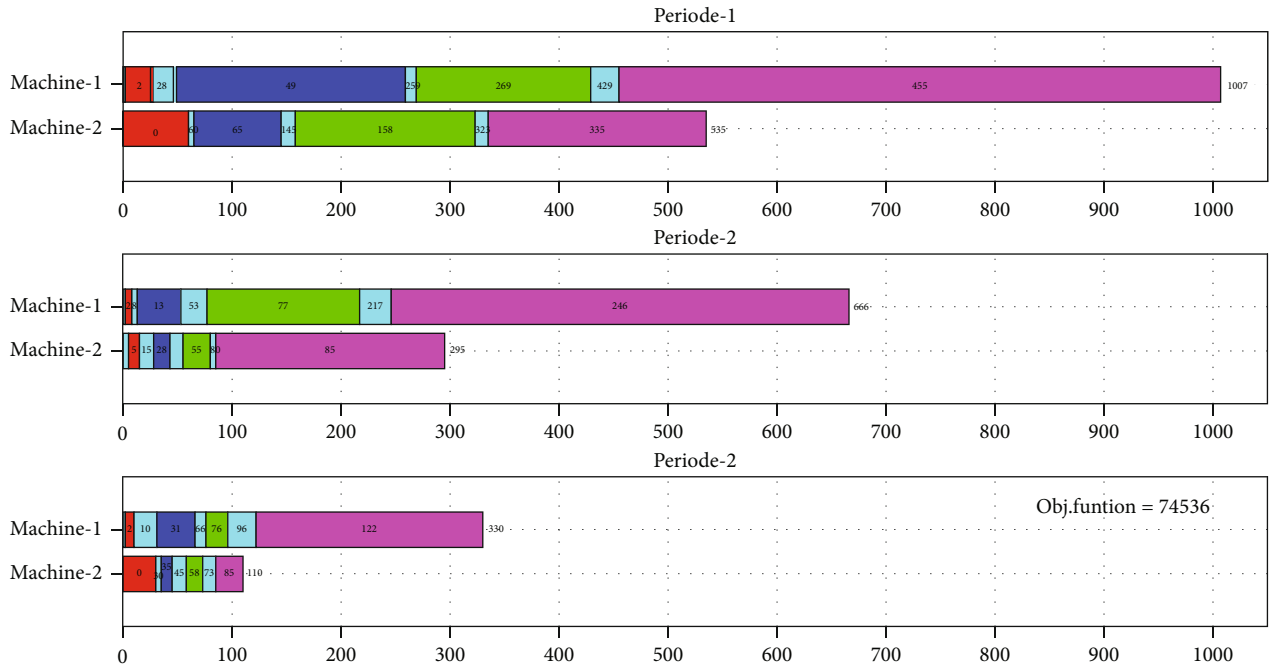


Figure 2: Scheduling plans for the 09J02M03P case. Each bar represents each of the different jobs.

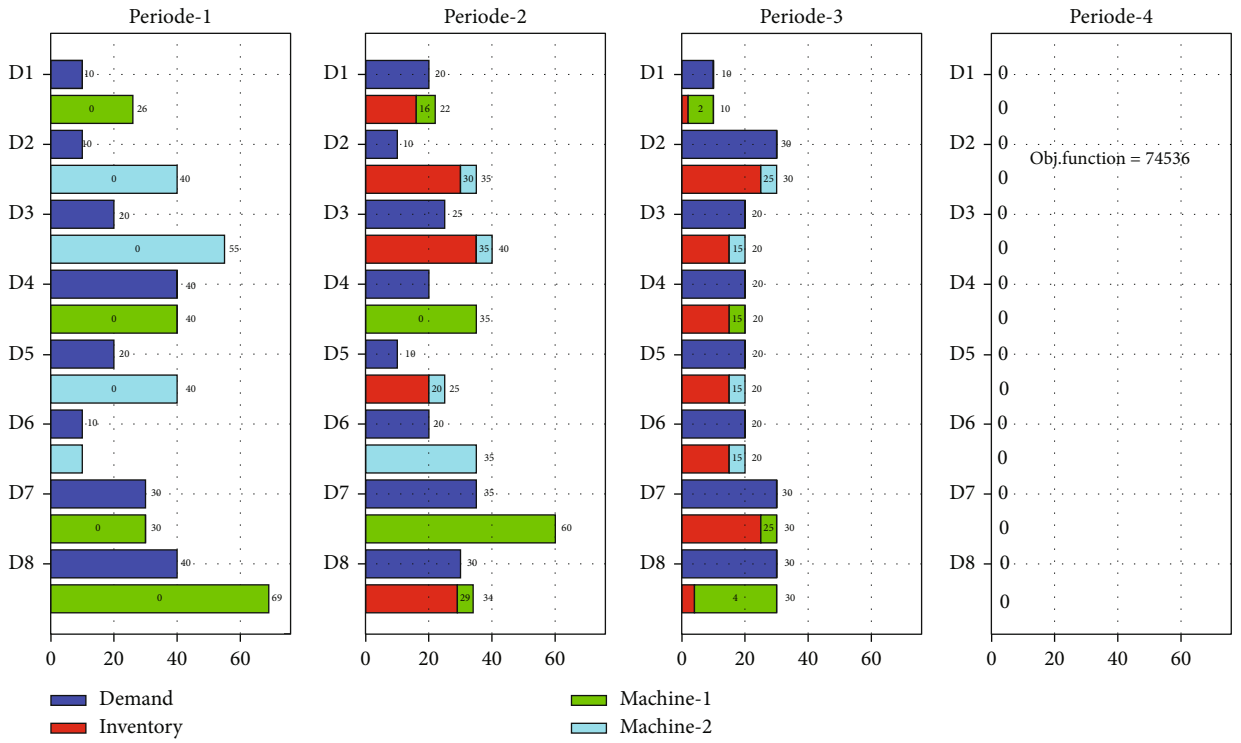


Figure 3: Periodical jobs that were ordered and stored in inventory and those being processed on each of the two machines.

percentage differences, compared to the mathematical model baselines, are shown. It can be concluded from both graphs that, for small problems, both ANNS and MTS methods yielded similar objective values to those by integer programming on mathematical model, with differences bounded by 8 and 10%, respectively.

4.3. Large Problems. Based on the same criterion, large problems were those that required more than 100 hours to schedule. In particular, according to Table 2, they were those with 10–30 jobs, to be scheduled on 4–10 machines, within 6–9 periods. In total, they would consist of 40 problems, but 15 ones, for instance, 10J04M06P, 15J06M09P, 25J08M06P,

Table 4: Comparison of objective value and scheduling time between mathematical model, ANNS, and MTS methods on the reference problems, i.e., 06J02M03P, 06J02M06P, and 09J02M03P.

Case	Run	Objective values			Scheduling time (H:M:S), S		
		Math	ANNS	MTS	Math	ANNS	MTS
06J02M03P	1	16,981	17,770	18,006	00:00:38	33.5	33.5
	2		17,562	17,541		30.5	34.2
	3		18,634	17,918		29.5	32.3
	4		17,074	17,918		34.5	31.9
	5		17,234	17,986		33.5	32.0
	6		17,394	18,006		36.5	32.0
	AVG		17,611	17,896		33.0	32.7
	Δ		630 (3.71%)	915 (5.39%)			
06J02M06P	1	257,251	262,467	282,864	95:23:59	38.5	36.7
	2		263,675	271,934		37.4	36.5
	3		269,307	259,421		38.1	37.2
	4		263,458	258,110		37.2	37.2
	5		265,133	282,864		39.2	36.5
	6		265,858	264,076		37.3	36.7
	AVG		264,983	269,878		38.0	36.8
	Δ		7732 (3.01%)	12,627 (4.91%)			
09J02M03P	1	74,536	75,834	76,878	101:23:12	35.1	33.5
	2		78,945	76,696		34.0	132.0
	3		75,912	74,952		34.0	41.0
	4		75,289	80,352		34.0	35.9
	5		79,981	77,408		41.0	32.2
	6		79,531	75,395		41.0	32.1
	AVG		77,582	76,947		36.5	51.1
	Δ		3046 (4.09%)	2411 (3.23%)			

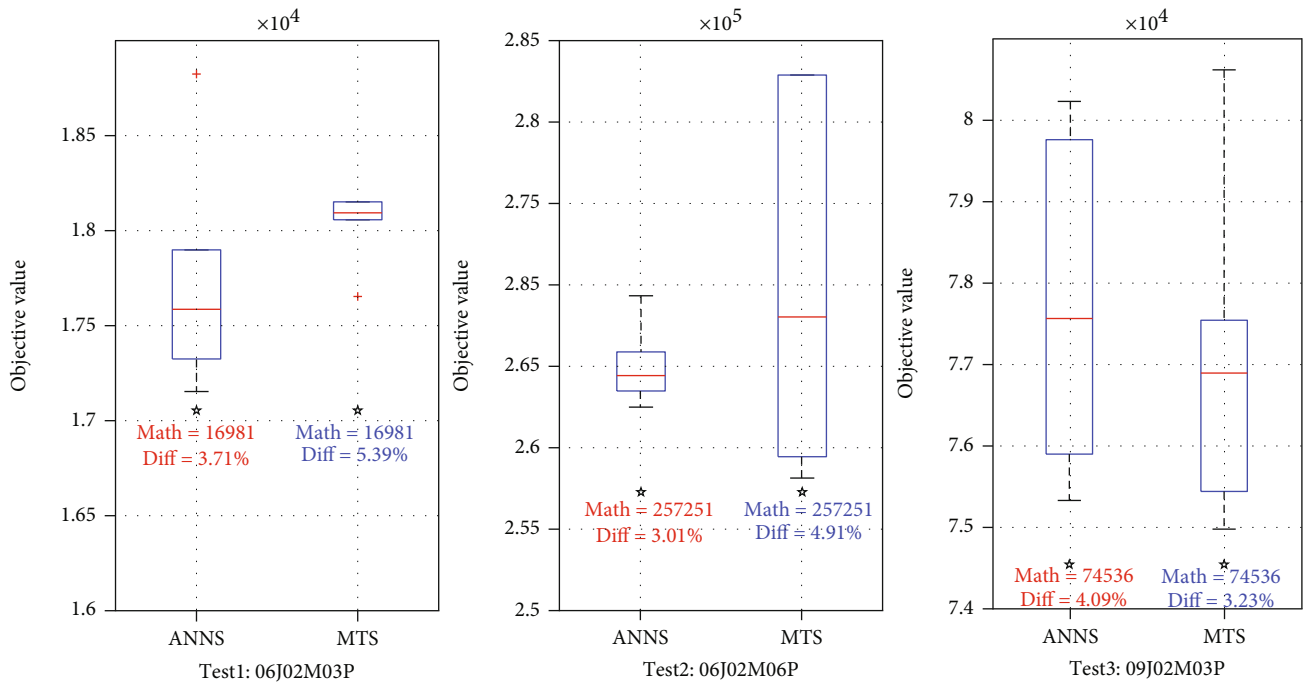


Figure 4: The solutions by ANNS and MTS algorithms compared to those by the mathematical models.

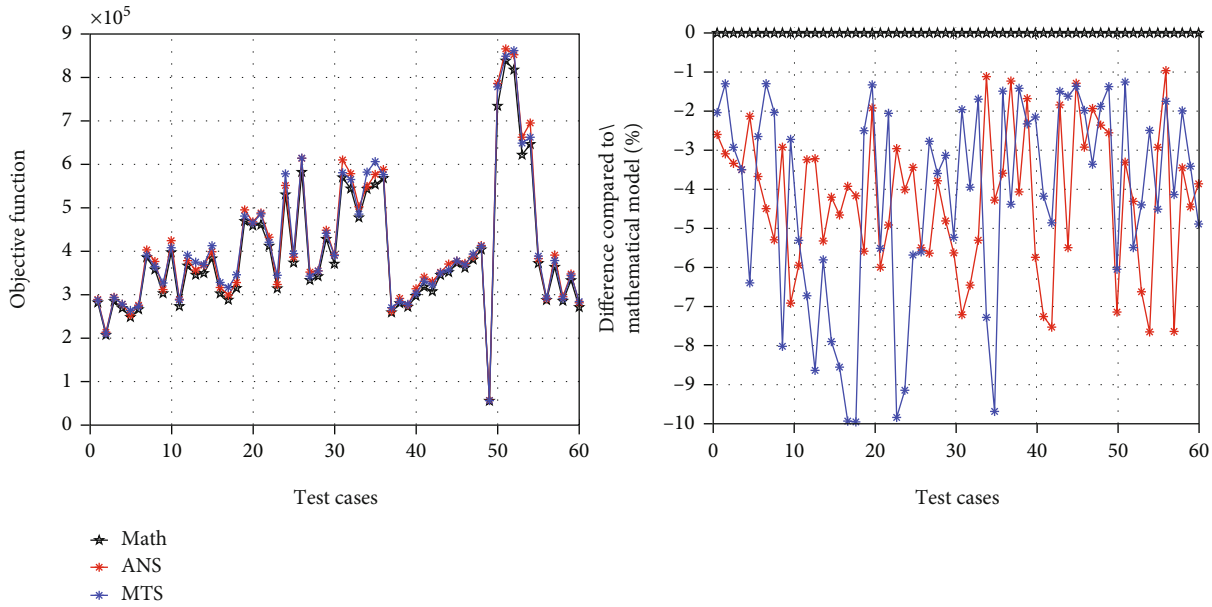


Figure 5: The (a) objective functions and (b) corresponding percentage differences to those obtained by the mathematical models, for 60 test cases of small problems.

Table 5: Problems and their respective ID considered in the large problem experiments.

ID	Problem	ID	Problem	ID	Problem
1	10J04M06P	6	15J06M09P	11	25J08M06P
2	10J04M09P	7	20J04M06P	12	25J10M06P
3	10J06M09P	8	20J04M09P	13	30J06M06P
4	15J04M06P	9	20J06M09P	14	30J08M06P
5	15J04M09P	10	25J06M06P	15	30J10M06P

and 30J10M06P, were empirically chosen, as listed in Table 5.

Due to the sizes of these problems being prohibitive for mathematical model, it was thus discarded from the experiments. Similarly, for each problem, four different parameters were randomly specified as per Table 2, resulting in sixty cases in total. Unlike Section 4.2, averaged objective functions and scheduling time over each problem are plotted in Figure 6.

The above results indicate that, with a single objective, consisting of periodical production, inventory storage, and sequence-dependent setup cost, ANNS and MTS required significantly less processing time than the mathematical model method. Particularly for large problems, they took just about 60 seconds on average, instead of 100 hours. Meanwhile, ANNS and MTS methods could schedule these problems with 3–5% and 10% difference objective values than the baseline one, for reference and small problems, respectively. The objective was the highest when scheduling many jobs on few machines, while running time was so when doing on many periods. Although their results were almost identical, closer inspection revealed that ANNS slightly outperformed MTS in terms of objectives and processing time.

4.4. Pareto Optimal Solutions to Biobjective Scheduling Problems. It is notable from Figure 2 that at global optimum, the utilization of the first machine was greater than that of the other one. Moreover, their utilization in the first period was much greater than that in the third. To resolve the balance between production cost and resource utilization issue, this section describes and compares three Pareto-based approaches, proposed in Section 3.3. The evaluations reported as follows were performed on 12 problems, i.e., 06J02M03P and 06J02M06P from reference problems, eight uniformly generated cases per 05H02M06P and 06J02M09P from small problems, and 07J03M12P and 08J03M09P with a case each, also from small problems. Due to its superior performances, ANNS was chosen as an optimizer.

4.4.1. Pseudo-Pareto Optimal. With the pseudo-Pareto strategy, a set (Z-List) of 10 production plans, whose Z values were minimized, was collected. Subsequently, their utilization metrics (U) were then evaluated and labelled as U-List . The statistical Z-score was then computed separately from means and standard deviation of the respective list. The optimal plan was the one that gave the best-averaged Z -score between Z and U lists. Scheduling based on optimal primary and secondary objectives as well as on their pseudo-Pareto optimal of two sample cases, i.e., 07J03M12P and 08J03M09P, is displayed in Figures 7(a) and 7(b).

4.4.2. Parallel Pareto Optimal. With this strategy, both Z and U were evaluated in parallel, during which 2 sets of 10 best production plans were separately compiled for both Z-List and U-List. Once converged, the optimal one that gave the best-averaged Z-score between Z and U lists was selected. The resultant scheduling on the same problem sets and parameters is plotted in Figures 7(c) and 7(d).

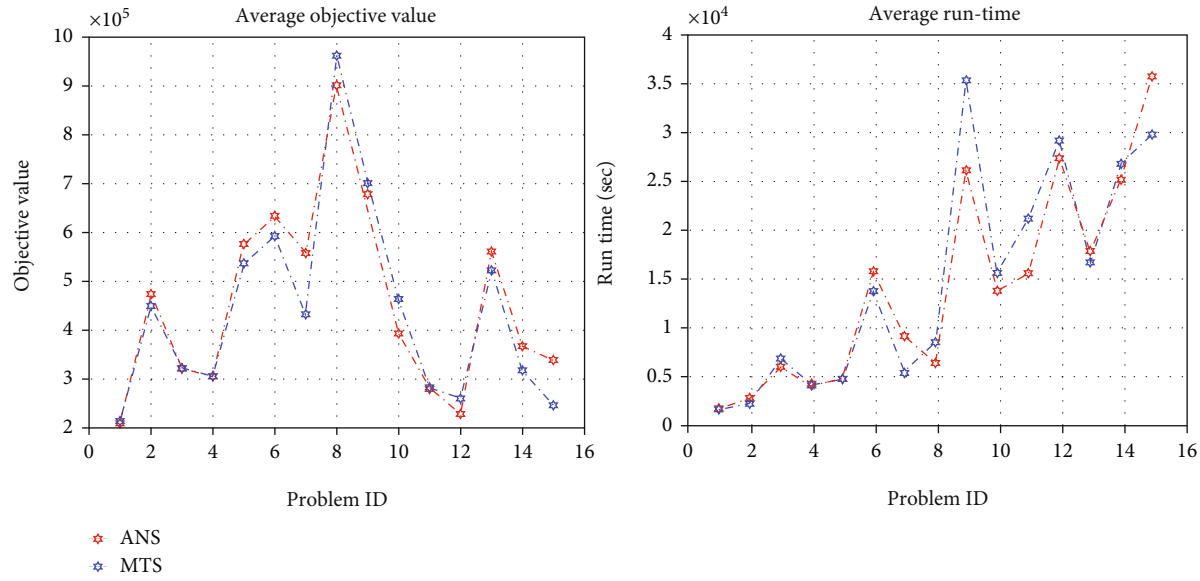


Figure 6: The (a) average objective functions and (b) processing time for 15 selected large problems, scheduled by ANNS and MTS methods.

4.4.3. Serial Pareto Optimal. This strategy was similar to the above strategies, except that after the primary objective was evaluated for each plan, it would then be evaluated by the secondary one. The plan with the best primary scores would only be stored in a candidate list, only if its secondary score was higher than an acceptable threshold. The resulted scheduling on the same problem sets and parameters is displayed in Figures 7(e) and 7(f).

In terms of efficiency for all twelve cases, the pseudo-Pareto strategy took the least computing time of 403 seconds on average (ranging between 79 and 1021 sec.), followed by serial and parallel of 536 seconds (ranging between 92 and 1820 sec) and 887 seconds (ranging between 164 and 2384 sec), respectively.

To assess the performance of the proposed strategies, Box-Whisker plots of resultant production costs in all twelve cases, based on production (left), utilization (middle), and biobjective (right) metrics, are illustrated in Figure 8. It is clearly seen that, when aiming at minimizing production cost (Z), the primary objective was generally low, while maximizing machine utilization (U) slightly raised it, but particularly in parallel cases, to greater extent (e.g., at almost 100%). On the other hand, scheduling with biobjective (Z and U) model resulted in balanced cost versus utilization, while maintaining relatively low overall costs. Nonetheless, marginal decrease in this balance is noticeable with the parallel Pareto strategy.

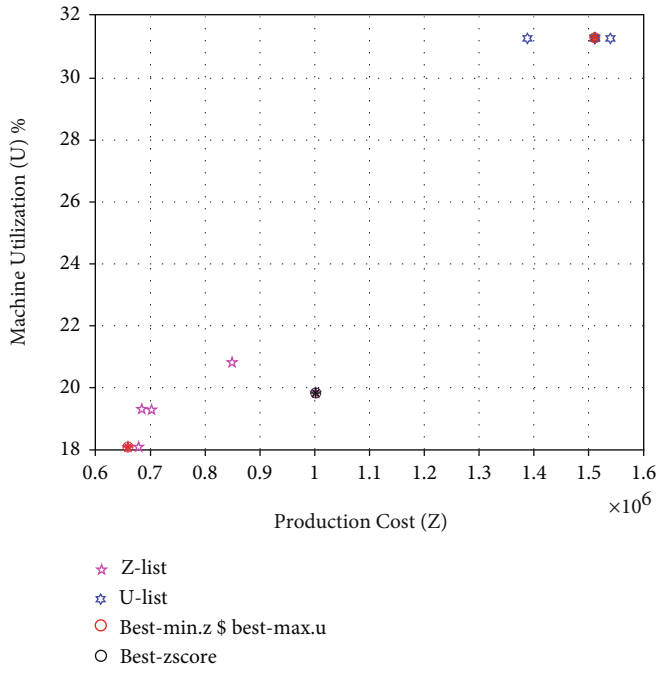
Similar analyses can be made regarding machine utilization. In Figure 9, Box-Whisker plots of resultant machine utilization in the same cases are plotted. It is clearly seen that, with utilization being optimized, their values tended to be a little higher than when focusing primarily on production cost. However, the biobjective model well balanced them, especially when taking pseudo-Pareto-based approach.

Further analyses on individual cases are reported in Figure 10. It revealed that, for small problems, both metrics remained similar regardless of an approach taken, except, however, as problems got larger, when the pseudo- and parallel Pareto-based approaches resulted in lower production cost and higher utilization, respectively. The graphs also suggest that the serial Pareto-based approach was the best compromise on these metrics.

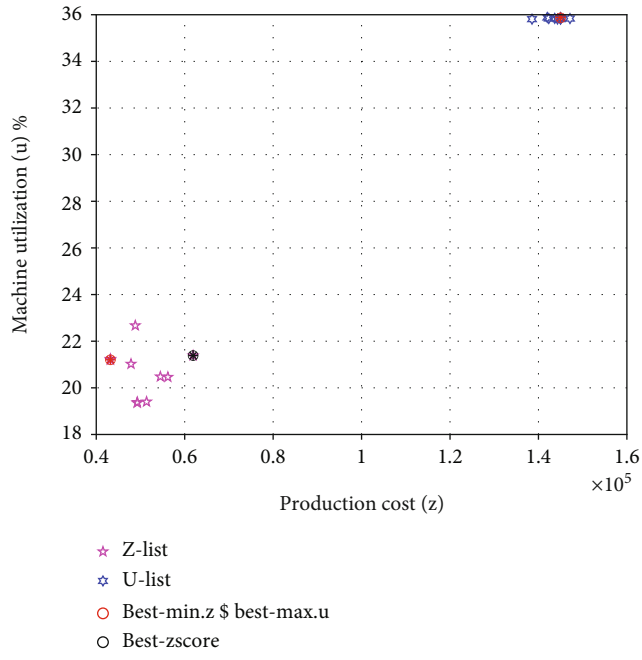
4.5. Assessments on the Number of Problem Instances. When formulating the reference problems, we devised exhaustive combinations of scheduling {6, 9} jobs on {2, 3, 6} machines within {3, 6} periods, resulting in 12 cases. But only 3 instances were executed. In addition, for small problems, out of 24 combinations of {5, 6, 7, 8} jobs on {6, 9, 12} machines within {2, 3} periods, 10 cases were selected. For each case, 6 production parameters (i.e., P, R, S, D, E, G, and O) were randomly generated as per Table 2, resulting in 60 instances being analyzed. Likewise, for large problems, a total of 15 cases, each with 6 production parameters, resulted in 90 instances being considered. Finally, on evaluating Pareto optimization, a total of 12 instances were drawn from references and small problems. Accordingly, there were in total 153 different scheduling instances involved in the above experiments.

Nonetheless, to elucidate that the number of instances was sufficient to draw conclusions, an experiment on additional cases, i.e., 06J04M10P, 06J04M15P, 09J04M15P, and 09J06M10P, each with 4 parameter settings, was performed by using ANN and MTS algorithms, in turn. Their resultant objectives were computed and then averaged over a given problem. The Box-Whisker plots comparing both algorithms and four cases are illustrated in Figure 11.

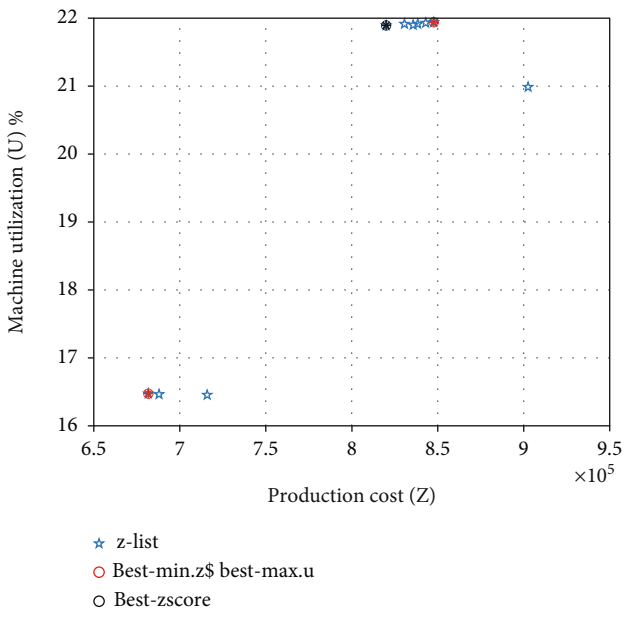
It is evident that regardless of manufacturing parameters and algorithms, each problem case exhibited similar objectives, i.e., as small as about 0.6–3.5% percent deviations. This implies that its performance is dependent only on the problem sizes but not variations over instances. The problem



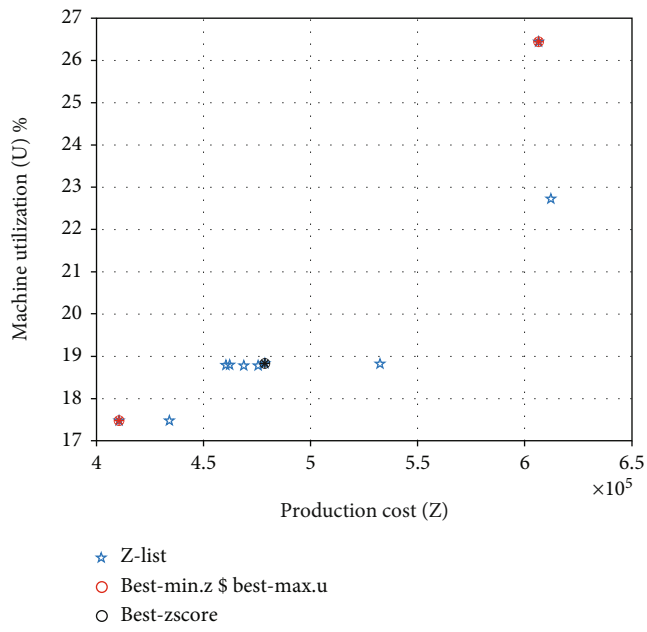
(a) Pseudo-biobjective 07J03M12P



(b) Pseudo-biobjective 08J03M09P

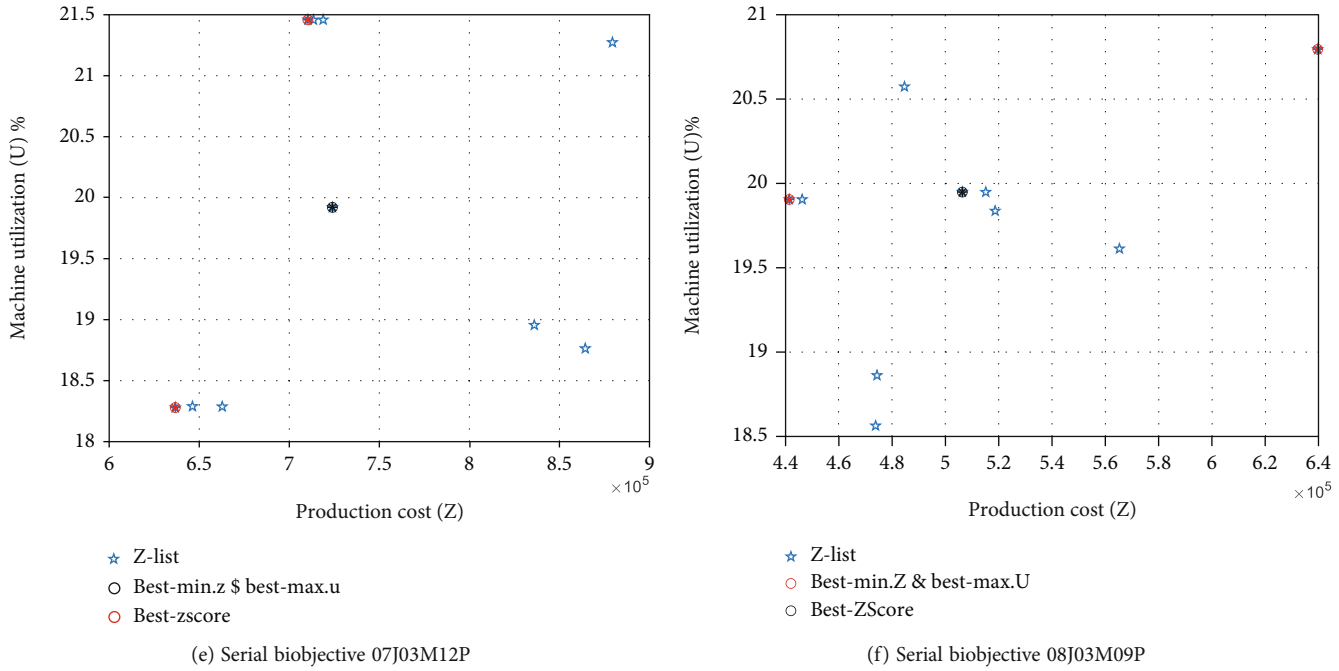


(c) Parallel biobjective 07J03M12P



(d) Parallel biobjective 08J03M09P

Figure 7: Continued.



(e) Serial biobjective 07J03M12P (f) Serial biobjective 08J03M09P

FIGURE 7: The biobjective optimal scheduling plans for the problems (a, c, e) 07J03M12P and (b, d, f) 08J03M09P, by using (a, b) pseudo, (c, d) parallel, and (e, f) serial objective strategies.

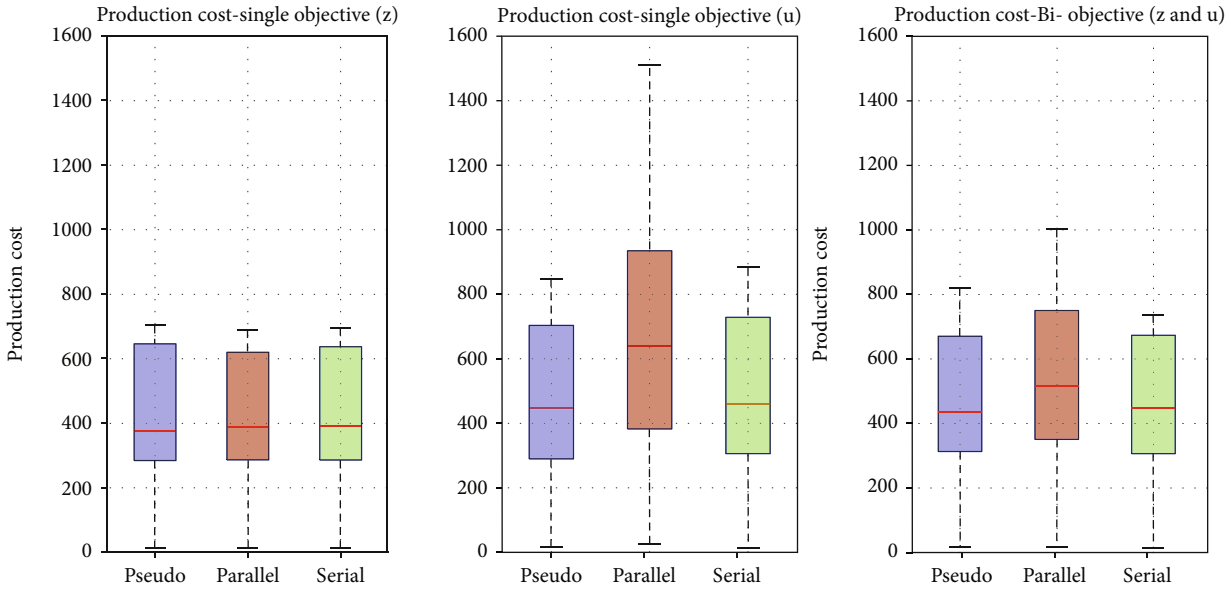


FIGURE 8: Performance assessments of the proposed strategies, in terms of production costs, in all twelve cases, based on (a) production, (b) utilization, and (c) biobjective metrics.

classifications and generated instances reported earlier are hence justified.

5. Conclusions

In unrelated parallel machine (UPM) scheduling, the involved resources are prescribed with varying production conditions and constraints, e.g., processing and job sequence-dependent setup time and their eligibility. Theoretically, the optimal solution to such problems may be solved by integer programming, given a mathematical objective model, but with *NP*-Hard complexity. This approach is thus not practical in actual JIT manufacturing environments, except for preliminary assessment on very small problems. To remedy complexity issue, this paper proposed heuristic and metaheuristic approaches, called ANNS and MTS, to this scheduling problem. It was demonstrated in the experiments that both ANNS and MTS yielded slightly less optimal production cost than the integer programming of mathematical model by merely 3–10% on average, in

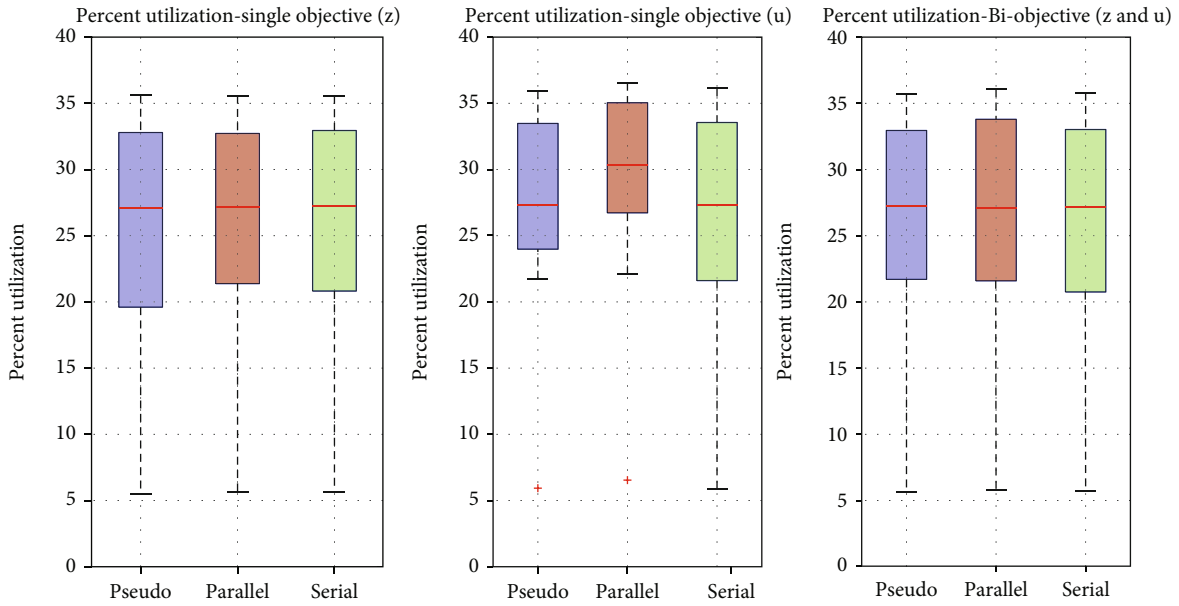


Figure 9: Performance assessments of the proposed strategies, in terms of percent machine utilization, in all twelve cases, based on (a) production, (b) utilization, and (c) biobjective metrics.

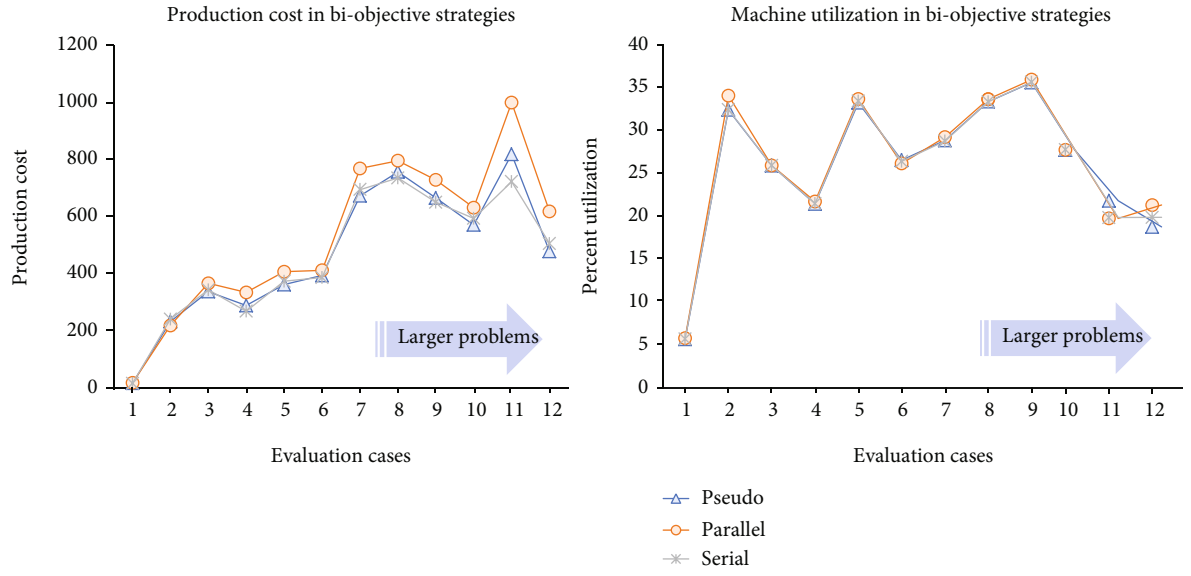


Figure 10: Production cost and machine utilization of involved cases obtained by biobjective strategies.

reference and small problems. Nonetheless, their computing performance was significantly better. They could complete all the problems well under the one-minute mark, whereas their counterpart would take days or, for larger problems, unable to compute at all.

Moreover, we also demonstrated that primarily optimizing production cost that consisted of process time and setup time, and inventory cost, could inevitably result in unbalanced resource utilization. To resolve this issue, we formulated biobjective UPM scheduling model in finite integer domains. These objectives involved minimizing production cost and machine utilization under various prescribed resource and production conditions. The final decision was the policy, assigning a sequence of jobs to UPMs and associate production quantity for a given period. To this end, we proposed the Pareto-based approaches, i.e., pseudo, parallel, and serial ones, where both production cost and utilization metrics were considered. Among these variations, their differences were undiscernible for small problems. However, as problems got larger and more complex, the pseudo-Pareto-based approach appeared to be the best compromise between both metrics. In terms of complexity, parallel Pareto strategy needed the greatest time to compute, followed by serial and pseudo variants, respectively. Their final solutions were, however, indiscernible in the studied problems.

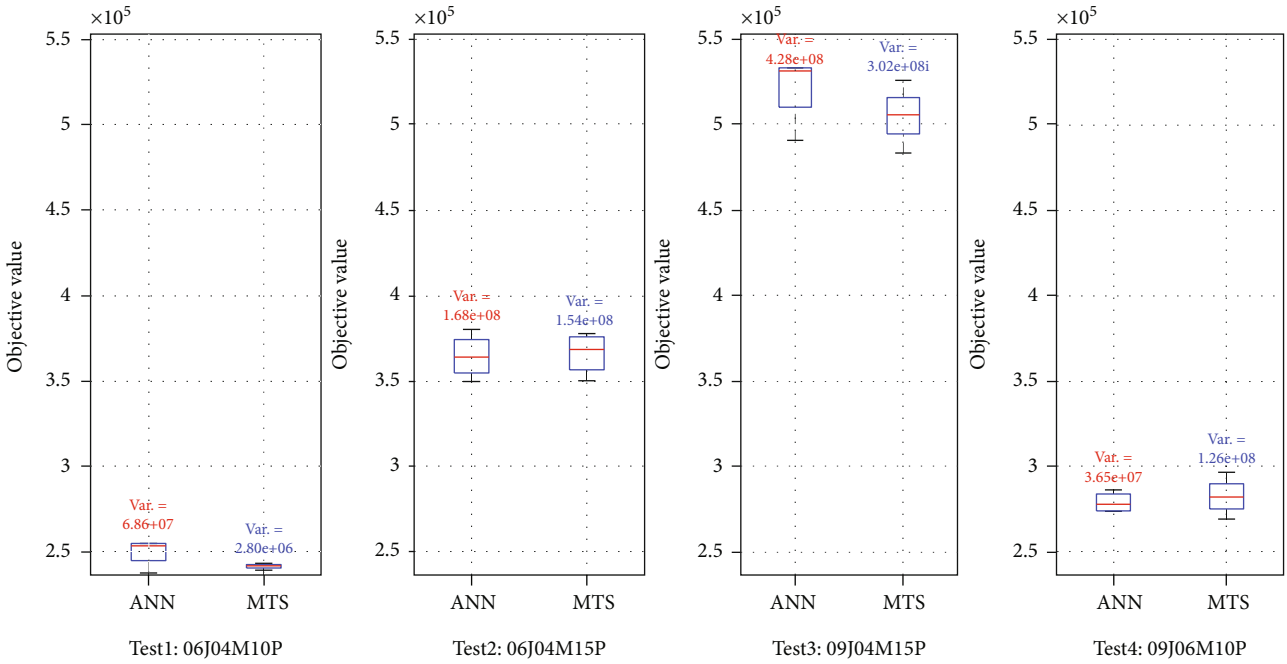


Figure 11: Box-Whisker plots comparing the objectives that resulted from ANN and MTS in four different cases.

In perspective, the developed metaheuristic schemes could be used to solve large problems. It was demonstrated in the experiments that optimizing small problems by ANNS and MTS was highly efficient. They gave solutions with as small deviations from the optimal ones as 3.69% and 4.51%, respectively, on average. As the problems became larger, both algorithms gave inferior solutions but by no greater than 5%, compared to those solved by a much time-consuming mathematical model. Specifically, depending on initial condition, the computation times were brought down from hundreds of hours to a matter of minutes, when both optimizers converged to similar outcomes. Furthermore, the proposed Pareto approaches to biobjective models allowed simultaneously minimizing both production cost and machine utilization. In practice, however, it may be preferable to maximize utilization, while keeping the cost low. To this end, adjustment to the proposed framework is trivial. However, further constraint on the utilizations being lower than say 80% for all periods is suggested, to prevent overload.

Unlike a previous work [73], which posed the problems on a single machine as a traveling salesman one (TSP), our work tackled them on multiple machines as vehicle routing problem (VRP). Compared with [74], where no setup time and stochastic process time were assumed, our model relieved these constraints and considered deterministic process time with SDS. As such, our model could be extended to precedence job shop scheduling, where conditions on previous and next jobs are specified. The developed model is applicable on both single and parallel machine environments. In the case of single machine, the setup time for a prohibited chain, e.g., from job j to k, on a machine, m, could be set to an extremely high cost so that it will be discarded during the optimization. However, this technique may fail, if the optimizer inserts a job in between, i.e., j, m, then k, in which case multilevel SDS may be needed. For parallel machine, since our method computes completion time for each job, order of a chain can be directed by setting its cost, based on the number of completed items for each job. For example, in a case where job k must succeed j, unless the number of items processed by job k is less than that by job j, then its cost will be set to some high value, to avoid being selected.

Other future directions worth focused on include investigation on soft-computing schemes such as convolutional neural network (CNN) and validation on more realistic conditions, e.g., taking into account unscheduled maintenance and ad hoc job insertion.

Data Availability

The scheduling problem in Microsoft Excel (.xlsx), results obtained from the mathematical model in LINGO (.lgr), and simulation results in MATLAB data (.mat) formats used to support the findings of this study are available from the corresponding author upon request.

Conflicts of Interest

The authors declare that there is no conflict of interest regarding the publication of this paper.

Acknowledgments

The authors would like to thank the Institute of Research and Development (IRD), Suranaree University of Technology for funding, and faculty members and supporting staffs from the Institute of Engineering and Centre for Scientific

and Technological Equipment (CSTE) for their contribution, critiques, advice, and supports in preparing and conducting this study.

References

[1] V. Kayvanfar, M. Zandieh, and E. Teymourian, "An intelligent water drop algorithm to identical parallel machine scheduling with controllable processing times: a just-in-time approach," *Computational and Applied Mathematics*, vol. 36, no. 1, pp. 159–184, 2017.

[2] M. M. Ahmadian, A. Salehipour, and T. C. E. Cheng, "A meta-heuristic to solve the just-in-time job-shop scheduling problem," *European Journal of Operational Research*, vol. 288, no. 1, pp. 14–29, 2021.

[3] G. I. Adamopoulos and C. P. Pappis, "Scheduling under a common due-data on parallel unrelated machines," *European Journal of Operational Research*, vol. 105, no. 3, pp. 494–501, 1998.

[4] K. C. Ying and S. W. Li, "Unrelated parallel machine scheduling with sequence- and machine-dependent setup times and due date constraints," *International Journal of Innovative Computing, Information and Control*, vol. 8, no. 1, pp. 1–19, 2012.

[5] J. R. Zeidi and S. MohammadHosseini, "Scheduling unrelated parallel machines with sequence-dependent setup times," *International Journal of Advanced Manufacturing Technology*, vol. 81, no. 9-12, pp. 1487–1496, 2015.

[6] V. Suresh and D. Chaudhuri, "Bicriteria scheduling problem for unrelated parallel machines," *Computers and Industrial Engineering*, vol. 30, no. 1, pp. 77–82, 1996.

[7] S. Zdrzalka, "Preemptive scheduling with release dates, delivery times and sequence independent setup times," *European Journal of Operational Research*, vol. 76, no. 1, pp. 60–71, 1994.

[8] S. Gawiejnowicz, "Scheduling deteriorating jobs subject to job or machine availability constraints," *European Journal of Operational Research*, vol. 180, no. 1, pp. 472–478, 2007.

[9] Y. H. Lee and M. Pinedo, "Scheduling jobs on parallel machines with sequence-dependent setup times," *European Journal of Operational Research*, vol. 100, no. 3, pp. 464–474, 1997.

[10] S. Wang, M. Kurz, S. J. Mason, and E. Rashidi, "Two-stage hybrid flow shop batching and lot streaming with variable sublots and sequence-dependent setups," *International Journal of Production Research*, vol. 57, no. 22, pp. 6893–6907, 2019.

[11] C. C. Han, K. G. Shin, and J. Wu, "A fault-tolerant scheduling algorithm for real-time periodic tasks with possible software faults," *IEEE Transactions on Computers*, vol. 52, no. 3, pp. 362–372, 2003.

[12] P. Wu and B. S. Manjunath, "Adaptive nearest neighbour search for relevance feedback in large image databases," in *Proceedings of the ninth ACM international conference on Multimedia*, pp. 89–97, Ottawa, Canada, October 2001.

[13] R. Alonso, J. A. Bloom, H. Li, and C. Basu, "An adaptive nearest neighbour search for a parts acquisition ePortal," in *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 693–698, Washington, D.C, August 2003.

[14] Z. C. Zhu, K. M. Ng, and H. L. Ong, "A modified tabu search algorithm for cost-based job shop problem," *Journal of the Operational Research Society*, vol. 61, no. 4, pp. 611–619, 2010.

[15] A. Kharrousheh, S. Abdullah, and M. Z. A. Nazri, "A modified tabu search approach for the clustering problem," *Journal of Applied Sciences*, vol. 11, no. 9, 2011.

[16] M. Fera, R. Macchiaroli, F. Fruggiero, and A. Lambiase, "A modified tabu search algorithm for the single-machine scheduling problem using additive manufacturing technology," *International Journal of Industrial Engineering Computations*, vol. 11, no. 3, 2020.

[17] L. Cheng Hao, W. Yan Hong, Q. J. Wei, D. W. Zhao, and S. Rui, "Product service scheduling problem with service matching based on tabu search method," *Journal of Advanced Transportation*, vol. 2020, Article ID 5748680, 9 pages, 2020.

[18] J. Blazewicz, K. H. Ecker, E. Pesch, G. Schmidt, and J. Weglarz, *Scheduling Computers and Manufacturing Processes*, Springer, Berlin, 1996.

[19] R. Slowinski, "Production scheduling on parallel machines subject to staircase demands," *Engineering Costs and Production Economics*, vol. 14, no. 1, pp. 11–17, 1988.

[20] G. Centeno and R. L. Armacost, "Parallel machine scheduling with release time and machine eligibility restrictions," *Computers & Industrial Engineering*, vol. 33, no. 1-2, pp. 273–276, 1997.

[21] K. R. Baker, *Introduction to Sequencing and Scheduling*, Wiley and Sons, New York, 1974.

[22] A. Chassein and A. Kinscherff, "Complexity of strict robust integer minimum cost flow problems: an overview and further results," *Computers and Operations Research*, vol. 104, pp. 228–238, 2019.

[23] A. A. Anwar and D. Clarke, "Irrigation scheduling using mixed-integer linear programming," *Journal of Irrigation and Drainage Engineering*, vol. 127, no. 2, pp. 63–69, 2001.

[24] J. Kim and K. Kim, "Dynamic programming for scalable just-in-time economic dispatch with non- convex constraints and anytime participation," *International Journal of Electrical Power & Energy Systems*, vol. 123, no. 1, article 106217, 2020.

[25] M. Bihari and P. V. Kane, "Evaluation and improvement of makespan time of flexible job shop problem using various dispatching rules-a case study," in *Advances in Mechanical Engineering*, Springer, Singapore, 2020.

[26] G. A. Süer, F. Pico, and A. Santiago, "Identical machine scheduling to minimize the number of tardy jobs when lot- splitting is allowed," *Computers and Industrial Engineering*, vol. 33, no. 1-2, pp. 277–280, 1997.

[27] J. C. Ho and Y. L. Chang, "Minimizing the number of tardy jobs for *m* parallel machines," *European Journal of Operational Research*, vol. 84, no. 2, pp. 343–355, 1995.

[28] D. Bai, H. Xue, L. Wang, C. C. Wu, W. C. Lin, and D. H. Abdulkadir, "Effective algorithms for single-machine learning-effect scheduling to minimize completion-time-based criteria with release dates," *Expert Systems with Applications*, vol. 156, no. 1, article 113445, 2020.

[29] D. W. Kim, K. H. Kim, W. Jang, and F. Frank Chen, "Unrelated parallel machine scheduling with setup times using simulated annealing," *Robotics and Computer-Integrated Manufacturing*, vol. 18, no. 3-4, pp. 223–231, 2002.

[30] M. Woolway and T. Majozi, "A novel metaheuristic framework for the scheduling of multipurpose batch plants," *Chemical Engineering Science*, vol. 192, no. 1, pp. 678–687, 2018.

[31] Z. Tong, L. Ning, and S. Debao, "Genetic algorithm for vehicle routing problem with time window with uncertain vehicle number," in *Fifth World Congress on Intelligent Control and*

Automation (IEEE Cat. No. 04EX788), pp. 2846–2849, Hangzhou, China, 2004.

[32] S. Prabu, K. Karthik, J. V. Rayudu, and D. G. Krishnan, "Optimization of machining parameters in wire EDM of EN24 steel," *International Journal of Mechanical and Production Engineering Research and Development*, vol. 7, no. 6, pp. 359–364, 2017.

[33] B. Zheng, Y. F. Li, and H. Z. Huang, "Aeroengine fault diagnosis method based on optimized supervised Kohonen network," *Journal of Donghua University*, vol. 32, no. 16, pp. 1029–1033, 2015.

[34] P. Horkaew and G. Z. Yang, "Construction of 3D dynamic statistical deformable models for complex topological shapes," in *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2004*, Springer, Berlin, Heidelberg, 2004.

[35] P. Horkaew and S. Puttinaovarat, "Entropy-based fusion of water indices and DSM derivatives for automatic water surfaces extraction and flood monitoring," *ISPRS International Journal of Geo-Information*, vol. 6, no. 10, p. 301, 2017.

[36] S. Bekiros, J. A. Hernandez, S. Hammoudeh, and D. K. Nguyen, "Multivariate dependence risk and portfolio optimization: an application to mining stock portfolios," *Resources Policy*, vol. 46, no. P2, pp. 1–11, 2015.

[37] R. M. Karp, *Reducibility among Combinatorial Problems: Complexity of Computer Computations*, Plenum Press, New York, 1972.

[38] P. He, "Optimization and simulation of remanufacturing production scheduling under uncertainties," *International Journal of Simulation Modelling*, vol. 17, no. 4, pp. 734–743, 2018.

[39] S. Rajakumar, V. P. Arunachalam, and V. Selladurai, "Workflow balancing in parallel machine scheduling with precedence constraints using genetic algorithm," *Journal of Manufacturing Technology Management*, vol. 17, no. 2, pp. 239–254, 2006.

[40] S. Tian, T. Wang, L. Zhang, and X. Wu, "Real-time shop floor scheduling method based on virtual queue adaptive control: algorithm and experimental results," *Measurement (Lond)*, vol. 147, no. 1, 2019.

[41] D. M. Ryan, C. Hjorring, and F. Glover, "Extensions of the petal method for vehicle routeing," *The Journal of the Operational Research Society*, vol. 44, no. 3, pp. 289–296, 1993.

[42] J. Renaud and F. F. Boctor, "A sweep-based algorithm for the fleet size and mix vehicle routing problem," *European Journal of Operational Research*, vol. 140, no. 3, pp. 618–628, 2002.

[43] C. Prins, P. Lacomme, and C. Prodhon, "Order-first split-second methods for vehicle routing problems: a review," *Transportation Research Part C*, vol. 40, no. 1, pp. 179–200, 2014.

[44] P. Hansen and N. Mladenović, "Variable neighborhood search for the _p_ -median," *Location Science*, vol. 5, no. 4, pp. 207–226, 1997.

[45] P. Hansen and N. Mladenović, "Variable neighborhood search," in *Search Methodologies*, E. K. Burke and G. Kendall, Eds., Springer, Boston, MA, 2005.

[46] L. Shen, M. F. Tasgetiren, H. Öztop, L. Kandiller, and L. Gao, "A general variable neighborhood search for the noIdle flowshop scheduling problem with makespan criterion," in *2019 IEEE Symposium Series on Computational Intelligence (SSCI)*, pp. 1684–1691, Xiamen, China, December 2019.

[47] B. Hu, M. Leitner, and G. R. Raidl, "Combining variable neighborhood search with integer linear programming for the generalized minimum spanning tree problem," *Journal of Heuristics*, vol. 14, no. 5, pp. 473–499, 2008.

[48] J. Kluabwang, D. Puangdownreong, and S. Sujitjorn, "Multipath adaptive tabu search for a vehicle control problem," *Journal of Applied Mathematics*, vol. 2012, Article ID 731623, 20 pages, 2012.

[49] S. Bechikh, N. Belgasmi, L. B. Said, and K. Ghédira, "PHC-NSGA-II: a novel multi-objective memetic algorithm for continuous optimization," in *008 20th IEEE International Conference on Tools with Artificial Intelligence*, vol. 1, pp. 180–189, Dayton, OH, USA, November 2008.

[50] C. Dandois, *Multi-Objective Evolutionary Concept Learner System, [M.S. Thseis]*, University of Namur, 2009.

[51] N. Srinivas and K. Deb, "Multiobjective function optimization using nondominated sorting genetic algorithms," *Evolutionary Computation*, vol. 2, no. 3, pp. 221–248, 1995.

[52] D. E. Goldberg, B. Korb, and K. Deb, "Messy genetic algorithms: motivation, analysis, and first results," *Complex Systems*, vol. 3, no. 5, pp. 493–530, 1989.

[53] C. M. Fonseca and P. J. Fleming, "Multiobjective optimization and multiple constraint handling with evolutionary algorithms. I. A unified formulation," *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans*, vol. 28, no. 1, pp. 26–37, 1998.

[54] A. J. Ruiz-Torres, F. J. López, and J. C. Ho, "Scheduling uniform parallel machines subject to a secondary resource to minimize the number of tardy jobs," *European Journal of Operational Research*, vol. 179, no. 2, pp. 302–315, 2007.

[55] D. W. Kim, D. G. Na, and F. F. Chen, "Unrelated parallel machine scheduling with setup times and a total weighted tardiness objective," *Robotics and Computer-Integrated Manufacturing*, vol. 19, no. 1-2, pp. 173–181, 2003.

[56] E. B. Edis and C. Oguz, "Parallel machine scheduling with flexible resources," *Computers and Industrial Engineering*, vol. 63, no. 2, pp. 433–447, 2012.

[57] S. Polyakovskiy and R. M. Hallah, "Minimizing weighted earliness and tardiness on parallel machines using a multi-agent system," *Operations Research Proceedings*, Springer, 2012.

[58] V. Kayvanfar, I. Mahdavi, and G. H. M. Komaki, "A drastic hybrid heuristic algorithm to approach to JIT policy considering controllable processing times," *Journal of Advanced Manufacturing Technology*, vol. 69, no. 1-4, pp. 257–267, 2013.

[59] Z. Zhang, L. Zheng, N. Li, W. Wang, S. Zhong, and K. Hu, "Minimizing mean weighted tardiness in unrelated parallel machine scheduling with reinforcement learning," *Computers and Operations Research*, vol. 39, no. 7, pp. 1315–1324, 2012.

[60] V. Kayvanfar and E. Teymourian, "Hybrid intelligent water drops algorithm to unrelated parallel machines scheduling problem: a just-in-time approach," *International Journal of Production Research*, vol. 52, no. 19, pp. 5857–5879, 2014.

[61] G. V. Kayvanfar, E. Teymourian, and K. Mashhadi Alizadeh, "Intelligent water drops algorithm on parallel machines scheduling," in *the Proceedings of IEEE 5th International Conference on Industrial Engineering and Operations Management (IEOM)*, Dubai, UAE, 2015.

[62] J. Lin and K. C. Ying, "ABC-based manufacturing scheduling for unrelated parallel machines with machine-dependent and job sequence-dependent setup times," *Computers & Operations Research*, vol. 51, pp. 172–181, 2014.

[63] I. V. Kayvanfar, G. H. M. Komaki, A. Aalaei, and M. Zandieh, "Minimizing total tardiness and earliness on unrelated parallel

machines with controllable processing times," *Computers & Operations Research*, vol. 41, pp. 31–43, 2014.

[64] K. M. Afzalirad and J. Rezaeian, "Resource-constrained unrelated parallel machine scheduling problem with sequence dependent setup times, precedence constraints and machine eligibility restrictions," *Computers & Industrial Engineering*, vol. 98, pp. 40–52, 2016.

[65] H. Soleimani, H. Ghaderi, P. W. Tsai, N. Zarbakhshnia, and M. Maleki, "Scheduling of unrelated parallel machines considering sequence-related setup time, start time-dependent deterioration, position-dependent learning and power consumption minimization," *Journal of Cleaner Production*, vol. 249, article 119428, 2020.

[66] D. O. Shahvari and R. Logendran, "A bi-objective batch processing problem with dual-resources on unrelated- parallel machines," *Applied Soft Computing*, vol. 61, pp. 174–192, 2017.

[67] B. Shahidi-Zadeh, R. Tavakkoli-Moghaddam, A. Taheri-Moghadam, and I. Rastgar, "Solving a bi-objective unrelated parallel batch processing machines scheduling problem: a comparison study," *Computers & Operations Research*, vol. 88, pp. 71–90, 2017.

[68] B. Yepes-Borrero, F. Perea, R. Ruiz, and F. Villa, "Bi-objective parallel machine scheduling with additional resources during setups," *European Journal of Operational Research*, vol. 292, no. 2, pp. 443–455, 2021.

[69] F. Kolahan and V. Kayvanfar, "A heuristic algorithm approach for scheduling of multi-criteria unrelated parallel machines," *World, Academy of Science, Engineering and Technology*, vol. 59, pp. 102–105, 2009.

[70] M. H. F. Zarandi and V. Kayvanfar, "A bi-objective identical parallel machine scheduling problem with controllable processing times: a just-in-time approach," *International Journal of Advanced Manufacturing Technology*, vol. 77, no. 1-4, pp. 545–563, 2015.

[71] Y. Jin and B. Sendhoff, "Pareto-based multiobjective machine learning: an overview and case studies," *EEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 38, no. 3, pp. 397–415, 2008.

[72] M. Afzalirad and J. Rezaeian, "Design of high-performing hybrid meta-heuristics for unrelated parallel machine scheduling with machine eligibility and precedence constraints," *Engineering Optimization*, vol. 48, no. 4, pp. 706–726, 2015.

[73] N. Xie and N. Chen, "Flexible job shop scheduling problem with interval grey processing time," *Applied Soft Computing*, vol. 70, pp. 513–524, 2018.

[74] E. Balas, N. Simonetti, and A. Vazacopoulos, "Job shop scheduling with setup times, deadlines and precedence constraints," *Journal of Scheduling*, vol. 11, no. 4, pp. 253–262, 2008.