JJMIE

Volume 19, Number 4, Dec. 2025
ISSN 1995-6665
Pages 763– 774
https://doi.org/10.59038/jjmie/190406
Editor-in-Chief: Prof. Faris M. AL-Oqla

Jordan Journal of Mechanical and Industrial Engineering

Developing a Hybrid DPSO+LS+SA Method for Smart Single-Machine Scheduling with Sequence-Dependent Setup Times and Learning Effects

Mohammed Said Obeidat[1*], Nader A. Al Theeb[2], Mohammad M. Hamasha[3], Asma'a Omar Qubaiah[2], Abdullah Obeidat[4], Hala AlSliti[2]

[1]*College of Engineering and Technology, American University of the Middle East, Egaila 54200, Kuwait.*
[2]*Industrial Engineering Department, Faculty of Engineering, Jordan University of Science and Technology, Irbid 22110, Jordan.*
[3]*Industrial Engineering Department, Faculty of Engineering, The Hashemite University, Zarqa 13133, Jordan.*
[4]*Systems Science and Industrial Engineering Department, Binghamton University State University of New York, Binghamton, NY 13902-6000*

Received 5 May 2025 *Accepted 8 October 2025*

Abstract

Most existing studies on single-machine scheduling have focused on individual metaheuristic algorithms (e.g., DPSO, SA, or LS) applied in isolation, without exploring their synergistic integration. Particularly, there is a lack of studies dealing with complex scheduling environments that simultaneously consider sequence-dependent setup times and learning effects. Therefore, this study involves the process of proposing a new hybrid methodology that combines the Discrete Particle Swarm Optimization (DPSO), Local Search (LS), and Simulated Annealing (SA) into a unified approach, named DPSO+LS+SA, to handle the scheduling problem of a Single Machine Total Weighted Tardiness (SMTWT). The study focuses on sequence dependent setup times by engaging in a job-agnostic position-based learning model to measure the learning effect. The success of the proposed approach was evaluated using two datasets, revealing that DPSO+LS+SA surpassed DPSO and SA individually, resulting in a superior total weighted tardiness scores and increased computational efficiency. These results offer key insights for the manufacturing and industrial sectors, providing highlights for reducing production time and improving product success rates by effectively meeting delivery deadlines and customer requirements.

Keywords: Single Machine Total Weighted Tardiness; Discrete Particle Swarm Optimization; Local Search; Simulated Annealing; Scheduling.

Nomenclature

j	Job
i	Individuals of population (particles)
t	Iteration or personal best
L_j	Latenessof job j
C_j	Completion time of job j
d_j	Due date of job j
T_j	Tardiness of job j
S_{ij}	Sequence dependent setup time
P_j	Processing time of job j
W_j	Weight of job j
r	Position of job j
$P_{j,r}$	Processing time of job j if it was scheduled in position r
a	Learning index
X^t	Swarm size
X_i^t	Current position of each particle
V_i^t	Current velocity of each particle
P_i^t	Current personal best position of each particle
G^t	Current global best position
r_1, r_2	Uniform random numbers
π_i^t	Permutation of jobs
π_{ij}^t	Assignment of job j of particle i in the permutation at iteration t
f_i^{pb}	Best fitness value
c_1, c_2	Social and cognitive parameters
K	Construction factor
δ	Control parameter
ΔTC	Change in the objective function
β	Experimental constant
α	Reduction rate
TF	Tardiness factor
$z_{i,j,r}$	Binary decision variable
$x_{j,1}$	Binary decision variable

1. Introduction

Industries and manufacturers tend to produce high quality products at a low cost due to the dynamic

* Corresponding author e-mail: Mohammed.Obeidat@aum.edu.kw.

customers' demand and competitive environmental loop [1], [2]. Scheduling and sequencing could be a decision-making method that play an essential role in the service and manufacturing industries [3], [4], [5].

Effective scheduling is an important sustainable requirement in modern competitive environments for meeting the scheduled date and avoiding delay penalties. Dynamic scheduling holds substantial importance across various industries, encompassing domains such as pharmaceuticals and food, among others. In these kinds of industries, the adaptability of optimizing schedules quickly is crucial for guaranteeing the operations effectively and meeting customer requirements [6]. An effective scheduling strategy must manage heavy virtual machines loads, minimize cost consumption, and reduce scheduling delay [7]. Usually, industrial scheduling requires multi-objective optimizations to obtain the Pareto optimal set, which meets the conflicting criteria and the matchless [8].Suitable resource scheduling can minimize the consumption of resources, and boost efficiency, to have high quality products or services at low cost[9], [10].

Resourcescould be in many form including machines in a workplace, workersin a construction site, airport runways, and many others, while tasks could be jobs in a production process, phases in construction projects, and landing and take-offs on an airport[4]. Scheduling is a kind of controlling workload that has been an attractive topic in industries, especially in manufacturing. In industries, the importance of scheduling appears mostly in assigning jobs or activities to machines, to meet due dates and avoid delay penalties[4].

The scheduling problem description includes the machine environment, which could be job, open, and flow shops, along with single or parallel machines, processing characteristics and constraints (processing time, released date, due date, and weight), and objectives to be optimized [4], [11]. Single machine environment is the simplest when compared to other kinds. This kind is one of the techniques used to simplify and dismantle a working environment consisting of complicated machines [4]. The single machine term indicates that one job starts at time zero, which could be processed at a time without any activity contributed with idling, and preemption are not allowed [12], [13]. On the other hand, computation in parallel machine systems is very complicated in solving scheduling and sequencing problems [14]. Furthermore, the parallel machines scheduling problem is considered a generalized form of the single machine one[15].

The scheduling objective function is associated with the job's completion time, which is affected by various performance measures (lateness, earliness, tardiness, and flow time) [8]. Earliness and tardiness have linguistically opposite means, but both have penalties against the promised due date during the production stage. By minimizing the consequences of both earliness and tardiness, companies could enhance delivery reliability and enhance customer satisfaction [16]. The consequences of being delayed are straightforward. Customer dissatisfaction, loss of profit and reputation, and the compensation cost included in contracts are the most common examples of delay penalties. On the other hand, the notification of earliness penalties is not explicit. Most of the concentration is placed on the external cost regardless of the internal one. The earliness cost includes inventory holding cost, insurance cost, and lateness of payment since the product is ready before the deadline [17].

Both lateness and tardiness have the same meaning but differ in mathematical formulas. According to [4], the lateness of job j is defined as $L_j = C_j - d_j$. Here, C_j is the time of completion for job j and d_j is the due date of job j. The positive sign of $L_j$showsthat the job is finished late, while the negative sign represents the earliness (completed early). The difference between lateness and tardiness is that tardiness (T_j) will never be negative, $T_j = max\,(0\,, L_j)$.

Additionally, the setup time is one performance evaluation measure. The setup time is the time needed to prepare the production resources (i.e. machines) for the next job. In scheduling problems, dealing with setup time differs according to the sequence dependency type. In sequence-independent problems, setup times are added directly to the processing times of a job, while these setup times depend on the last scheduled job besides the job being currently scheduled in sequence-dependent problems[4].

Achieving the minimum job weighted tardiness is important, which results in meeting the due date and eliminating the unwanted costs[18], [19]. For this purpose, many exact and heuristic approaches were proposed to solve many variants of scheduling problems, which both have various advantages and limitations. The branch and bound and dynamic programming algorithms are examples of exact solution methods. In addition, examples of these algorithms of heuristics include the Simulated Annealing (SA), Tabu Search (TS),Particle Swarm Optimization (PSO), Genetic Algorithms (GA), Ant Colony Optimization (ACO), and Local Search (LS) algorithms [20].

This study provides a new approach that combines LS, Discrete Particle Swarm Optimization (DPSO), and SA algorithms to deal with a scheduling problem that involves a single machine. The objective is to minimize the total weighted tardiness of jobs while considering the influence of sequence dependent setup times and the learning effect. The integration of these algorithms aims to enhance performance, computation time, and optimization compared to individual approaches. Notably, this research contributes to the existing literature by introducing a solution methodology for the specific problem at hand that incorporates the proposed algorithm combination.

2. Literature Review

2.1. Scheduling Problems with Setup Considerations

The computational complexity and challenges of scheduling problems, especially for huge datasets, have encouraged researchers to develop more sophisticated meta-heuristic and hybrid algorithms that can find near-optimal solutions in some reasonable time periods. This is true for both sequence-dependent setup times and objectives like minimizing weighted tardiness problems.

The investigation of the setup time has been addressed in many studies. Allahverdi, et al. [21] reviewed comprehensively scheduling problems that include setup times. The review covered about 200 papers that were categorized according to the machine environments in the period (1960s – 1988). The paper concluded that the setup

time received the most attention for the single-machine scheduling problems, due to their relative simplicity, while the multi-stage scheduling and parallel machines problems with batch setup times had less attention. Moreover, the weighted number of tardy jobs, total weighted earliness and tardiness, maximum tardiness, and other measures associated with the due date have been considered more in single machine scheduling problems[8], [18]. A second thorough review of scheduling problems with setups consideration, performed by [21]. The review covered the period from mid of 2006 to the end of 2014 including about 500 papers. The researcher concluded that 75% of papers addressed the sequence-independent setup times in the single machine environment. In addition, the review highlighted the need for more research in the association of sequence dependent setup times and single machine environments.

2.2. Scheduling Algorithms

Heuristics and optimizing methods (such as SA, GA, ACO, PSOand TS algorithms) have the most attention for recent research to find optimal or near-optimal solutions, regarding single machine scheduling problems, with sequence dependent setup time. Since they are suitable for solving large-size problems that require high computational efforts. Although several meta-heuristic algorithms have been implemented to single machine scheduling problems with sequence-dependent setup times, a single algorithm usually struggles to balance exploration and exploitation effectively. To remove these individual methods' limitations, hybrid approaches have become the focus of recent research, which combines different algorithms strengths to enhance computational efficiency and solution quality.

2.2.1. The Particle Swarm Optimization (PSO) Algorithm

The PSO algorithm was initially suggested by [22]. For all swarm types, many particles are moving to search for food and better places. This approach is utilized in the coded PSO, in which each moving particle presents a candidate solution. Letm represents the number of particles/solutions in the swarm. Each job (*j*) in each particle (k) has a random initialized velocity ($velo_{kj}$) and position (pos_{kj}) in a search space of n dimension; in that the position array is of m by n size, as each job has a position value in each particle. Based on the fitness/objective values, each particle stores its updated current position (pos), current velocity (velo), local best (LB), which reflects the best positions visited within subgroups of particles, global best (GB), showing the best position visited by all particles, and personal best (PB), which corresponds to the particle's own best value of the objective function [8].

The PSO algorithm is considered as one of the latest meta-heuristic methods, which is currently considered a root-based algorithm [23]. It relatively has a simple concept and structure, easy implementation, speed of obtaining the results, robustness in controlling the parameters, and computational efficiency compared to other techniques [24]. In addition, the PSO algorithm has an improved search capability by means of both the population diversity and the convergence rate [25].

In the PSO algorithm, members or individuals are called particles. Each particle transfers through a multi-dimensional search space, with its velocity and position updated according to both its own experience and the collective experience of the entire swarm. For the problem of single machine scheduling, the particle represents a complete and feasible solution for all jobs sequenced [26], [8].

The DPSO algorithm for the single machine scheduling problem starts with randomly initializing parameters and generating the initial population, then applying the SPV rule on each particle to get the discrete form of PSO (which is called permutation), thus, the fitness value and the total weighted tardiness of each particle could be evaluated [27].The DPSO is a binary version of the standard PSO. The need for this version came from the limited applicability of PSO that is only suitable for continuous problems [28]. Tasgetiren, et al. [29]have studied the possibility of employing the PSO in solving the Single Machine Total Weighted Tardiness(SMTWT) scheduling problems. The problem was a type of typical combinatorial optimization. They used the smallest position value (SPV) rule to facilitate using PSO in discrete combinatorial optimization problems and the attained results were "good"[29]. The applicability of applying the DPSO approach in SMTWT with the availability of sequence dependent setup times[30]. The results showed success that was competitive compared to other algorithms.

2.2.2. Simulated Annealing (SA) Algorithm

The SA algorithm was established by[31].It is afamous metaheuristic method that has been used for the purpose of solving several combinatorial optimization problems. The inspiration for the SA algorithm came from metalwork annealing process. The procedure of SA was established based on the effect of cooling rate on the liquid metal; The atoms of liquid metal form a pure crystal structure in response to the minimum energy produced by the slow cooling rate [32], [31].

The SA algorithm was widely applied using several performance measures. Ghodratnama et al. [33]employed the SA algorithm to solve a multi-objective single-machine scheduling problem. Their aim was targeting the minimization of the sum of total weighted completion times and total weighted tardiness, while maximizing the jobs' total weighted value. The study considered several factors including the repair and maintenance times, the learning effect during the work process, and job deterioration. The praise on the efficiency of the SA algorithm was presented in [34]. This technique achieved excellent results in solving single machine criteria scheduling problems compared to randomly generated schedules.

2.2.3. Local Search Algorithm

Local search (LS) is a heuristic method used to solve optimization problems that are hard to be computed [35]. The LS heuristic is usually used to strengthen the Genetic Algorithm [36]. The procedure of LS stands on moving from one solution to another (neighborhood) to optimize the objective function based on a pre-specified rules[37]. Each solution has another one that is close to it, which is called the neighborhood solution[38]. In [39], a greedy randomized adaptive search procedure (GRASP) with path

re-linking and space-based local search heuristics was proposed for SMTWT, with sequence dependent setup times. The conclusion was that the space-based local search was faster than other algorithms except for using GRASP with path re-linking. On the other hand, the GRASP generally needs additional time than that of the Ant Colony Optimization heuristic, but it usually provides more accurate solutions[39]. Moreover, an iterated local search heuristics algorithm based on a multi-type of perturbation mechanism (ILS-MP) was applied in [40], to minimize the total of quadratic tardiness and linear earliness penalties for the single machine scheduling problem. The ILS-MP approach showed superior performance in terms of both computational efficiency and solution quality in many cases. In a recent study be [41], the researchers developed an optimal overnight centralized charging schedule for battery electric buses, considering both battery degradation and setup time costs. The implemented scheduling problem was modeled as a multi-traveling salesman problem and solved based on a local search grouping genetic algorithm with a 2-opt local search operator. The results showed a 4.48% reduction in operating costs, supporting more sustainable public transport.

2.2.4. Learning Effect

The job's processing time is assumed to be constant in several scheduling problems. However, assuming processing time as a variable that relies on the performance of both workers and machines is more realistic, in that it could be improved by repeating operations; hence, the actual job processing time would be shorter when performed at late time of the schedule [42]. This is known as the learning effect phenomenon. The learning effect in scheduling has been introduced in various models, which are classified into two classes: processing time-based and position-based learnings[43]. The position-based (volume-dependent job processing time function) considers that the actual processing time of a job relies on its scheduled location and is placed only in setups. The other class is the summation of processing time-based learning (a time-dependent job processing time function) accounts for the cumulative experience that workers achieve while performing jobs [42], [44].

The first time involving the learning effect in production stages was in 1966 by [45]. They studied the influence of the learning effect on calculating the optimal production lot size. They found that even though a small rate of learning was applied, a good effect on the optimal production pattern was attained.

2.2.5. Research Gap

Prior studies have not explored this approach. Furthermore, there is a significant gap in comprehensive research focusing on improving the production process by effectively reducing product delivery times through scheduling techniques. The novelty of this study lies in the synergistic integration of the three algorithms, not only their sequential application. The (DPSO) is usually implemented as the primary global search mechanism and it effectively explores a wide range of potential solutions in the search space. The (LS) is applied to the solutions found by DPSO, and it acts as a local exploitation mechanism that can apply fine-tuning of the promising solutions by exploring their immediate neighborhood. The (SA) is integrated at a critical point, offering a mechanism for escaping poor local optima. The results of this study demonstrate that this hybrid approach outperforms the individual algorithms (DPSO, and SA) in terms of computational efficiency and solution quality (lower total weighted tardiness). This reflects the idea that the synergy between the algorithms leads to a better overall performance compared to the sum of their individual parts. Furthermore, the study shows an improved convergence, in that the integration of LS and SA provides faster and more reliable convergence to a near-optimal solution. Moreover, when considering the effective handling of complex Constraints, the proposed methodology is well-suited for solving complex scheduling problems with constraints including the sequence-dependent setup times and learning effects.

3. Problem Description and Formulation

The aim of this study was to discuss and solve the single-machine total weighted tardiness scheduling problem with sequence dependent setup time and learning effect (STWTSDS). The single machine consists of a set of jobs $j = \{1, 2, 3 \ldots n\}$that will be processed[27]. Each job has a sequence dependent setup time (S_{ij}), which should be waited before pursuing a new job, regular processing time (p_j), completion time (C_j), due date (d_j) that is specified from customers, and weight (W_j), which indicates the job importance. The assumption used was that the machine is continuously available; there is no release date, which means that the required jobs are available to be processed from the beginning. In addition, the machine can perform just a single job at a time, and each one is to be processed just once without any interruption and preemption.

Decision Variables:

$x_{j,r}$	$x_{j,r} \in \{0,1\}$: 1 if job j is assigned to position r.
$z_{i,j,r}$	$z_{i,j,r} \in \{0,1\}$: 1 if job i is at position r and job j is at position r + 1
$C_r \geq 0$	Completion time at position r
$C_j \geq 0$	Completion time at position j
$T_j \geq 0$	Tardiness of job j

The objective function of the study is:

$$Minimize \sum_{j \in J} T_j \, w_j \tag{1}$$

Subject to:

$$T_j = Max\left(C_j - d_j \, , 0\right) \quad \forall \, j \in J \tag{2}$$

$$C_{r+1} = C_r + \sum_{i \in J} \sum_{j \in J} z_{i,j,r} \, s_{i,j} + \sum_{j \in J} x_{j,r+1} p_{j,r+1}, \quad r = 1, \ldots, n-1 \tag{3}$$

With

$$C_1 = \sum_{j \in J} x_{j,1} (S_{0,j} + p_{j,1})$$

$$p_{j,r} = p_j r^a, a = log_2(LR), \; a \leq 0 \tag{4}$$

$$\sum_{r=1}^{n} x_{i,r} = 1, \qquad \forall\, j \in J \tag{5}$$

$$\sum_{i \in J} x_{i,r} = 1, \qquad \forall \mathrm{r} = 1, \dots, \mathrm{n} \tag{6}$$

The objective function is subject to constraints explained as follows:

1. Constraint 2: is a non-linear constraint that defines the tardiness time of each job j.
2. Constraint 3: defines the completion time for each job.
3. Constraint 4: determines each job processing time. The position-based learning model was used. According to [46], r is the position of the job j in the sequence, p_{jr} is the processing time of job j if it was scheduled in position r in a sequence, p_j the normal processing time for job j if it is scheduled first in a sequence, and a is the learning index which is a random variable usually a ≤0, given that the logarithm is to the base 2 of the learning rate.
4. Constraint 5, which is used to ensure that each job must be processed only once.
5. Constraint 6, which is used to ensure that each job should be succeeded and proceeded by only one single job.

4. Methodology

The aim of this study was to minimize the total weighted tardiness scheduling problem with sequence dependent setup time and the learning effect. A combination of DPSO, LS, and SA was presented. The following subsections provide more details.

4.1. The PSO Algorithm

According to [27], the population size NP denotes the swarm size represented as $X^t = [\, X_1^t, X_2^t, \dots, X_{NP}^t]$. Subsequently, each particle within the swarm population will possess the subsequent notations: current position represented as $X_i^t = [x_{i1}^t, x_{i2}^t, \dots, x_{in}^t]$, current velocity $V_i^t = [v_{i1}^t, v_{i2}^t, \dots, v_{in}^t]$, best location of the current personal represented as $P_i^t = [p_{i1}^t, p_{i2}^t, \dots, p_{in}^t]$, and a current global best location presented as $G^t = [g_1^t, g_2^t, \dots, g_n^t]$. The following steps were presented by[27], [26]:

Step I: Initialization

- Set the initial value of t (iteration or personal best) = 0, NP = ten times the dimensions' number.
- Randomly generate the NP particles based on the equation $\{X_i^0, i = 1, 2, \dots, NP\}$, where $X_i^0 = [x_{i1}^0, x_{i2}^0, \dots, x_{in}^0]$. The next equation is utilized to generate the initial value of continuous position: $[x_{ij}^0 = x_{min} + (x_{max} - x_{min}) * r_1]$, where $x_{min} = -1, x_{max} = 1$ and r_1 is a uniform random number between 1 and 0.
- Randomly generate the initial velocities for particle i using the equation $\{V_i^t, i = 1, 2, \dots, NP\}, where\ V_i^0 = [v_{i1}^0, v_{i2}^0, \dots, v_{in}^0]$. The next equation is utilized to generate the initial continuous velocity value: $[v_{ij}^0 = v_{min} + (v_{max} - v_{min}) * r_2]$, where $v_{min} = -1$, $v_{max} = 1$ and r_2 is a uniform random number between 1 and 0.
- The smallest position value (SPV) is used on particles to convert the problem into discrete job permutation by introducing a new variable π_i^t,which is a permutation of implied jobs by the particle X_i^t. It might be addressed as $\pi_i^t = [\pi_{i1}^t, \pi_{i2}^t, \dots, \pi_{in}^t]$, where $\pi_{ij}^t$is the assignment of job j of particle $i$in the permutation at iteration t.
- Calculate the value of the objective function for each particle i in the swarm based on the objective function $f_i^0$for $i = 1, 2, \dots, NP$. Where:
  $f_i^t\left(\pi_i^t \leftarrow X_i^t\right) = \sum_{j=1}^{n} w_{ij} T_{ij}$
- For any particle$i$in the swarm, set the personal best to $P_i^0 = X_i^0$, where $P_i^0 = \left[p_{i1}^0 = x_{i1}^0, p_{i2}^0 = x_{i2}^0, \dots, p_{in}^o = x_{in}^0\right]$,along with its best fitness value, $f_i^{pb} = f_i^0$ for $i = 1, 2, \dots, NP$.
- Find the value of best fitness among the whole swarm such that $f_l = \min f_i^0$for $i = 1, 2, \dots, NP$along with its corresponding positions X_l^0.
- Set the global best to $G^0 = X_l^0$ such that $G^0 = \left[g_1 = x_{1,1}, g_2 = x_{1,2}, \dots, g_n = x_{l,n}\right]$with its fitness value $f^{gb} = f_l$.

Step II: Update counter iteration

$$t = t + 1$$

Step III: Inertia weight updating

- The updating technique used to balance exploration and exploitation in this work was introduced in [47]. The inertia weight decreases linearly in the range [$\omega_{max}$to ω_{min}] during iterations as:

$$\omega_n = \omega_{max} - \left(\frac{\omega_{max} - \omega_{mim}}{NGenration}\right) \times n$$

Step IV: Velocity updating

- Three equations were used to update the velocity introduced by [47], [48], and [49], respectively:

$$\begin{aligned} v_i^n = {} & \left(w_n \times v_i^{n-1}\right) \\ & + \left(c_1 \times r_1^n \right. \\ & \left. \times \left(p_i^{n-1} - x_i^{n-1}\right)\right) \\ & + \left(c_2 \times r_2^n \right. \\ & \left. \times \left(g^{n-1} - x_i^{n-1}\right)\right) \end{aligned} \tag{7}$$

$$\begin{aligned} v_i^n = K\Big[& v_i^{n-1} + \left(c_1 \times r_1^n \right. \\ & \left. \times \left(p_i^{n-1} - x_i^{n-1}\right)\right) \\ & + \left(c_2 \times r_2^n \right. \\ & \left. \times \left(g^{n-1} - x_i^{n-1}\right)\right)\Big] \end{aligned} \tag{8}$$

Where $K = \frac{2}{\left|2 - \varphi - \sqrt{\varphi^2 - 4\varphi}\right|}, \varphi = c_1 + c_2$

  - Volume 19, Number 4 (ISSN 1995-6665)

$$v_i^n = K\Big[\big(w_n \times v_i^{n-1}\big) + \Big(c_1 \times r_1^n \times \big(p_i^{n-1} - x_i^{n-1}\big)\Big) + \Big(c_2 \times r_2^n \times \big(g^{n-1} - x_i^{n-1}\big)\Big)\Big] \quad (9)$$

Where K is a construction factor that controls the convergence behavior of particles, $c_1$and c_2, whichare the social and cognitive parameters of values between (1, 4), $r_1$and $r_2$are the uniform random numbers between (0, 1). In this work, the simplest form for updating velocity of [47]was used.

Step V: Update the postion

$$x_i^n = v_i^n + x_i^{n-1} \quad (10)$$

Step VI: Calculating the permutation

- Finding the the permutation using SPV rule, $\pi_i^t = [\pi_{i1}^t, \pi_{i2}^t, \dots, \pi_{in}^t]$for $i = 1, 2, \dots, NP$.

Step VII: Updating the personal best

- Every particle is evaluated using the permutation to determine whether the personal best will be improved. Hence, if $f_i^t < f_i^{pb}$for $i = 1, 2, \dots, NP$. Then the personal best will be updated as $P_i^t = X_i^t$ and $f_i^{pb} = f_i^t$ for $i = 1, 2, \dots, NP$.

Step VIII: Updating the global best

- Finding the minimum personal best value. That is $f_l^t = \min\{f_i^{pb}\}$, for $i = 1, 2, \dots, NP$; where $l \in \{i; i = 1, 2, \dots, NP\}$.
- If $f_l^t < f^{gb}$, then the global best is updated as $G^t = X_l^t$ and $f^{gb} = f_l^t$.

Step IX: Stopping criterion

- Once the number of iterations exceeds the maximum iterations' number, then stop; otherwise move to step II.

4.2. SA Algorithm

The SA algorithm, which was introduced by [31], is considered as one of the most usable methods for solving combinatorial optimization problems. The following is the SA procedure:

Step I: Setting the initial values and creating a random initial solution. In this study, a combination of the DPSOand LS algorithms was used to obtain the initial solution.

Step II: The algorithm selects a random solution that is close to the current one, measures its quality, and then decides upon moving to it or staying with the current solution. The selected solution is being accepted if the value of the objective function is smaller than or equal to that of the original one. Alternatively, a solution can be accepted with a certain probability that reduces as the process progresses. The probability of acceptance is derived from an exponential distribution, as follows:

$$P(accept) = \exp(-\delta \times \Delta TC) \quad (11)$$

Where δ is the parameter to be controlled, and ΔTC is the change in the objective function. The method of changing δ at the kth iteration is obtained from [50]and is presented by:

$$\delta = \frac{k}{\beta} \quad (12)$$

Where β is experimental constant.

Step III: Continue decreasing the temperature slowly; accept fewer bad moves at each temperature level until the stopping condition is met. Resulting in either the system is sufficiently cooled, or a good enough solution being found.

4.3. LS Algorithm

The algorithm begins with a candidate solution $(S)$and continues to move into a neighboring solution until the optimal solution is reached. The solution will be analyzed based on the fitness function. The smaller value of the fitness function will be accepted, otherwise it will be rejected or time-bound elapsed [35], [4].

According to [37] and [4],the local search procedure is summarized as:

Step I: Initialization.

- The initial schedule is considered as the current solution and will be subjected to the objective function.

Step II: Neighbor generation

- Selecting a neighbor solution of the current one and subjecting it to the objective function.

Step III: Acceptance test

- The movement from the current to the neighbor solution will be subjected to an acceptance test; the smaller value will be accepted.

Step IV: Termination test

- The termination is applied when computation time exceeds the prespecified limit or after the prespecified number of iterations is completed. In other words, the best solution is obtained; otherwise, another neighbor solution will be generated.

Generating the neighbor's solutions are found by applying local changes of the following types:

1. Transpose neighborhood, where two adjacent jobs exchange their position:
   (1, 2, 3, 4, 5, 6, 7) → (1, 3, 2, 4, 5, 6, 7)
2. Swap neighborhood, where two arbitrary jobs change their position:
   (1, 2, 3, 4, 5, 6, 7) → (1, 6, 3, 4, 5, 2, 7)
3. Insert (shift) neighborhood; this is done by changing the position of specific job:
   (1, 2, 3, 4, 5, 6, 7) → (1, 3, 4, 5, 6, 2, 7)

After several computational experiments, the efficiency of the insert (shift) and transpose neighborhood method has not been proven in single machine scheduling, however, using the variable neighborhood descent (VND) together with the swap and insert neighborhood resulted in an effective time consumption. Therefore, the neighborhood generation in this paper followed the simple swap neighborhood.

5. Parameters Inserting and Dataset Generation

The DPSO, LS and SA techniques have been merged into one algorithm to obtain a better solution. The

parameters of each algorithm and the way of generating the datasets used are shown in this section.

5.1. Algorithms and their Parameters

This study follows a specific approach of merging the algorithms as follows:

1. The DPSO Algorithm. Here, the parameters used were; the population size that was 100, the number of generations that was 500, social and cognitive parameters,c_1 and c_2, whichwere random integer numbers in the range of 1 to 4, as most of the studies had recommended, the uniform parameters r_1 and r_2, whichwere random in the range of 0 to1, and the initial inertia weight that was set to $w_0 = 1$ and in the range of (0, 1).
2. DPSO with LS.
3. DPSO + LS + SA. For the SA algorithm, the number of iterations was set to 250, the number of sub-iterations was 15, the initial temperature T_0 was 0.025, the temperature reduction rateα was 0.99, and the decrement factor $\beta$was 0.975.

5.2. Datasets Generation

The generation of datasets was divided into two categories in response to evaluating the performances of the proposed algorithms. The following subsections provide more details.

5.2.1. Data Generation I

The performances of the proposed algorithms were evaluated using 150 datasets generated randomly using MATLAB and saved in MS-Excel sheets. Each data set includes six sheets; one sheet for first setup time that includes n values (n is the number of jobs) generated randomly in the range of (1, 40), one sheet for sequence dependent setup time with the size of $n \times n$; ($n \times n$ represents a **matrix** where every job is paired with every other job), here, sequence dependent means that the setup time for job (i) depends on the job just finished before (j), for example, if the plan is to setup time for job 5, and the job that it is just finished was 10, then $S_{5\,10} = 20\ minutes$; one sheet for normal processing time that includes n values generated randomly in the range of (50, 150), one more sheet for the processing time calculations using [46] equation of $n \times n$ size; where the values of the learning index a are selected randomly in the range of (-0.2, -0.6). In addition, another sheet for the weight that includes n values, which are generated randomly in the range of (1, 10). Finally, one sheet for the due date that contains n values generated randomly in the following range of:

$$(\{0.3\times (sumofsetuptimes + sumofthediagonalvaluesoftheprocessingtimes)\},\{0.8\times (sumofsetuptimes + sumofthediagonalvaluesoftheprocessingtimes)\}) \quad (13)$$

Those 150 datasets were classified into three groups; each group contains 50 datasets that differ in the number of jobs. The jobs of each group were assigned to every 50 datasets and selected randomly from ranges of (10, 30),(31, 80), and (81, 150), respectively. This range provides that there were small, medium, and large groups.

5.2.2. Data **Generation** *II*

The performances of the proposed algorithms were evaluated based on the benchmark set of datasets that were randomly generated, available from the OR Library [51]. The datasets in the library were generated considering this procedure: For each job $j, (j = 1 \ldots n)$, an integer processing weight $w_j$was also generated from the uniform distribution of the range (1, 10), and an integer processing time $P_j$was generated from the uniform distribution of the range (1, 100). For a given average tardiness factor(TF),$(TF$ = 0.2, 0.4, 0.6, 0.8, 1.0) along with a given relative range of due dates $(RDD), (RDD = 0.2, 0.4, 0.6, 0.8, 1.0)$. The integer due date $d_j$for each job j was generated randomly using a uniform distribution:

$$[\,P\left(1 - TF - \frac{RDD}{2}\right), P\left(1 - TF + \frac{RDD}{2}\right)],$$

$$\text{where } P = \sum_{j=1}^{n} p_j \quad (14)$$

Three values of Range of Due Dates(RDD) were selected: (0.2, 0.6, and1.0), and the value of TF was equal to 0.6, and combination of the RDD and TF generates nine datasets for the three number of jobs $n = 20, 60, and\ 100$) as shown in Table 1.

Table 1. The number of jobs and the values of RDD and TF assigned to datasets.

Dataset No.	No. of Jobs	RDD	TF
1	20	0.2	0.6
2	20	0.6	0.6
3	20	1	0.6
4	60	0.2	0.6
5	60	0.6	0.6
6	60	1	0.6
7	100	0.2	0.6
8	100	0.6	0.6
9	100	1	0.6

6. Computational Results

The MATLAB software was used to code the merging of DPSO, LS, and SA algorithms for the SMTWT problem along with learning effect and sequence dependent setup time. The generated datasets were used to evaluate the performances of DPSO, DPSO+LS, SA, DPSO+LS+SA. Afterwards, a comparison was made between these performances.

6.1. Data Generation I Results

The generation code had been run five times for the three groups. For simplicity, three datasets were randomly selected of each group as follows: the small group of $n = 10, n = 20\ and\ n = 30$, the median group of$n = 50, n = 60\ and\ n = 80$, and the large group of $n = 100, n = 120\ and\ n = 140$.

The maximum tardiness, minimum tardiness, average tardiness, maximum time, minimum time, and average time values were evaluated and recorded for each dataset considering the following sequence: DPSO, DPSO+LS, SA, and DPSO+LS+SA. The results for the average tardiness are presented in Figure 1 for the different algorithms.

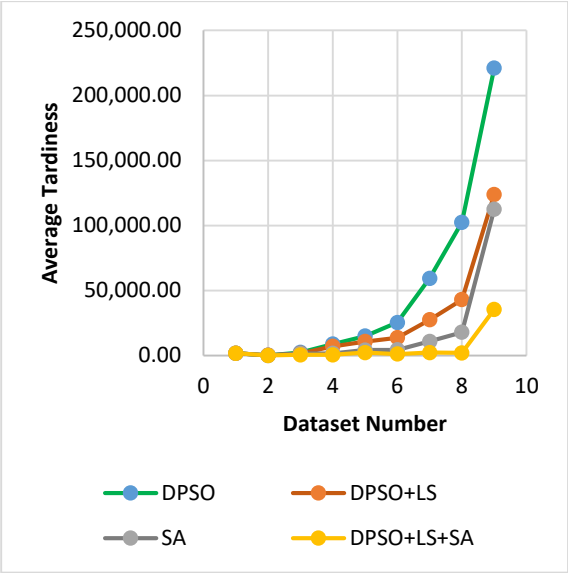

Figure 1. The MATLAB results of the four algorithms for data generations I.

Considering the outcomes, the second dataset yielded the lowest average tardiness value while considering the specified sequence for small groups, with the following achieved values: of 62.5, 91.31, 27.89, and 3.96, associated with the average executing time was 0.6, 2.89, 0.67, and 0.66, respectively. For the medium group, the fourth dataset in all groups generated the minimum value of average tardiness of 8,742.05, 6,996.87, 1,592.82, and 442.55, respectively, with average executing times of 0.89 in the performance evaluation of DPSO and 7.9 in the combination of DPSO+LS. In contrast, the value was 0.86 for both SA and DPSO+LS+SA. It is clearly shown that the DPSO+LS+SA combination provides the minimum average tardiness.

For the large groups, the minimum value of average tardiness for the DPSO, DPSO+LS, and SA was achieved in the seventh dataset with the following values: 59,382.68, 27,444.25, and 10,818.99, while in the combination of DPSO+LS+SA the value was 1,917.58 in the eighth dataset, and the average executing times were 1.5, 23.24, 1.30, and 1.55, respectively.

The boxplot was used to provide an accurate description of the distribution of the achieved results as shown in Figure 2.According to Figure 2, the proposed algorithm has reduced the total weighted tardiness of various amounts of jobs. The time consumed for executing DPSO, SA, and DPSO+LS+SA algorithms were nearly the same for the different number of jobs, but the execution time of DPSP+LS was the highest in all ranges.

6.2. Data Generation II Results

The generation code was run twelve times for a total of nine datasets. The maximum tardiness, minimum tardiness, average tardiness, maximum time, minimum time, and average time values were calculated for each dataset and recorded using DPSO, DPSO+LS, SA, and DPSO+LS+SA, respectively. Figure 3 shows the results of the average tardinessfpr the different algorithms.

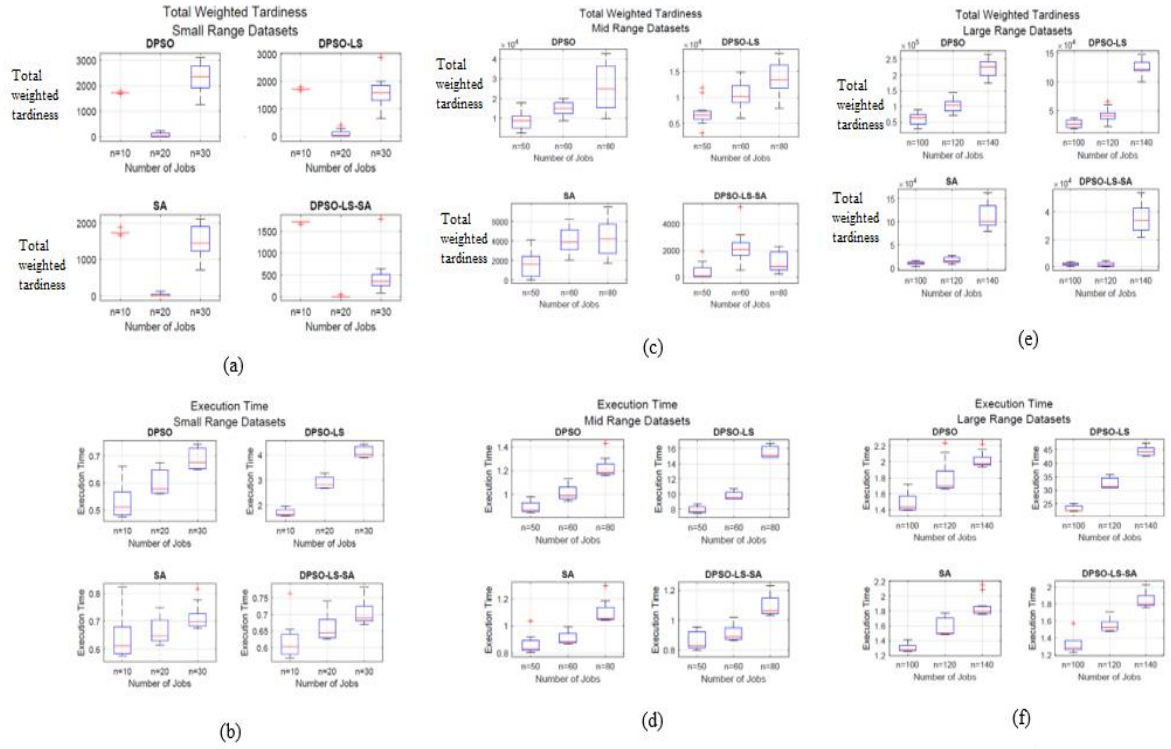

Figure 2. Boxplot of total weighted tardiness and execution times for all dataset ranges of generation I; (a) and (b) for small range dataset, (c) and (d) for the medium-range dataset, and (e) and (f) for large range dataset.

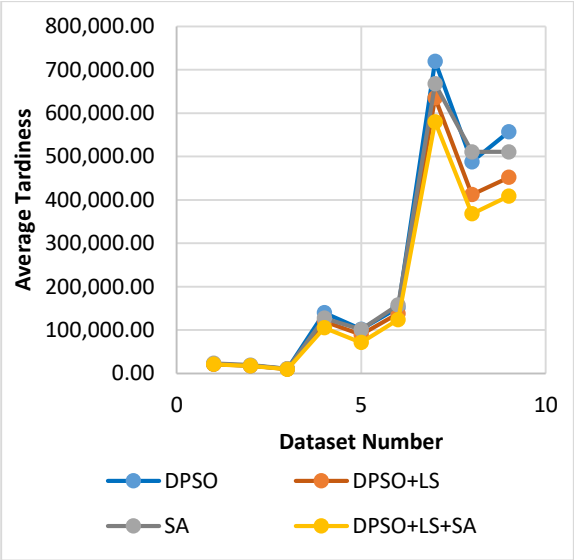

Figure 3. The MATLAB results of the four algorithms for data generation II.

Based on the results in Figure 3, the minimum value of average tardiness considering the small group in the mentioned sequence had been achieved in the third dataset with values of 10,423.58, 9,895.25, 10,724.58, and 9,736.75, while the average executing time was 0.56, 2.65, 0.62 and 0.62, for the DPSO, DPSO+LS, SA, and DPSO+LS+SA, respectively. For the medium group, the fifth dataset in all groups generated the minimum value 102,109.1, 89,333.17, 101,307.8, and 71,119.58, for the DPSO, DPSO+LS, SA, and DPSO+LS+SA, respectively, with average executing times of 0.88 in the performance evaluation of DPSO and 0.86 in combination of DPSO+LS. In contrast, the value was 0.71 for both SA and DPSO+LS+SA. In the large group, the minimum values of average tardiness were achieved in the eighth dataset; 487,786.6, 412,083.4, 368,022.2, and 368,022.2, and the average executing times were 1.5, 23.24, 1.30, and 1.55, respectively. Figure 4 shows the boxplot of total weighted tardiness values and execution times for all dataset ranges of generation II.

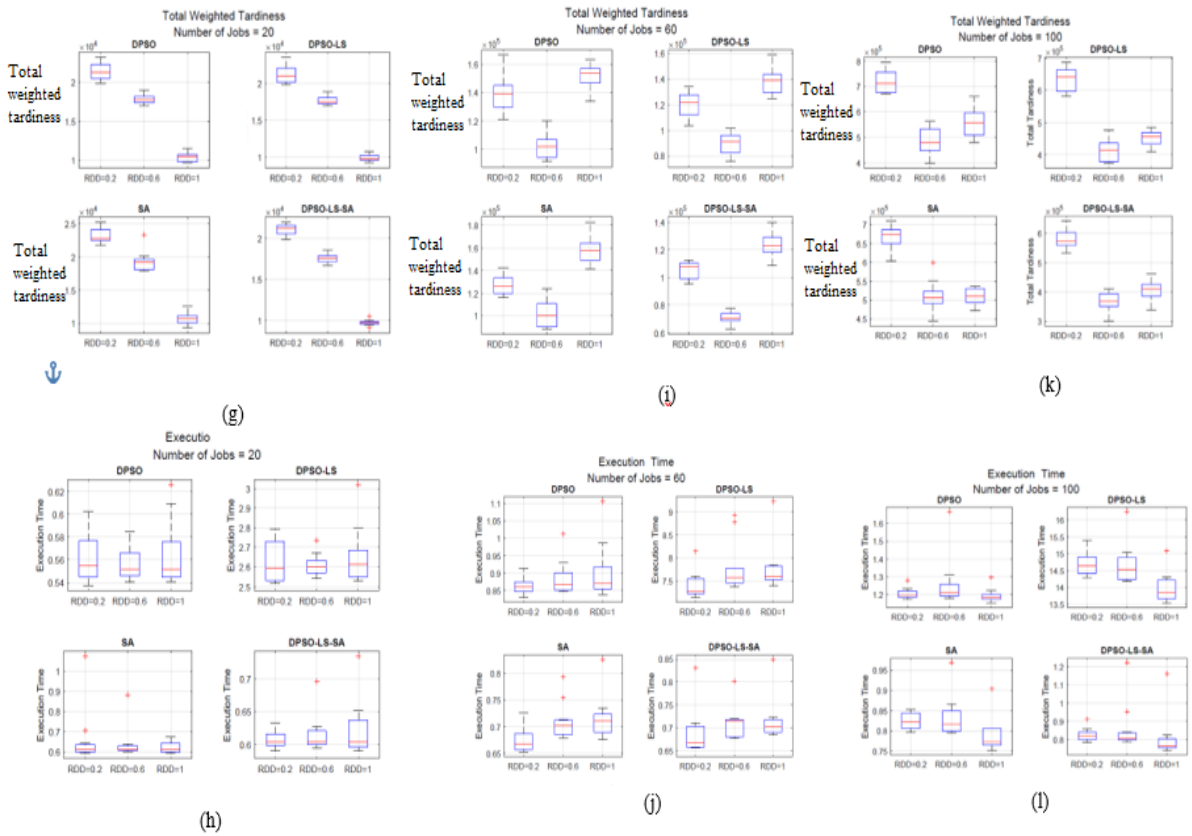

Figure 4. The boxplot of total weighted tardiness values and execution times for all dataset ranges of generation II; (g) and (h) for small range dataset, (i) and (j) for the medium-range dataset, and (k) and (l) for large range dataset.

Based on Figure 4, the proposed algorithms had reduced the total weighted tardiness of various amounts of jobs. In addition, the total weighted tardiness decreases for each algorithm as the RDD increases. The times consumed for executing DPSO, SA, and DPSO+LS+SA algorithms were nearly the same for different values of RDD, but the execution time of DPSP+LS was the highest.

7. Limitations

Although the results clearly indicate that DPSO+LS+SA performs better than DPSO and SA across all datasets, this study did not include formal statistical tests such as the Wilcoxon signed-rank test. The focus was on developing and demonstrating the proposed hybrid algorithm. In future work, the plan is to carry out more detailed statistical analyses and broaden the comparisons to include additional state-of-the-art algorithms to further validate the effectiveness of the approach.

8. Conclusions

The DPSO algorithm has emerged as a promising evolutionary optimization method, widely applied in various domains. However, it faces limitations when addressing a single machine's total weighted tardiness scheduling problem with sequence dependent setup times. In such cases, the DPSO algorithm often struggles to escape local optima in high-dimensional spaces and exhibits a low convergence rate during the iterative process. To overcome these challenges and improve the DPSO algorithm's performance, this study incorporates a combination of DPSO, LS, and SA algorithms. By integrating these techniques, the algorithm offers enhanced diversity and a better ability to converge towards optimal solutions.

The study utilizes actual data to initialize the position and velocity values of the DPSO algorithm, incorporating job weights, due dates, and processing times. This approach reduces randomness and expedites the search for improved solutions. Additionally, the algorithm is hybridized with a neighborhood search technique to further enhance solution quality. Finally, the SA algorithm is applied to refine the obtained solution.

To evaluate the performance of the proposed algorithm, two sets of data were used: one generated randomly using MATLAB, and the other obtained from the OR library. The results demonstrate that the first data generations outperformed the latter in terms of total weighted tardiness values. Moreover, the combination of DPSO, LS, and SA algorithms exhibited superior improvements compared to the standalone DPSO algorithm.

Overall, this research contributes to advancing the capabilities of the DPSO algorithm for addressing single machine total weighted tardiness scheduling problems with sequence dependent setup times. The proposed algorithm shows promise in achieving optimized scheduling solutions and improving production processes. This study offers practical value for managers in the industrial and manufacturing sectors by providing a hybrid DPSO+LS+SA scheduling method, which reduces total weighted tardiness while accounting for real-world complexities such as learning effects and sequence-dependent setup times.

As an extension to this study, future research should incorporate rigorous statistical testing (i.e., Analysis of Variance (ANOVA) and Wilcoxon signed-rank test), to validate the robustness of the proposed hybrid DPSO+LS+SA scheduling approach using diverse datasets. In addition, testing the proposed approach in multi-machine, job shop, or flow shop scheduling conditions would expand its applicability to more complex industrial and manufacturing systems. Moreover, future research might enhance the adaptive or self-learning mechanisms, which automatically tune algorithm parameters, thus enhancing real-time decision-making in smart industrial and manufacturing settings.

Funding

This work was not funded.

Authors' Contributions

Mohammed Said Obeidat, and Nader A. Al Theeb: conceptualization, formal analysis, investigation, methodology, project administration, resources, supervision, validation, visualization, writing - original draft, and writing - review & editing. **Asma'a OmarQubaiah:** conceptualization, data curation, investigation, methodology, resources, software, validation, visualization. **Mohammad M. Hamasha, Abdullah Obeidat, and HalaAlsliti:** writing - original draft, and writing - review & editing.

Ethical approval

This article does not contain any studies with human participants or animals performed by any of the authors.

Conflict of interest

The authors declare no competing interests.

References

[1] Q. Khalid, S. Azim, M. Abas, A. Babar and I. Ahmad, "Modified particle swarm algorithm for scheduling agricultural products," *Engineering Science and Technology, an International Journal*, vol. 24, no. 3, p. 2, 2021.

[2] M. Obeidat, O. AlShraiedeh, D. Al-Quran and H. Alsliti, "The effects of COVID-19 pandemic on the e-shopping logistics in Jordan," *International Journal of Logistics Systems and Management*, vol. 49, no. 1, pp. 123-143, 2024.

[3] S. Goren and I. Sabuncuoglu, "Robustness and stability measures for scheduling: Single-machine environment," *IIE Transactions*, vol. 40, no. 1, pp. 66-83, 2008.

[4] M. Pinedo, Scheduling: Theory, algorithms, and systems, 6th ed., New York, NY, USA: Springer, 2022.

[5] M. Obeidat, E. Al Abed Alhalim and B. Melhim, "Systematic approach for selecting a cleaning method to solar panels based on the preference selection index approach," *Jordan Journal of Mechanical and Industrial Engineering*, vol. 14, no. 3, pp. 279 - 287, 2020.

[6] W. Bouazza, Y. Sallez and D. Trentesaux, "Dynamic scheduling of manufacturing systems: a product-driven approach using hyper-heuristics," *International Journal of Computer Integrated Manufacturing,* vol. 34, no. 6, pp. 641-665, 2021.

[7] P. Pabitha, N. K., C. Gunavathi and B. Panjavarnam, "A chameleon and remora search optimization algorithm for handling task scheduling uncertainty problem in cloud computing," *Sustainable Computing: Informatics and Systems,* vol. 41, p. 100944, 2024.

[8] N. Al Theeb, M. Obeidat, M. Aljarrah and T. Alhwiti, "Solution approach for solving multi-objective single-machine problem with sequence-dependent setup times and setup costs," *Industrial Engineering and Management Systems,* vol. 18, no. 4, pp. 728-740, 2019.

[9] C. Yang, F. Liao, S. Lan, L. Wang, W. Shen and H. G.Q., "Flexible resource scheduling for software-defined cloud manufacturing with edge computing," *Engineering,* vol. 22, pp. 60-70, 2023.

[10] M. Obeidat and M. Hayajneh, "Implementing an insert-node heuristic travelling salesman problem: a case study to reduce the travelling cost of a medical representative," *International Journal of Logistics Systems and Management,* vol. 33, no. 1, pp. 59-72, 2019.

[11] K. C. Tan, "Minimizing tardiness on a single processor with sequence-dependent setup times: a simulated annealing approach," *Omega,* vol. 25, no. 6, pp. 619-634, 1997.

[12] V. Portougal and D. Trietsch, "Setting due dates in a stochastic single machine environment," *Computers & Operations Research,* vol. 33, no. 6, pp. 1681-1694, 2006.

[13] C. Koulamas, "The single-machine total tardiness scheduling problem: Review and extensions," *European Journal of Operational Research,* vol. 202, no. 1, pp. 1-7, 2010.

[14] K. Abdellafou, H. Hadda and O. Korbaa, "An improved Tabu search meta-heuristic approach for solving scheduling problem with non-availability constraints," *Arabian Journal for Science and Engineering,* vol. 44, p. 3369–3379, 2019.

[15] G. Bektur, "An NSGA-II-Based memetic algorithm for an energy-efficient unrelated parallel machine scheduling problem with machine-sequence dependent setup times and learning effect," *Arabian Journal for Science and Engineering,* vol. 47, pp. 3773-3788, 2022.

[16] P. Palominos, M. Mazo, G. Fuertes and M. Alfaro, "An improved marriage in honey-bee optimization algorithm for minimizing earliness/tardiness penalties in single-machine scheduling with a restrictive common due date," *Mathematics,* vol. 13, no. 3, p. 418, 2025.

[17] V. Portougal and D. Trietsch, "Setting due dates in a stochastic single machine environment," *Computers & Operations Research,* vol. 33, no. 6, pp. 1681-1694, 2006.

[18] H. Kellerer, K. Rustogi and V. Strusevich, "A fast FPTAS for single machine scheduling problem of minimizing total weighted earliness and tardiness about a large common due date," *Omega,* vol. 90, p. 101992, 2020.

[19] L. Wan, J. Mei and J. Du, "Two-agent scheduling of unit processing time jobs to minimize total weighted completion time and total weighted number of tardy jobs," *European Journal of Operational Research,* vol. 290, no. 1, pp. 26-35, 2021.

[20] M. F. Tasgetiren, M. Sevkli, Y.-C. Liang and G. Gencyilmaz, "Particle swarm optimization algorithm for single machine total weighted tardiness problem," in *Proceedings of the 2004 Congress on Evolutionary Computation (IEEE Cat. No.04TH8753)*, Portland, OR, USA, 2004.

[21] A. Allahverdi, "The third comprehensive survey on scheduling problems with setup times/costs," *European Journal of Operational Research,* vol. 246, no. 2, pp. 345-378, 2015.

[22] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proceedings of ICNN'95 - International Conference on Neural Networks*, Perth, WA, Australia, 1995.

[23] M. Malik, D. Nandan, C. Prabha, M. Uddin, B. Acharya and Y. Hu, "A bio-inspired metaheuristic approach for cloud task scheduling using lateral hyena based particle swarm optimization," *Multimedia Tools and Applications,* vol. 84, pp. 20023-20046, 2025.

[24] K. Lee and J. Park, "Application of particle swarm optimization to economic dispatch problem: Advantages and disadvantages," in *2006 IEEE PES Power Systems Conference and Exposition*, Atlanta, GA, USA, 2006.

[25] W. Liu, Z. Wang, N. Zeng, Y. Yuan, F. Alsaadi and X. Liu, "A novel randomised particle swarm optimizer," *International Journal of Machine Learning and Cybernetics,* vol. 12, p. 529–540, 2021.

[26] N. Al Theeb and T. Alhwiti, "Solving single-machine weighted tardiness and total setup costs scheduling problem with sequence dependant setup times and sequence dependant setup cost using discrete particle swarm," in *2979-2988*, Montreal, Canada, 2014.

[27] M. Tasgetiren, Y. Liang, M. Sevkli and G. Gencyilmaz, "Particle swarm optimization and differential evolution for the single machine total weighted tardiness problem," *International Journal of Production Research,* vol. 44, no. 22, pp. 4737-4754, 2006.

[28] H. Noory, A. Liaghat, M. Parsinejad and O. Haddad, "Optimizing irrigation water allocation and multicrop planning using discrete PSO algorithm," *Journal of Irrigation and Drainage Engineering,* vol. 138, no. 5, pp. 437-444, 2012.

[29] M. F. Tasgetiren, M. Sevkli, Y.-C. Liang and G. Gencyilmaz, "Particle swarm optimization algorithm for single machine total weighted tardiness problem," Portland, OR, USA, 2004.

[30] D. Anghinolfi and M. Paolucci, "A new discrete particle swarm optimization approach for the single-machine total weighted tardiness scheduling problem with sequence-dependent setup times," *European Journal of Operational Research,* vol. 193, no. 1, pp. 73-85, 2009.

[31] S. Kirkpatrick, C. D. Gelatt Jr and M. P. Vecchi, "Optimization by simulated annealing," *SCIENCE,* vol. 220, no. 4598, pp. 671-680, 1983.

[32] B. Suman and P. Kumar , "A survey of simulated annealing as a tool for single," *Journal of the Operational Research Society,* vol. 57, no. 10, p. 1143–1160, 2006.

[33] A. Ghodratnama, M. Rabbani, R. Tavakkoli-Moghaddam and A. Baboli, "Solving a single-machine scheduling problem with maintenance, job deterioration and learning effect by simulated annealing," *Journal of Manufacturing Systems,* vol. 29, no. 1, pp. 1-9, 2010.

[34] E. Karasakal and M. Köksalan, "A simulated annealing approach to bicriteria scheduling problems on a single machine," *Journal of Heuristics,* vol. 6, no. 3, p. 311–327, 2000.

[35] E. G. Negenman, "Local search algorithms for the multiprocessor flow shop scheduling problem," *European Journal of Operational Research,* vol. 128, no. 1, pp. 147-158, 2001.

[36] A. Rezaeipanah, S. Matoori and G. Ahmadi, "A hybrid algorithm for the university course timetabling problem using the improved parallel genetic algorithm and local search," *Applied Intelligence,* vol. 51, p. 467–492, 2021.

[37] M. Pirlot, "General local search methods," *European Journal of Operational Research,* vol. 92, no. 3, pp. 493-511, 1996.

[38] D. Johnson, C. Papadimitriou and M. Yannakakis, "How easy is local search?," *Journal of Computer and System Sciences*, vol. 37, no. 1, pp. 79-100, 1988.

[39] S. Gupta and J. Smith, "Algorithms for single machine total tardiness scheduling with sequence dependent setups," *European Journal of Operational Research*, vol. 175, no. 2, pp. 722-739, 2006.

[40] T. Qin, B. Peng, U. Benlic, T. Cheng, Y. Wang and Z. Lü, "Iterated local search based on multi-type perturbation for single-machine earliness/tardiness scheduling," *Computers & Operations Research*, vol. 61, pp. 81-88, 2015.

[41] Z. Wang, F. Zheng, S. Hamdan and O. Jouini, "Charging scheduling optimisation of battery electric buses with charging setup time," *International Journal of Production Research*, vol. 63, no. 10, pp. 3538-3563, 2025.

[42] D. Yang, T. E. Cheng and W. Kuo, "Scheduling with a general learning effect," *The International Journal of Advanced Manufacturing Technology*, vol. 67, no. 1, p. 217–229, 2013.

[43] X. Xin, Q. Jiang, C. Li and K. Chen, "Permutation flow shop energy-efficient scheduling with a position-based learning effect," *International Journal of Production Research*, vol. 61, no. 2, p. In Press, 2023.

[44] D. Biskup, "A state-of-the-art review on scheduling with learning effects," *European Journal of Operational Research*, vol. 188, no. 2, pp. 315-329, 2008.

[45] E. Keachie and R. Fontana, "Effects of learning on optimal lot size," *Management Science*, vol. 13, no. 2, pp. 102-108, 1966.

[46] D. Biskup, "Single-machine scheduling with learning considerations," *European Journal of Operational Research*, vol. 115, no. 1, pp. 173-178, 1999.

[47] Y. Shi and R. Eberhart, "A modified particle swarm optimizer," Anchorage, AK, USA, 1998.

[48] M. Clerc and J. Kennedy, "The particle swarm - explosion, stability, and convergence in a multidimensional complex space," *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 1, pp. 58 - 73, 2002.

[49] M. Lovbjerg and T. Krink, "Extending particle swarm optimisers with self-organized criticality," Honolulu, HI, USA, 2002.

[50] D. Ben-Arieh and O. Maimon, "Annealing method for PCB assembly scheduling on two sequential machines," *International Journal of Computer Integrated Manufacturing*, vol. 5, no. 6, pp. 361-367, 1992.

[51] J. E. Beasley, "OR-Library: distributing test problems by electronic mail," *Journal of the Operational Research Society*, vol. 41, no. 11, pp. 1069-1072, 1990.