Hindawi
Complexity
Volume 2021, Article ID 6627081, 14 pages
https://doi.org/10.1155/2021/6627081

WILEY | Hindawi

Research Article

Two-Agent Single Machine Order Acceptance Scheduling Problem to Maximize Net Revenue

Jiaji Li,[1] **Yuvraj Gajpal**,[1] **Amit Kumar Bhardwaj**,[2] **Huangen Chen**,[3] **and Yuanyuan Liu**[4]

[1]*Department of Supply Chain Management, I.H. Asper School of Business, University of Manitoba, Winnipeg, MB R3T 5V4, Canada*
[2]*L.M Thapar School of Management, Thapar Institute of Engineering & Technology, Dera Bassi Campus, Patiala, Punjab, India*
[3]*School of Business Administration, Southwestern University of Finance and Economics, Chengdu, China*
[4]*School of International Business, Southwestern University of Finance and Economics, Chengdu, China*

Correspondence should be addressed to Yuanyuan Liu; liuyuanyuan@swufe.edu.cn

Received 1 November 2020; Revised 17 January 2021; Accepted 20 January 2021; Published 8 February 2021

Academic Editor: Shangce Gao

The paper considers two-agent order acceptance scheduling problems with different scheduling criteria. Two agents have a set of jobs to be processed by a single machine. The processing time and due date of each job are known in advance. In the order accepting scheduling problem, jobs are allowed to be rejected. The objective of the problem is to maximize the net revenue while keeping the weighted number of tardy jobs for the second agent within a predetermined value. A mixed-integer linear programming (MILP) formulation is provided to obtain the optimal solution. The problem is considered as an *NP-hard* problem. Therefore, MILP can be used to solve small problem instances optimally. To solve the problem instances with realistic size, heuristic and metaheuristic algorithms have been proposed. A heuristic method is used to determine and secure a quick solution while the metaheuristic based on particle swarm optimization (PSO) is designed to obtain the near-optimal solution. A numerical experiment is piloted and conducted on the benchmark instances that could be obtained from the literature. The performances of the proposed algorithms are tested through numerical experiments. The proposed PSO can obtain the solution within 0.1% of the optimal solution for problem instances up to 60 jobs. The performance of the proposed PSO is found to be significantly better than the performance of the heuristic.

1. Introduction

Order acceptance scheduling has been studied by researchers for the last few decades. In many cases, the number of orders accepted by a firm is not necessarily positively related to its profit, especially, when the capacity is bounded. Firms need to reject some of the orders to maximize their profit when the associated costs for some of the orders exceed their revenues. Accepting orders without a sensible and logical consideration might have a directly proportionate impact on the increase in workload. When the workload is excessively heavy, compared to the capacity of the firm, it might incur late deliveries of orders, therefore decreasing the level of customer satisfaction or even losing important customers. Hence, the study having order acceptance as its core focus has grabbed the attention of many researchers.

In the field of scheduling, researchers considered order acceptance in the last few decades. In the scheduling problem, decision-makers need to decide which jobs to be processed and what will be the sequence of the accepted jobs (Li et al. [1]). There is a trade-off between the revenues and penalty function while deciding acceptance or rejection of the jobs. There are always some penalty functions available,

related to the due dates, such as lateness and tardiness, which are worth looking into, depending on the scenario requirement. The order acceptance scheduling problem aims to maximize the profit gained by the revenues of accepted orders minus the specific penalty function.

So far, many researchers have tried to apply the study to the order acceptance problems, testing its usefulness in multiagent settings. In the multiagent scheduling problem, the jobs of two agents with conflicting objectives are processed by common resources (Gajpal and Li [2]; Li et al. [3]). The objectives of the order acceptance scheduling problem and multiagent scheduling problem are similar. Both refer to the construction of the settings for more explicitly specified conditions. The paper of Reisi-Nafchi and Moslehi [4] depicts a more practical application portraying the combination of the order acceptance with multiagent scheduling problems. In their research, they formulated a mathematical model for their problem and proposed a hybrid genetic algorithm to solve this problem.

Inspired by the work of Reisi-Nafchi and Moslehi [4], this paper extends their study, highlighting the following three perspectives: (1) propose a new variant of the two-agent order acceptance scheduling problem; (2) propose a mathematical model for the new variant of the problem; and (3) design algorithms that can be used to solve both versions of the problem. This paper focuses on two problems named as lateness penalty problem (*LPP*) and the tardiness penalty problem (*TPP*). The *LPP* problem is proposed by Reisi-Nafchi and Moslehi [4]; however, the *TPP* problem has been introduced in this paper, for the very first time. Both problems are the two-agent single machine order acceptance scheduling problems but their objective functions are different. The objective function of both problems considers how to maximize the net revenue of the first agent, subject to an upper bound on the weighted number of tardy jobs for the second agent. The net revenue is calculated as revenue from the accepted jobs subtracted by the penalty function. Penalty functions take the punishment by decreasing the objective function value when job processes after their due dates. The penalty value of the *LPP* problem is the weighted lateness of the first agent while the penalty value of the *TPP* problem is the weighted tardiness of the first agent.

To solve these problems, one metaheuristic and one heuristic algorithm have been introduced. A mathematical model based on MILP formulation is developed to find the optimal solution for the problem. The MILP formulation can solve only small size problem instances because the CPU time for solving MILP formulation increases exponentially with the size of the problem. To solve the problem instances with realistic size, heuristic and metaheuristic algorithms have been proposed.

The scheduling problem considered in this paper could be applied to the following scenario. A restaurant provides service to two kinds of customers: (1) customers who dine in the restaurant and (2) customers who order take-out services. Jobs in this scheduling problem are the dishes to be served to the customers. The two types of customers are treated as two agents. During the rush hours, the restaurant could not accept all the orders. Thus, it needs to reject some of the orders based on the profitability of the orders. The objective of the restaurant is to maximize the net revenue coming from all the orders while the weighted number of tardy jobs from the take-out customers could not exceed the upper bound. Also, the problem considered in this paper has many practical applications in the manufacturing industry as well as in the service industry. One of the applications that fall under this paper's criteria is the cloud-computing environment (Bhardwaj et al. [5]). In the cloud-computing environment, the service provider accepts the order of processing jobs from different users along with his own jobs. The service provider can be considered as an agent A and all other users combined can be considered as an agent B. The objective of the cloud-computing service provider is to maximize the net revenue from all the accepted jobs, while the weighted number of tardy jobs from all other users is kept within a predefined limit set by the management.

The remainder of the paper is organized as follows. Related work is reviewed in Section 2. A detailed description of the problems and the problem formulation is presented in Sections 3 and 4. The proposed algorithms are described in Section 5. Numerical results performed in Section 6 are used for comparing the effectiveness of the proposed algorithms. Eventually, Section 7 presents a summary of this paper.

2. Literature Review

Guerrero and Kern [6] are the pioneers in the study of order acceptance problems. They pointed out the importance of selecting a proper number of orders instead of accepting orders without any sensible cogitation. Pourbabai [7] first developed a mathematical model to help the manufacturers select the orders among all the orders. Slotnick and Morton [8] worked on an order acceptance problem in a single machine environment, of which the objective is the revenues minus weighted lateness penalties. They proposed an exact algorithm (branch and bound) and two heuristics to solve the problem. Ghosh [9] extended the work of Slotnick and Morton [8] by providing the NP-hardness proof of the problem. Also, they provided two pseudopolynomial algorithms to solve the problem, along with a fully polynomial-time approximation scheme. Lewis and Slotnick [10] worked on a multiperiod order acceptance scheduling problem in which decisions on order acceptance are made through different periods. They assume that job rejection leads to the loss of customers, and the profitability of decisions is scrutinized and assessed over each period. They proposed a dynamic programming algorithm and some heuristics to solve the problem. Slotnick and Morton [11] considered another order acceptance scheduling problem by replacing the penalty function with weighted tardiness. Rom and Slotnick [12] applied a genetic algorithm to solve the order acceptance scheduling problem with tardiness revenues.

Many researchers added variants to the order acceptance scheduling problem. Charnsirisakskul et al. [13] studied a single machine order acceptance problem with job preemption. Oğuz et al. [14] considered sequence-dependent setup times in their problem and arbitrarily assigned deadlines to the orders. Zhong et al. [15] added the variant of

machine availability to order acceptance scheduling problems, in which jobs could only be processed during specific time intervals due to the availability of the machines. When the workload is heavier than the capacity, manufacturers should decide either to reject or to outsource some of the orders. Xiao et al. [16] worked on an order acceptance scheduling problem in a permutation flow shop environment. They formulated their problem into an integer-programming problem and applied simulated annealing based on partial optimization to solve the problem. Lei and Guo [17] considered an order acceptance problem in a workshop environment. They contrived, putting together this particular problem and blending it into a mixed-integer programming problem, and applied the parallel neighborhood search algorithm to it.

There are some studies on order acceptance scheduling with bicriteria objectives. Lei and Guo [17] considered the objective of maximizing total net revenue and minimization of makespan simultaneously. Ou and Zhong [18] worked on a bicriteria order acceptance scheduling problem, with upper bounds on the number of the rejected job as well as the total processing time of accepted jobs. Noroozi et al. [19] considered order accepting criteria in a third-party logistic distribution system. They analyzed the trade-off among accepted orders revenue, delivery cost, and tardiness penalty in integrated production-distribution to maximize benefit. Silva et al. [20] considered an order acceptance scheduling problem with sequence-dependent setup time to maximize profit. They solved the problem using the Lagrangian relaxation and column generation-based branch and bound method. Sarvestani et al. [21] addressed an integrated order acceptance, supplier selection, and scheduling problem to maximize profit. They used genetic algorithm (GA) and variable neighborhood search (VNS) to solve the problem. Li and Ventura [22] considered a single-agent single machine scheduling problem with order acceptance criteria to maximum profit. The profit function considers the revenue minus the tardiness penalty. Recently, order acceptance criteria are combined with different settings/environments such as parallel machine setting (Wang and Ye [23]; Wu et. al [24]; Palakiti et al. [25]); carbon reduction environment (Chen et al. [26]); cluster supply chain environment (Li et al. [27]); production system (Wang et al. [28]; Yavari et al. [29]); and dynamic environment (Li et al. [30]; Melchiors et al. [31]). The increasing number of research articles in the order accepting criteria under different settings shows the importance and popularity of order acceptance scheduling problems.

The article by Reisi-Nafchi and Moslehi [4] is the closely related model considered in this paper. They integrated the two-agent scheduling situation into the order acceptance scheduling problem to maximize the net revenues from the first agent, subject to an upper bound of a weighted number of tardy jobs from the second agent. The net revenue is considered as a revenue of accepted orders with a penalty of total weighted lateness of accepted jobs. They provided an integer-programming formulation and hybrid genetic algorithm to solve the problem. This paper extends the work of Reisi-Nafchi and Moslehi [4] by introducing a new variant of the order acceptance criteria in the "two agents scheduling" setting. The literature review also stipulates the common trend of using metaheuristics to solve the scheduling problem. This observation has motivated us to develop and incorporate a metaheuristic algorithm to solve our problem.

3. Problem Definition and Notations

This paper considers a single machine scheduling problem with two agents, namely, agent A and agent B. Each of the agents has a job set n^x to be processed in a single machine. The processing time p_i, weight w_i, due date D_i , and revenue R_i associated with the job i are known. All the jobs are available at time zero (i.e., the release time is zero). A single machine processes all the jobs and at a time only one job is processed. Once a machine has started processing a job, it will continue running on that job until the job is completed. The problem seeks to determine the sequence of jobs σ. For a given sequence σ, the complete time $C_i(\sigma)$ of accepted job i can be calculated by the summation of processing time of all the jobs (including job i) sequenced before job i. The lateness value L_i of the job is defined as a difference between completion time and due date (i.e., $L_i = C_i(\sigma) - D_i$). The lateness value is positive if the job is completed after its due date. On the other hand, the lateness value is negative if the job is completed before the due date. Slotnick and Morton [8] introduced the lateness penalty to capture the penalty and reward in one function. Finishing a job late incurs a penalty in terms of loss of goodwill while finishing the job early incurs an appreciation from the customer. However, in many situations, only late completion incurs a penalty and the early completion does not bring any appreciation. This situation is captured in a tardiness value which only measures the delay in the completion time of a job (i.e., $T_i = \text{Max}\{0, C_i(\sigma) - D_i\}$). The existing literature studies the order acceptance scheduling problem in which lateness penalizes revenue obtained from the accepted jobs. This paper introduces a new order acceptance scheduling problem variant in which tardiness penalizes revenue received from the accepted jobs. The summary of notations used in this paper is summarized as follows:

(i) X: set of agents $X = \{A, B\}$

(ii) n_x: the number of jobs for agent $x \in X$

(iii) n: total number of jobs, $n = n_A + n_B$

(iv) N: set of all n jobs, $N = \{1, 2, \ldots, n\}$

(v) N_A: set of A jobs, consisting of n_A jobs, $N_A = \{1, 2, \ldots, n_A\}$

(vi) N_B: set of B jobs, consisting of n_B jobs, $N_B = \{n_A + 1, n_A + 2, \ldots, n\}$

(vii) p_i: the processing time of job i

(viii) R_i: revenue of job i

(ix) w_i: weight of job i

(x) D_i: due date of job i

(xi) σ: ordered set of jobs already scheduled

(xii) $C_i(\sigma)$: completion time of accepted job i under sequence σ.

(xiii) $f^x(\sigma)$: objective function of agent $x \in X$ under sequence σ.

(xiv) Y_i: binary decision variable taking value 1 if job i is accepted

(xv) U_i: binary decision variable taking value 1 if job i is tardy

(xvi) Z_{ij}: binary decision variable taking value 1 if job i precedes job j

(xvii) T_i: decision variable representing tardiness of the job i

(xviii) L_i: decision variable representing lateness of the job i

(xix) (xx) Q: the upper bound, a constant numberbig M: a large number

This paper showcases an attempt to maximize the net revenue subject to an upper bound of agent B. We consider two versions of the order acceptance two-agent scheduling problems, the *LPP* problem and the *TPP* problem. The objective of the *LPP* problem is to maximize the net revenue from all the accepted orders subtracted by the total weighted lateness from the accepted jobs of agent A, subject to an upper bound of the total weighted number of tardy jobs from agent B. In the scheduling terminology, this problem can be represented as $1|OA, \sum_{i\in N_B} w_i U_i Y_i \le Q|\sum_{i\in N} R_i Y_i - \sum_{i\in N_A} w_i L_i Y_i$. The term OA is used for order acceptance. The objective of the *TPP* problem is to maximize the net revenue from all the accepted orders subtracted by the total weighted tardiness from the accepted jobs of agent A, subject to an upper bound of the total weighted number of tardy jobs from agent B. This problem can be represented as $1|OA, \sum_{i\in N_B} w_i U_i Y_i \le Q|\sum_{i\in N} R_i Y_i - \sum_{i\in N_A} w_i T_i Y_i$.

The computational complexity of the *LPP* problem is proved to be NP-hard by Reisi-Nafchi and Moslehi [4]. They showed that the *LPP* problem is a special case $1|\sum_{i\in N_B} w_i U_i \le Q|\sum_{i\in N_A} w_i L_i$; therefore, the *LPP* problem is also an NP-hard problem. The *TPP* problem reduces to $1|\sum_{i\in N_B} w_i T_i$ when Q is set to infinite. Since $1|\sum_{i\in N_B} w_i T_i$ is *NP-hard*, the *TPP* problem is also an NP-hard problem.

4. Problem Formulation

Reisi-Nafchi and Moslehi [4] formulated the *LPP* problem into an integer-programming mathematical model. Based on their formulation, we have brought forward an evolved version of the paper, which is developed as an integer-programming model for the *TPP* problem.

When formulating the mathematical model for the *TPP* problem, a new binary variable Z_{ij} is introduced, which denotes the relative position between jobs i and j. The binary variable Z_{ij} takes the value of 1 if job i precedes job j, otherwise zero. It is significant to note here that the precedence here is not the immediate precedence (i.e., Z_{ij} takes the value of 1; this does not mean that job i and job j are adjacent to each other). Using the above notations, the *TPP* problem can be formulated as follows:

$$\text{Max} \sum_{i\in N} R_i Y_i - \sum_{i\in N_A} w_i T_i, \tag{1}$$

$$\text{subject to } Y_i + Y_j \le 1 + Z_{ij} + Z_{ji}, \quad i, j \in N,\ i < j, \tag{2}$$

$$Y_i + Y_j \ge 2\left(Z_{ij} + Z_{ji}\right), \quad i, j \in N,\ i < j, \tag{3}$$

$$Z_{ik} + Z_{kj} + Z_{jk} \le 2, \quad i, j, k \in N,\ i \ne j,\ j \ne k,\ i \ne k, \tag{4}$$

$$T_i \ge \sum_{j\in N} p_j Z_{ji} + p_i Y_i - D_i, \quad i \in N_A, \tag{5}$$

$$T_i \ge 0, \quad i \in N_A, \tag{6}$$

$$\sum_{j\in N} p_j Z_{ji} + p_i Y_i \le D_i Y_i + \text{Big}\, M \times U_i, \quad i \in N_B, \tag{7}$$

$$\sum_{j\in N_B} w_j U_j \le Q, \tag{8}$$

$$U_i \in \{0, 1\}, \quad i \in N_B, \tag{9}$$

$$Y_i \in \{0, 1\}, \quad i \in N, \tag{10}$$

$$Z_{ij} \in \{0, 1\}, \quad i, j \in N. \tag{11}$$

Equation (1) represents the objective function of the *TPP* problem, which is the total revenue of all the accepted orders penalized by the weighted tardiness of agent A orders. Constraints (2)–(4) determines the relative positions of jobs. Constraint (2) requires that a job i must precede another job j if both jobs are accepted. Constraint (3) sets the value of Z_{ij} or Z_{ji} to zero when job i or j is rejected, preventing the situation that Z_{ij} and Z_{ji} both are equal to zero. Constraint (4) ensures that there are no job loops in the sequence. Job loop represents a situation in which job i precedes job j, job j precedes job k, and job k precedes job i, which is logically wrong. Constraints (5) and (6) determine the tardiness value for job i. Constraint (7) determines the value of U_i. Constraint (8) ensures that the upper bound of the weighted number of tardy jobs from agent B is not exceeding. Constraints (9)–(11) impose the binary values for binary decision variables U_i, Y_i, and Z_{ij}.

Some of the optimal solution property of the *LPP* problem also holds true for *TPP* problem. The *TPP* problem satisfies the following three lemmas for which proof can be found in the work of Reisi-Nafchi and Moslehi [4].

Lemma 1. *There exists an optimal sequence, in which the agent B tardy accepted orders are sequenced arbitrarily at the end of the sequence.*

Lemma 2. *There exists an optimal sequence, in which the global arrangement of the agent B nontardy accepted orders follows the earliest due date (EDD) order and they are placed at the last possible positions before their due dates.*

Lemma 3. *There exists an optimal sequence, in which the adjacent orders of agent A follow the weighted shortest processing time (WSPT) order.*

5. Proposed Algorithms

In this section, algorithms including a heuristic and PSO are the proposed approaches to resolve the *LPP* and *TPP* problems.

5.1. Heuristic Algorithm. The proposed heuristic is based on the *WSPT* and *EDD* rule. In the WSPT rule, jobs are sequenced in nondecreasing order of the p_i/w_i ratio. In the EDD rule, jobs are sequenced in nondecreasing order of their due dates D_i. The WSPT rule minimizes the total weighted completion time while the EDD rule minimizes the total tardiness objective for a single machine and a single-agent setting.

The heuristic first creates partial schedules σ_A and σ_B for A and B types of jobs according to WSPT and EDD rules, respectively. The sequence σ_A is modified by removing unprofitable jobs. The sequence σ_B is divided into two subsequences σ_B^E and σ_B^L for an early and late set of the jobs. The final sequence σ is created by first sequencing σ_A , then σ_B^E , and finally σ_B^L. If the resultant sequence σ is infeasible, then the jobs from the sequence σ are removed one by one to make the solution feasible. The pseudocode of the proposed heuristic algorithm is presented as follows (Algorithm 1).

The computational complexity of the proposed algorithm can be described as follows. The initial solution generation has four separate subprocesses: (1) sequence of A jobs with complexity $O(n_1^2)$; (2) sequence of B jobs with complexity $O(n_2^2)$; (3) split of the sequence of B jobs with complexity $O(n_2)$; and (4) resequencing of late B jobs with complexity $O(n_2^2)$. Thus, the overall complexity of the initial solution process is $O(n_1^2) + O(n_2^2) + O(n_2) + O(n_2^2) = O(n_1^2) + O(n_2^2)$. The feasibility phase requires the maximum removal of n jobs and after each removal, it requires objective function calculation which can be performed in $O(n)$ time. Thus, the feasibility phase can be performed in $O(n^2)$ time. Therefore, the overall complexity of the heuristic algorithm is $O(n_1^2) + O(n_2^2) + O(n^2) = O(n^2)$.

5.2. Particle Swarm Optimization. Particle swarm optimization (PSO) is a metaheuristic technique for solving NP-hard problems. PSO is derived from the social behavior of fish and birds. In this algorithm, individual solutions are updated by the other members of the group. Although PSO applies to many scheduling problems, there are some challenges while applying it to solve a specific problem. The main challenge for this algorithm is to design a solution representation mechanism [32, 33] for the particular problem. In other words, this refers to the mechanism, which establishes the rules to interpret the data of particles into an actual schedule. Additional efforts are required to make it pertinent to the order acceptance scheduling problems because two subproblems are residing within it. While the first subproblem determines the set of accepted jobs, the objective of the second subproblem is to find the sequence for the accepted jobs. Therefore, a PSO algorithm containing parallel parameters of position values and velocities has been proposed. It is difficult for the classical PSO with only one group of parameters to solve the problems in this paper; two groups of position values and velocities are embedded in the algorithm. The first group focuses on the selection of accepted jobs, and the second group focuses on finding the sequence of the accepted jobs. Each particle in PSO has n dimensions representing n jobs. Let X_j^1 and X_j^2 represent the position value vector related to the order acceptance decision making and the job sequencing, respectively. The following are the notations that we have employed to describe the proposed PSO algorithm:

(i) m: number of particles

(ii) k: iterations index

(iii) K: maximum number of iterations

(iv) x_{ij}^1: position value of i^{th} particle in j^{th} dimension for order acceptance criteria

(v) X_i^1: position vector of i^{th} particle for order acceptance criteria

(vi) v_{ij}^1: velocity value of i^{th} particle in j^{th} dimension for order acceptance criteria

(vii) V_i^1: velocity vector of i^{th} particle for order acceptance criteria

(viii) x_{ij}^2: position value of i^{th} particle in j^{th} dimension for job sequencing criteria

(ix) X_i^2: position vector of i^{th} particle for job sequencing criteria

(x) v_{ij}^2: velocity value of i^{th} particle in j^{th} dimension for job sequencing criteria

(xi) V_i^2: velocity vector of i^{th} particle for job sequencing criteria

(xii) (xiii) pb_{ij}: personal best position value of i^{th} particle in j^{th} dimension for job sequencing criteriagb$_j$: global best position value in j^{th} dimension for job sequencing criteria

(xiv) S_i^{PB}: personal best solution vector for i^{th} particle

(xv) S^G: global best solution

(xvi) w: inertia weight

(xvii) c_1, c_2: social cognitive numbers

The pseudocode of the proposed PSO algorithm is as follows (Algorithm 2).

5.2.1. Parameter Initialization. At the beginning, parameters are generated using the following equations:

$$\begin{aligned} x_{ij}^1 &= X_{\min}^1 + \left(X_{\max}^1 - X_{\min}^1\right) \times U(0,1), \\ v_{ij}^1 &= V_{\min}^1 + \left(V_{\max}^1 - V_{\min}^1\right) \times U(0,1), \\ x_{ij}^2 &= X_{\min}^2 + \left(X_{\max}^2 - X_{\min}^2\right) \times U(0,1), \\ v_{ij}^2 &= V_{\min}^2 + \left(V_{\max}^2 - V_{\min}^2\right) \times U(0,1), \end{aligned} \tag{12}$$

where $X_{\max}^1, V_{\max}^1, X_{\max}^2, V_{\max}^2$ are equal to 4; $X_{\min}^1, V_{\min}^1, X_{\min}^2, V_{\min}^2$ are equal to -4. $U(0,1)$ is a random number within the range of [0, 1]. Therefore, all these parameters remain within the range of $[-4, 4]$ and the same is maintained throughout the algorithm.

5.2.2. Sequence Generator. As mentioned above, the sequence generating process in the proposed PSO consists of two steps. The first step is to decide and govern the accepted jobs and the second step decides the sequence of the accepted jobs. Job j in particle i is accepted when x_{ij}^1 is greater than 0; otherwise, it is rejected. Jobs are sequenced according to the minimum position value (MPV) rule, which specifies that the jobs are ordered in nondecreasing order of their position values x_{ij}^2.

5.2.3. Updating Parameters. Parameters are updated at the end of each iteration. Before updating the parameter, the personal best solution S_i^{pb} and the personal best position value pb_{ij} for each particle are updated. If the new solution σ_i is found to be better than S_i^{pb}, then the personal best solution S_i^{pb} is replaced by the current solution $\sigma_i$and the personal best position value pb_{ij} is replaced by the position value x_{ij}^2. Similarly, the global best solution and the global best position values are updated. Two different ways are applied for updating job sequencing and order accepting criteria. The order acceptance parameters as updated as follows:

$$\begin{aligned} v_{ij}^1 &= w \times v_{ij}^1 + c_1 \times r_1 \times z_1 + c_2 \times r_2 \times z_2, \\ x_{ij}^1 &= x_{ij}^1 + v_{ij}^1. \end{aligned} \tag{13}$$

Parameters of z_1 and z_2 are variables representing the status of job j. Variable z_1 takes the value of 1 when job j in i^{th} particle is accepted in personal best solution; otherwise, it takes the value of -1. Likewise, the value of the variable z_2 is determined by whether the job is accepted in the global best solution or not. Parameters of c_1 and c_2 are the social coefficients quantifying the extent to which an individual relies on its experiment (i.e., to its own personal best or to the global best). Inertia w is the resistance of any physical object to any change in the state of motion. It controls the velocity and direction of all the particles. Variables of r_1 and r_2 are random numbers within the range of [0, 1]. In this paper, we set $c_1 = 0.9, c_2 = 0.9, w = 0.9$. The job sequencing parameters are updated as follows:

$$\begin{aligned} v_{ij}^2 &= w \times v_{ij}^2 + c_1 \times r_1 \times \left(\text{pb}_{ij} - x_{ij}^2\right) + c_2 \times r_2 \times \left(\text{gb}_{ij} - x_{ij}^2\right), \\ x_{ij}^2 &= x_{ij}^2 + v_{ij}^2. \end{aligned} \tag{14}$$

5.2.4. Local Search. There are two types of local search schemes proposed to improve the solution of each particle. These two local search schemes are named as an *accepted job insertion local search (AJILS)* and a *rejected job insertion local search (RJILS)*. The *AJILS* tries to improve the solution by repositioning the accepted jobs to a different position. At first, the job is removed from its original position, and then it was reinserted in all the other positions. The best-inserted position is selected for the insertion of the job and the resultant solution is used for further evaluation. All the accepted jobs are selected one by one for possible solution improvement. The *RJILS* attempts to insert the rejected jobs into the solution. The rejected jobs are inserted in the best feasible position if it improves the objective function value. All the rejected jobs are singled out one by one for possible solution improvement.

5.2.5. Normalization of Parameters. The absolute values of parameters tend to be very large after many iterations, enabling the solutions to get stuck in the local optimum. Thus, the parameters are normalized after each iteration. The parameters exceeding the upper limit are forced to take a value of 4, and the parameters that are less than the lower limit are forced to take the value of -4.

5.2.6. Computational Complexity of the PSO. The three main components of the PSO algorithm are (1) sequence generation step, (2) local search method, and (3) parameter updating step. In the sequence generation step, determining the set of accepted and rejected jobs can be performed in $O(n)$ time while the sequencing operation can be performed in $O(n^2)$ time. Thus, the overall complexity of the sequence generation step is $O(n^2)$. The local search uses the two schemes with the complexity of (n^2) for each scheme. Thus, the overall complexity of the local search scheme is $O(n^2)$. The parameter updating can be performed in $O(n)$ time. Thus, the complexity of one iteration of PSO is $O(n^2) + O(n^2) + O(n) = O(n^2)$. The PSO performs K iterations, which brings the overall complexity of the PSO to be $O(Kn^2)$.

6. Numerical Results

This section evaluates the performance of the proposed algorithms, assessing their effectiveness and application utility on two problems. Numerical analysis is provided based on the solution and CPU time of the proposed algorithms. The proposed algorithms are coded in the C++ programming language. The numerical experiments are

```
(1) /* Initial Solution */
(2) Create a sequence σ_A for A jobs according to the WSPT rule
(3) for k = 1 to n_1 do
(4)    Select job i = σ_A(k)
(5)    if R_i < w_i × (C_i(σ) − D_i) then
(6)       Remove job i from sequence σ_A
(7)    end if
(8) end for
(9) Create a sequence σ_B for B jobs according to the EDD rule
(10) Create null sequences σ_B^E = {φ} and σ_B^L = {φ}
(11) for k = 1 to n_2 do
(12)    Select job i = σ_B(k)
(13)    if C_i(σ_A) + p_i <= D_i then
(14)       append job i in sequence σ_B^E
(15)    else
(16)       append job i in sequence σ_B^L
(17)    end if
(18) end for
(19) Update σ_B^L by arranging jobs in nondecreasing order of R_i/w_i ratio
(20) Create initial sequence σ = {σ_A ∪ σ_B^E ∪ σ_B^L}
(21) /* Feasibility phase */
(22)    U(σ) ⟵ f^B(σ) //Calculate the weighted number of tardy jobs for σ
(23) for k = 1 to n do
(24)    if U(σ) ≤ Q then
(25)       break k loop
(26) end if
(27)    σ ⟵ Update sequence by removing first job
(28)    U(σ) ⟵ f^B(σ) //Calculate the weighted number of tardy jobs for σ
(29) end for
(30) return solution σ
```

Algorithm 1: Heuristic algorithm.

performed in a computer environment with AMD Opteron 2.3 GHz with 16 GB of RAM on Unix OS. The linear-programming-based mathematical model proposed for the *TPP* problem is solved by AMPL software with CPLEX solver, in the computer system of iMac desktop with 3.3 GHz 8 GB RAM.

There are three sections in this section. The first section explains the settings of the instances used in this paper. The second section aims to test the robustness of proposed algorithms by comparing them with the existing solution of Reisi-Nafchi and Moslehi [4]. The third section evaluates the proposed algorithms on the *TPP* problem. The following are notations used to report and evaluate the numerical results:

(i) *CPLEX*: value of objective function generated by the mathematical model

(ii) *ABS*: absolute value of objective function generated by algorithms

(iii) ABS_{best}: *ABS* value generated by the algorithm with the best performance

(iv) *APD*: absolute percentage of deviation, APD = ((ABS − CPLEX)/CPLEX ∗ 100%)

(v) *RPD*: relative percentage of deviation, RPD = $((ABS - ABS_{best})/ABS_{best} * 100\%)$

(vi) *CPU*: running time of the proposed algorithm

Additionally, *Avg*, *Min*, and *Max* are used for representing average, minimal, maximal values of corresponding results, respectively. We also performed a paired t-test to compare the performance of the proposed algorithms at the 95% significance level. In the paired t-test, the population mean of the two methods is compared. Let $\mu_D = \mu_1 - \mu_2$ denote the true average difference between the average solution values of the comparing algorithms. Here, μ_1 and μ_2 denote the population mean of the comparing algorithms, respectively. The null hypothesis is given by H_0: $\mu_D = 0$, which states that there is no difference between the average solution value of the algorithms. The alternative hypothesis is given by H_1: $\mu_D > 0$ stating that the average solution of the first algorithm is greater than the average solution of the second algorithm.

6.1. Instance Setting. In this paper, instances generated in the study of Reisi-Nafchi and Moslehi [4] are recognized as the benchmark instances. Different algorithms are evaluated and appraised based on their performance on these benchmark instances. The setting of the benchmark instances is as follows.

Processing times and weights are generated randomly from a uniform distribution within the interval of [1, 10]. When generating due dates, two additional parameters are

(1) **Initialization**
(2) Set $k = 0$, $m = 50$, $c_1 = 0.8$, $c_2 = 0.8$, $w = 0.9$
(3) Initialize $X_i^1, V_i^1, X_i^2, V_i^2$
(4) **Main phase**
(5) **do while** $(k \leq K)$
(6) $k = k + 1$
(7) **for** $i = 1$ **to m do**
(8) $\sigma_i \leftarrow$ Generate sequence $(X_i^1, \ldots, X_m^1, X_i^2, \ldots, X_m^2)$
(9) $\sigma_i \leftarrow$ Local search(σ_i)
(10) Update S^G and S_i^{PB}, $\forall i = 1, \ldots, m$
(11) Update $X_i^1, X_i^2, V_i^1, V_i^2$, $\forall i = 1, \ldots, m$
(12) **end for**
(13) **end do**
(14) **return solution** S^G

Algorithm 2: PSO algorithm.

introduced: tardiness factor τ and the range factor r. Due dates are generated randomly within the range of $[TPT^*(1 - \tau - (r/2)), TPT^*(1 - \tau - (r/2))]$, in which the TPT denotes the sum of processing times from all the jobs. The revenues are generated from a uniform distribution within the interval of $[1, 2 * p_i]$; upper bound Q is generated within the range of $[0, \sum_{i=n_1+1}^{n} w_i]$. The instances are sorted into eight groups according to the parameter setting, the sizes of instances ranging from 20 to 150. In the first four groups, $n_1 = 2n_2$; in the remaining groups, $2n_1 = n_2$. Parameter of τ takes the value of 0.3 in the groups of G01, G02, G05, and G06, and it takes the value of 0.7 in the groups of G03, G04, G07, and G08. The parameter of r takes the value of 0.2 in groups of G01, G03, G05, and G07. In the benchmark instances, there are 1280 instances in total and 20 instances for each parameter setting. The numerical analysis is primarily focused on the average results, which could be obtained from a varied set of results in different parameter settings.

6.2. Numerical Analysis for the LPP Problem. As mentioned above, the performance of PSO and heuristic is compared and weighed against the results in Reisi-Nafchi and Moslehi [4]. The numerical results of GA and *CPLEX* solution in Reisi-Nafchi and Moslehi [4] are obtained from the authors. In small instances, the performance of these algorithms along with the GA algorithm is compared with the *CPLEX* solution, while in large instances, the illustrations include the performance of PSO, heuristic, and GA being compared with each other. It is noteworthy that the CPU time of the heuristic is not shown in the corresponding tables since the proposed heuristic algorithm itself could solve all of the instances within 0.1 seconds.

Table 1 displays the computational results of integer-programming formulation. According to the results, it could be seen that the existing mathematical model could solve problem instances with up to 60 jobs. It could also be seen that the CPU time of G03 and G07 is significantly longer than other groups, when $\tau = 0.3$ and $r = 0.6$. The maximum CPU time in these groups exceeds 1000 seconds, indicating that it is more difficult for the system to search for the optimal solutions in these groups. The results further indicate that the CPU time of the CPLEX solver increases exponentially and thus the larger problem instances cannot be solved using the CPLEX solver. Therefore, heuristic and metaheuristic algorithms are used to solve bigger problem instances.

Table 2 reports the APD values of the three algorithms for the *LPP* problem. As shown in the table, the PSO algorithm performs best in small instances in terms of the solution quality. The solutions of PSO are only 0.01 percent away from the CPLEX solutions. Besides, the maximal APD of PSO is lower than 0.1%, which is lower than other algorithms. On the other hand, though the heuristic deviates furthest from CPLEX solutions, the average APD of the heuristic is lower than 1.5%. This observation also indicates the effectiveness of the heuristic algorithm in solving the small problem instances.

The paired t-test between heuristic and GA found the p value to be 0.00105. Since the p value is less than α (i.e., $0.00105 < 0.05$), the null hypothesis is rejected. The rejection of the null hypothesis state that the average value performance of the GA is statistically better than the performance of the heuristic algorithm at the 95% significance level. Similarly, the paired t-test between heuristic and PSO found the p value to be 0.0024, which indicates the better performance of PSO over heuristic on average solution value. The paired t-test between and GA found the p value to be 0.1955. This analysis indicates that on average results of GA and PSO are not significantly different statistically. However, the average RPD of PSO is slightly better than the average RPD of GA.

Table 3 shows the results of numerical experiments for the *LPP* problem. From Table 3, it could be seen that the PSO also outperforms the algorithms of GA and heuristic for the large instances. The *RPD* of PSO is lower than those of other algorithms with only a few exceptions. The average *RPD* of PSO in all groups is only 0.05, indicating the robustness of PSO for solving the *LPP* problem. Nonetheless, observation from the perspective of CPU time somehow provides a different picture. The average CPU time of PSO is 183.54

Table 1: CPU time of CPLEX solutions in the *LPP* problem.

Group	n	CPU		
		Avg	Min	Max
G01	20	0.09	0.02	0.98
	30	0.37	0.07	3.27
	40	1.5	0.18	7.53
	60	1.51	0.41	15.63
G02	20	0.03	0.02	0.15
	30	0.08	0.06	0.1
	40	0.22	0.17	0.3
	60	0.4	0.31	0.64
G03	20	0.39	0.02	1.18
	30	4.3	0.09	15.07
	40	28.78	0.24	121.75
	60	308.44	0.53	1314.29
G04	20	0.07	0.02	0.64
	30	0.34	0.07	1.46
	40	1	0.17	4.59
	60	11.7	0.48	138.64
G05	20	0.06	0.02	0.4
	30	1.45	0.14	12.28
	40	1.16	0.16	3.85
	60	27.34	0.76	201.91
G06	20	0.04	0.02	0.11
	30	0.4	0.13	1.93
	40	0.22	0.12	0.64
	60	1.77	0.58	8.1
G07	20	0.32	0.08	0.98
	30	4.51	0.55	16.78
	40	35.92	0.48	200.73
	60	519.62	23.01	1442.53
G08	20	0.06	0.02	0.48
	30	0.43	0.08	1.88
	40	0.83	0.17	3.21
	60	7.33	0.59	88.78

seconds, which is longer than the CPU time of GA. The CPU time of both metaheuristics increases significantly as the size of instances increases. Thus, when the size of problem instances becomes larger than the upper bound of 150, the performance of the heuristic algorithm might be more prominent and noteworthy, which is projected through its efficiency. Besides, it is noteworthy to point out that the average value and the maximal value of *RPD* for the heuristic algorithm are 1.21 percent and 4.61 percent, respectively, which also justify the use of heuristic for solving bigger problem instances.

The paired t-test among different algorithms for large problem instances shows a similar indication as to the small problem instances. The PSO and GA are statistically better than the heuristic algorithm. The performances of GA and PSO are not significantly different statistically. However, the average RPD of 0.05 for PSO is better than the average RPD of 0.28 for GA. Based on the discussion above, it could be concluded that both PSO and heuristic could provide reliable solutions. PSO provides better solutions than heuristic and competitive solutions as the GA algorithm, while the heuristic could solve the problem in a shorter CPU time.

6.3. Numerical Analysis for TPP Problem. Table 4 conveys the CPU time for the mathematical model of the *TPP* problem. The *TPP* problem maximizes the net revenue of agent A, subject to an upper bound of the total weighted number of tardy jobs from agent B. As shown in the table, the mathematical model formulation provided in this paper manifests how it could only solve instances with the size limit of 20 jobs, which might be owing to the computational complexity of the problem scenario. In addition to that, the CPU time consumed for the mathematical model is longer than other settings when the tardiness factor takes a value of 0.3 instead of 0.7. In G05, the average CPU time to get the CPLEX solutions is 2089.13 seconds. For one of the instances belonging to this group, it takes more than ten hours (38991.2 seconds) to find the solution.

Table 5 displays the results of PSO and the heuristic algorithm for the small instances. As shown in the table, PSO outperforms the heuristic in all groups in terms of the objective function value. It deviates from the CPLEX solutions only by 0.11 percent. Contrarily, the solution quality of the heuristic algorithm is not as good as PSO; the average of the APD values from the five groups (G03, G04, G06, G07, and G08) exceeds 10 percent. Although it appears that the heuristic in this paper does not perform well for the *TPP* problem, it is noteworthy to mention that the heuristic could still be used as a good comparison for PSO due to its efficiency. The paired t-test between heuristic and PSO for small problem instances found the p value to be 0.00338. The analysis indicates that the average performance of the PSO is statistically better than the average performance of the heuristic algorithm at the 95% significance level.

The numerical results of PSO and heuristic in the *TPP* problem are presented in Table 6 to showcase larger problem instances. The results indicate that the PSO still shows its superiority over the heuristic. It could be concluded that the PSO can provide a better solution for the problem considered in this paper. The heuristic can generate a solution in a shorter time with an acceptable percentage deviation. The paired t-test between heuristic and PSO for large problem instances found the p value to be 2.3×10^{-9} . The analysis strongly suggests that the average performance of the PSO is statistically better than the average performance of the heuristic algorithm at the 95% significance level.

The numerical analysis performed for *LPP* and *TPP* problem indicates the superior performance of PSO over existing algorithms. The superior performance can be attributed to the hybridization of PSO with local search schemes. The superior performance of the hybrid PSO algorithm indicates that the algorithm has the potential to solve different problems arising in different industries. In past, PSO

Table 2: Comparison of algorithms of *LPP* problem in small instances.

Group	n	GA			Heuristic		PSO		
		Average ABS	Average APD	Average CPU time	Average ABS	Average APD	Average ABS	Average APD	Average CPU time
G01	20	2518.30	0	0.94	2517.60	0.03	2518.35	0	0.56
	30	5682.55	0.02	1.95	5680.75	0.05	5683.30	0	1.96
	40	9663.80	0.01	3.9	9658.3	0.06	9664.60	0	5
	60	21319.3	0.01	8.09	21317	0.02	21321.25	0	17.75
G02	20	2460	0.01	1.01	2458.55	0.02	2460.25	0	0.6
	30	5166	0	1.66	5165.15	0	5166.10	0	2.12
	40	9784.05	0	3.54	9784.05	0	9784.40	0	5.38
	60	21093	0	9	21093.15	0	21093.15	0	19.13
G03	20	920.50	0.11	1.94	904.70	2.25	921.30	0.03	0.47
	30	1951.60	0.21	3.76	1940.90	0.7396	1954.40	0.06	1.89
	40	3386.15	0.33	7.01	3370.60	0.81	3394.65	0.05	4.58
	60	7149.85	0.34	15.82	7119.25	0.749	7168.10	0.07	16.25
G04	20	927.95	0.27	1.41	921.25	1.11	929.90	0.01	0.51
	30	2104.65	0.24	3.01	2067.50	2.019	2109.15	0.04	1.79
	40	3606.20	0.26	6.16	3555.45	1.81	3616.10	0.02	4.59
	60	8228.95	0.11	13.44	8108.20	1.56	8236.50	0.02	16.62
G05	20	3945.65	0.04	1.74	3943.60	0.09	3947.25	0	0.79
	30	9043.75	0.02	4.44	9034.70	0.13	9045.20	0	2.90
	40	16682.30	0.03	8.84	16683.30	0.03	16687.75	0	7.73
	60	35684.45	0.03	20.79	35681.90	0.03	35684.45	0	27.86
G06	20	4051.50	0.09	1.67	4035.35	0.44	4052.10	0	1.07
	30	9087.05	0.02	3.36	9045	0.49	9088.50	0	4.19
	40	15829.60	0	6.10	15762.45	0.44	15830.40	0	10.50
	60	36173.25	0	13.8	35997.25	0.5	36174.90	0	37.70
G07	20	1179.8	0.16	1.70	1145.80	3.20	1181.65	0	0.38
	30	2422.85	0.97	3.73	2335.45	4.52	2446.05	0.02	1.41
	40	5065.50	0.69	7.16	5096.70	2.53	4971.40	0.08	3.60
	60	10551.75	1.17	19.06	10276.80	3.77	10674.25	0.02	13.17
G08	20	1414.75	0.02	0.88	1361.90	4.16	1414.90	0	0.40
	30	3572.65	0.04	2.76	3422.10	4.65	3573.50	0	1.54
	40	6018.05	0.13	5.04	5747.70	4.62	6026	0	4.02
	60	13176.25	0.05	11.7	12704.15	3.66	13180.60	0.01	14.58
Average		8745.69	0.17	6.27	8685.52	1.39	8750.01	0.01	7.22

Table 3: Comparison of algorithms for *LPP* problem in large problem instances.

Group	n	GA			Heuristic		PSO		
		Average ABS	Average RPD	Average CPU time	Average ABS	Average RPD	Average ABS	Average RPD	Average CPU time
G01	80	39999.70	0	18.36	39996.55	0.01	40001.35	0	47.17
	100	60348.05	0	33.35	60344.20	0.01	60351	0	98.50
	120	90745.90	0	53.99	90746	0	90747.45	0	175.65
	150	136730.50	0	95.76	136730.25	0	136732.45	0	349.23
G02	80	40321.55	0	19.09	40321.80	0	40322.25	0	50.24
	100	59533.70	0	35.84	59533.25	0	59534.45	0	106.82
	120	88112.60	0	55.2	88112.85	0	88113.20	0	193.73
	150	135944.40	0	93.05	135945.20	0	135945.20	0	375.07
G03	80	13218.35	0.18	33.01	13169.90	0.57	13241.65	0	43.49
	100	21454.10	0.10	51.24	21390.30	0.41	21474.45	0	90.17
	120	29716.55	0.24	75.88	29621.80	0.56	29772.05	0.04	154.81
	150	45783.90	0.17	144.39	45646.40	0.47	45850.05	0.02	333.35

Table 3: Continued.

Group	n	GA			Heuristic		PSO		
		Average ABS	Average RPD	Average CPU time	Average ABS	Average RPD	Average ABS	Average RPD	Average CPU time
G04	80	16005.40	0.08	27.18	15764.05	1.63	16017	0	43.59
	100	23260.65	0.13	43.34	22756	2.39	23290.40	0	86.89
	120	33656.80	0.11	64.35	33128.60	1.77	33688.35	0.02	148.34
	150	55188.40	0.03	113.93	54435.75	1.40	55145.45	0.13	297.54
G05	80	65441.85	0.02	41.28	65444.25	0.02	65454.45	0	72.36
	100	105339.85	0.01	69.83	105337.10	0.01	105351.90	0	152.88
	120	144242	0.01	130.24	144235.25	0.01	144246.15	0	273.69
	150	224901.75	0.01	201.08	224900.70	0.01	224919.10	0	565.53
G06	80	67786.35	0	27.94	67596.85	0.30	67786.15	0	95.73
	100	102016.45	0	58.30	101679.35	0.33	102016.30	0	200.17
	120	142779.90	0	93.90	142188.55	0.41	142761.25	0.02	332.17
	150	229024.20	0	151.58	228191.40	0.36	228882.65	0.06	663.42
G07	80	18552.30	1.46	48.56	18146.20	3.59	18823.95	0	11
	100	31254.30	2.01	94.25	31044.10	2.69	31896.40	0	64.70
	120	44185.65	2.16	154.37	43879.85	2.88	45117.95	0.10	116.44
	150	66529.35	1.99	327.84	66258.80	2.39	67843.05	0.04	240.82
G08	80	23772	0.02	24.75	22833.25	4.07	23770	0.03	36.53
	100	36552.20	0.13	41.10	34912.65	4.67	36548.90	0.10	67.50
	120	55435.10	0	57.59	53400	3.77	55273.10	0.30	123.48
	150	85601.85	0	104.03	84987.10	3.98	82243.20	0.74	262.19
Average		72919.86	0.28	80.77	72583.70	1.21	72911.29	0.05	183.54

Table 4: CPU time of CPLEX solutions in *TPP* problem.

Group	n	CPU		
		Avg	Max	Min
G01	20	202.06	3001.29	0.29
G02	20	7.46	117.49	0.18
G03	20	75.50	428.41	0.20
G04	20	5.19	27.88	0.13
G05	20	2089.13	38991.2	0.19
G06	20	9.69	156.75	0.18
G07	20	75.60	448.80	2.80
G08	20	9.53	49.89	0.71

Table 5: Comparison of algorithms in *TPP* problem in small instances.

Group	n	Heuristic		PSO		
		Average ABS	Average APD	Average ABS	Average CPU time	Average APD
G01	20	116.80	0.68	117.60	0.30	0
G02	20	112.50	2.29	115.15	0.35	0
G03	20	98.30	14.21	112.85	0.53	0.14
G04	20	96.75	11.46	108.50	0.61	0.10
G05	20	110.45	6.02	117.45	0.46	0
G06	20	107.15	10.24	119.50	0.53	0
G07	20	71.30	23.57	92.60	0.35	0.23
G08	20	70.20	29.53	98.15	0.49	0.44
Average		97.93	12.25	110.23	0.45	0.11

Table 6: Comparison of algorithms in *TPP* problem in large instances.

n	Group	Heuristic		PSO		
		Average ABS	Average RPD	Average ABS	Average CPU time	Average RPD
30	G01	173.75	1.96	176.95	0.89	0
40		225.55	3.05	232.60	2.45	0
60		344.05	1.29	348.60	6.35	0
80		463.15	1.10	468.50	17.24	0
100		608.30	1.20	615.50	32.93	0
120		726.05	0.23	727.70	52.49	0
150		892.95	0.23	894.95	105.01	0
30	G02	169.90	0.61	170.85	0.90	0
40		239.55	0.15	239.90	2.20	0
60		363.25	0	363.25	6.94	0
80		485.35	0.09	485.80	17.87	0
100		601.40	0.19	602.60	40.77	0
120		723.30	0.05	723.65	64.81	0
150		903.70	0	903.70	113.56	0
30	G03	146.90	11.37	164.65	1.86	0
40		204.55	9.94	225.35	4.10	0
60		292.95	10.14	323.45	13.87	0
80		414.80	7.53	447.30	36.97	0
100		537.10	7.21	575.75	70.42	0
120		581.60	8.15	625.75	122.50	0
150		807.35	5.61	852.75	247.89	0
30	G04	146.40	14.27	169.85	2.41	0
40		200.45	9.95	221.75	5.64	0
60		308.90	9.59	341.15	18.73	0
80		419.95	8.50	458	53.93	0
100		549.25	8.75	601	114.19	0
120		633.95	10.49	705.30	203.20	0
150		753.05	11.06	843.50	371.27	0
30	G05	170.25	6.08	180.75	1.58	0
40		234.40	3.34	242.40	3.72	0
60		346.85	4.61	363.10	12.29	0
80		463.80	2.58	476.40	28.42	0
100		590.40	2.65	606.35	54.21	0
120		692.50	3.94	720.15	95.59	0.01
150		853	2.75	877.15	205.54	0
30	G06	171.55	7.68	185.90	1.72	0
40		211.50	7.34	227.20	4.01	0
60		343.10	8.22	373.85	13	0
80		431.15	6.29	460.10	26.34	0
100		570.90	5.90	606.85	55.61	0
120		647.70	8.14	704.20	109.97	0
150		822.70	6.22	877.15	198.15	0
30	G07	114.20	23.30	146.70	1.14	0
40		147.10	23.20	190.40	2.75	0
60		226.35	20.75	284.50	9.13	0
80		297.55	21.60	377.20	20.83	0
100		368.55	20.70	462.10	40.88	0
120		460.10	19.70	570	72.39	0
150		559.60	18.74	685.60	142.20	0
30	G08	131.35	23.49	171.15	1.96	0
40		158.10	26.43	214.60	4.81	0
60		243.80	24.31	321.50	18.36	0
80		326.85	23.24	424.55	50.35	0
100		396	24.61	525.35	101.70	0
120		493.05	25.71	662.35	181.65	0
150		588.45	25.46	788.05	374.11	0
Average		428.18	9.64	468.49	63.64	0

has been successfully used to solve the option price model (Sharma et al., [34]). In the future, the hybrid PSO can be applied to solve different pricing and index-based models such coupon bond model (Jin et al. [35]), option pricing model (Jin et al., [36]), and risk index model (Jin and Zhu, [37]).

7. Conclusions

This paper works on the order acceptance two-agent scheduling problems, which are regarded as an NP-hard problem. This paper contemplates and examines the two variants of the problem named the *LPP* problem and the *TPP* problem. The objective function of the *LPP* problem is to maximize the total revenues of accepted orders minus the total weighted lateness of the jobs from agent A. The objective function of the *TPP* problem is to maximize the total revenues of accepted orders minus the total weighted tardiness of jobs from agent A. Both variants of the problem impose a constraint that the sum of the weighted number of tardy jobs belonging to agent B is not allowed to exceed the given upper bound. Algorithms including a metaheuristic (PSO) and a heuristic are proposed to solve these problems. Additionally, a mathematical model is provided. The proposed algorithms are designed in such a way so that it can decode and find a solution to both problems that are being considered in this paper.

The numerical results stipulate that the proposed algorithm is competitive with the existing GA algorithm. The solution produced by the proposed PSO algorithm is within less than 1% of the optimal solution for the small problem instances. The PSO could provide better solutions while the heuristic could provide solutions with an acceptable deviation percentage from the PSO solution in shorter CPU time.

In the future, the mathematical model formulation could be improved and tweaked by removing the redundant constraints and decision variables to solve bigger problem instances. Also, experimental efforts could essentially be performed, focusing on the aspects like reducing CPU time of the proposed PSO algorithm.

Data Availability

Data used in this research work are available from Yuvraj Gajpal (yuvraj.gajpal@umanitoba.ca).

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This research was partially supported by NSERC, grant 318689.

References

[1] Q. Li, D. Zhang, S. Wang, and I. Kucukkoc, "A dynamic order acceptance and scheduling approach for additive manufacturing on-demand production," *The International Journal of Advanced Manufacturing Technology*, vol. 105, no. 9, pp. 3711–3729, 2019.

[2] Y. Gajpal and H. Li, "Single machine scheduling with two agents for total completion time objectives," in *Applied Operational Research: 8th International Conference, ICAOR 2016*ORLAB Analytics, Rotterdam, The Netherlands, 2016.

[3] H. Li, Y. Gajpal, and C. R. Bector, "Single machine scheduling with two-agent for total weighted completion time objectives," *Applied Soft Computing*, vol. 70, pp. 147–156, 2018.

[4] M. Reisi-Nafchi and G. Moslehi, "A hybrid genetic and linear programming algorithm for two-agent order acceptance and scheduling problem," *Applied Soft Computing*, vol. 33, pp. 37–47, 2015.

[5] A. K. Bhardwaj, Y. Gajpal, C. Surti, and S. S. Gill, "Heart: unrelated parallel machines problem with precedence constraints for task scheduling in cloud computing using heuristic and meta-heuristic algorithms," *Software: Practice and Experience*, vol. 50, no. 12, pp. 2231–2251, 2020.

[6] H. H. Guerrero and G. M. Kern, "How to more effectively accept and refuse orders," *Production and Inventory Management Journal*, vol. 29, no. 4, pp. 59–63, 1988.

[7] B. Pourbabai, "A short term production planning and scheduling model," *Engineering Costs and Production Economics*, vol. 18, no. 2, pp. 159–167, 1989.

[8] S. A. Slotnick and T. E. Morton, "Selecting jobs for a heavily loaded shop with lateness penalties," *Computers & Operations Research*, vol. 23, no. 2, pp. 131–140, 1996.

[9] J. B. Ghosh, "Job selection in a heavily loaded shop," *Computers & Operations Research*, vol. 24, no. 2, pp. 141–145, 1997.

[10] H. F. Lewis and S. A. Slotnick, "Multi-period job selection: planning work loads to maximize profit," *Computers & Operations Research*, vol. 29, no. 8, pp. 1081–1098, 2002.

[11] S. A. Slotnick and T. E. Morton, "Order acceptance with weighted tardiness," *Computers & Operations Research*, vol. 34, no. 10, pp. 3029–3042, 2007.

[12] W. O. Rom and S. A. Slotnick, "Order acceptance using genetic algorithms," *Computers & Operations Research*, vol. 36, no. 6, pp. 1758–1767, 2009.

[13] K. Charnsirisakskul, P. M. Griffin, and P. Keskinocak, "Order selection and scheduling with leadtime flexibility," *IIE Transactions*, vol. 36, no. 7, pp. 697–707, 2004.

[14] C. Oğuz, F. Sibel Salman, and Z. Bilgintürk Yalçin, "Order acceptance and scheduling decisions in make-to-order systems," *International Journal of Production Economics*, vol. 125, no. 1, pp. 200–211, 2010.

[15] X. Zhong, J. Ou, and G. Wang, "Order acceptance and scheduling with machine availability constraints," *European Journal of Operational Research*, vol. 232, no. 3, pp. 435–441, 2014.

[16] Y.-Y. Xiao, R.-Q. Zhang, Q.-H. Zhao, and I. Kaku, "Permutation flow shop scheduling with order acceptance and weighted tardiness," *Applied Mathematics and Computation*, vol. 218, no. 15, pp. 7911–7926, 2012.

[17] D. Lei and X. Guo, "A parallel neighborhood search for order acceptance and scheduling in flow shop environment," *International Journal of Production Economics*, vol. 165, pp. 12–18, 2015.

[18] J. Ou and X. Zhong, "Bicriteria order acceptance and scheduling with consideration of fill rate," *European Journal of Operational Research*, vol. 262, no. 3, pp. 904–907, 2017.

[19] A. Noroozi, M. M. Mazdeh, M. Heydari, and M. Rasti-Barzoki, "Coordinating order acceptance and integrated production-distribution scheduling with batch delivery

considering Third Party Logistics distribution," *Journal of Manufacturing Systems*, vol. 46, pp. 29–45, 2018.

[20] Y. L. T. V. Silva, A. Subramanian, and A. A. Pessoa, "Exact and heuristic algorithms for order acceptance and scheduling with sequence-dependent setup times," *Computers & Operations Research*, vol. 90, pp. 142–160, 2018.

[21] H. K. Sarvestani, A. Zadeh, M. Seyfi, and M. Rasti-Barzoki, "Integrated order acceptance and supply chain scheduling problem with supplier selection and due date assignment," *Applied Soft Computing*, vol. 75, pp. 72–83, 2019.

[22] X. Li and J. A. Ventura, "Exact algorithms for a joint order acceptance and scheduling problem," *International Journal of Production Economics*, vol. 223, p. 107516, 2020.

[23] S. Wang and B. Ye, "Exact methods for order acceptance and scheduling on unrelated parallel machines," *Computers & Operations Research*, vol. 104, pp. 159–173, 2019.

[24] G.-H. Wu, C.-Y. Cheng, H.-I. Yang, and C.-T. Chena, "An improved water flow-like algorithm for order acceptance and scheduling with identical parallel machines," *Applied Soft Computing*, vol. 71, pp. 1072–1084, 2018.

[25] V. P. Palakiti, U. Mohan, and V. K. Ganesan, "Order acceptance and scheduling in a parallel machine environment with weighted completion time," *European J. of Industrial Engineering*, vol. 12, no. 4, pp. 535–557, 2018.

[26] S.-H. Chen, Y.-C. Liou, Y.-H. Chen, and K.-C. Wang, "Order acceptance and scheduling problem with carbon emission reduction and electricity tariffs on a single machine," *Sustainability*, vol. 11, no. 19, p. 5432, 2019.

[27] J. Li, X. Zeng, C. Liu, and X. Zhou, "A parallel Lagrange algorithm for order acceptance and scheduling in cluster supply chains," *Knowledge-Based Systems*, vol. 143, pp. 271–283, 2018.

[28] Z. Wang, Y. Qi, H. Cui, and J. Zhang, "A hybrid algorithm for order acceptance and scheduling problem in make-to-stock/make-to-order industries," *Computers & Industrial Engineering*, vol. 127, pp. 841–852, 2019.

[29] M. Yavari, M. Marvi, and A. H. Akbari, "Semi-permutation-based genetic algorithm for order acceptance and scheduling in two-stage assembly problem," *Neural Computing and Applications*, vol. 32, no. 8, pp. 2989–3003, 2020.

[30] H. Li, Y. Gajpal, and C. R. Bector, "A survey of due-date related single-machine with two-agent scheduling problem," *Journal of Industrial & Management Optimization*, vol. 13, no. 5, p. 1, 2019.

[31] P. Melchiors, R. Leus, S. Creemers, and R. Kolisch, "Dynamic order acceptance and capacity planning in a stochastic multi-project environment with a bottleneck resource," *International Journal of Production Research*, vol. 56, no. 1-2, pp. 459–475, 2018.

[32] S. N. Sahu, Y. Gajpal, and S. Debbarma, "Two-agent-based single-machine scheduling with switchover time to minimize total weighted completion time and makespan objectives," *Annals of Operations Research*, vol. 269, no. 1-2, pp. 623–640, 2018.

[33] H. Li, Y. Gajpal, C. Surti, D. Cai, and A. K. Bhardwaj, "Nature-inspired metaheuristics for two-agent scheduling with due date and release time," *Complexity*, vol. 2020, Article ID 1385049, 13 pages, 2020.

[34] B. Sharma, R. K. Thulasiram, and P. Thulasiraman, "Normalized particle swarm optimization for complex chooser option pricing on graphics processing unit," *The Journal of Supercomputing*, vol. 66, no. 1, pp. 170–192, 2013.

[35] T. Jin, Y. Sun, and Y. Zhu, "Time integral about solution of an uncertain fractional order differential equation and application to zero-coupon bond model," *Applied Mathematics and Computation*, vol. 372, p. 124991, 2020.

[36] T. Jin, H. Ding, H. Xia, and J. Bao, "Reliability index and asian barrier option pricing formulas of the uncertain fractional first-hitting time model with caputo type. chaos," *Solitons & Fractals*, vol. 142, Article ID 110409, 2020.

[37] T. Jin and Y. Zhu, "First hitting time about solution for an uncertain fractional differential equation and application to an uncertain risk index model," *Chaos, Solitons & Fractals*, vol. 137, p. 109836, 2020c.