Hindawi Publishing Corporation
Mathematical Problems in Engineering
Volume 2013, Article ID 182598, 12 pages
http://dx.doi.org/10.1155/2013/182598

Research Article

Clarifying Cutting and Sewing Processes with Due Windows Using an Effective Ant Colony Optimization

Rong-Hwa Huang[1] and Shun-Chi Yu[2]

[1] *Department of Business Administration, Fu Jen Catholic University, New Taipei City, Taiwan*
[2] *Graduate School of Business Administration, Fu Jen Catholic University, New Taipei City, Taiwan*

Correspondence should be addressed to Rong-Hwa Huang; 026299@mail.fju.edu.tw

Received 7 December 2012; Revised 3 February 2013; Accepted 19 February 2013

Academic Editor: Tsan-Ming Choi

The cutting and sewing process is a traditional flow shop scheduling problem in the real world. This two-stage flexible flow shop is often commonly associated with manufacturing in the fashion and textiles industry. Many investigations have demonstrated that the ant colony optimization (ACO) algorithm is effective and efficient for solving scheduling problems. This work applies a novel effective ant colony optimization (EACO) algorithm to solve two-stage flexible flow shop scheduling problems and thereby minimize earliness, tardiness, and makespan. Computational results reveal that for both small and large problems, EACO is more effective and robust than both the particle swarm optimization (PSO) algorithm and the ACO algorithm. Importantly, this work demonstrates that EACO can solve complex scheduling problems in an acceptable period of time.

1. Introduction

Garment producers must continuously adjust their production systems, including product design, production, and distribution, to remain competitive in the market. They also use new production techniques and equipment, increase outsourcing, and increase the number of strategic alliances with logistic coordinators to shorten processing times and ensure quick responses (QR) (Uruk et al. [1] and Xiong et al. [2]). However, managers focus less on how an effective sequence of tasks can reduce costs by decreasing wasted time and even wait time during the manufacturing process. A good scheduling technique can efficiently reduce wait time, support the just-in-time shipping of garments, and maximize the marginal value for customer requirements (Huang et al. [3] and Liou et al. [4]).

In garment manufacturing, material must be cut before it is sewn. Waiting for predetermined patterns in the cutting process consumes time (Wong et al. [5]). This work considers cutting and sewing as critical processes in the garment production system. Currently, most producers rely on manual scheduling to collect clothes, causing excessive makespan and overhead costs. The cutting process with due windows before the sewing process can be regarded as a two-stage flow shop problem. Garments must complete the cutting process neither earlier nor later than the completion of the sewing process to ensure economic efficiency. This work considers flexible management using an increased number of identical machines to help enterprises increase profits and lower overhead costs by applying a novel algorithm to solve scheduling problems. Operations scheduling is widely conducted to optimize resource use, increasing production efficiency and customer satisfaction. Production management commonly integrates production-related activities to reduce earliness, tardiness, machine idle costs, inventory costs, and other operating costs. Numerous schemes have been developed to solve scheduling problems, each of which has had mixed success (Huang et al. [3] and Cheng et al. [6]). Of the scheduling problems considered herein, the two-stage flexible flow shop scheduling problem (FFSP) is the most common in the real world. The yield rates of spreading, cutting, and sewing processes are critical to the yield rate of the entire production process. In a highly competitive environment, enterprises expend much considerable effort to minimize earliness and tardiness by production scheduling to reduce operating costs, ensure a short time to market, and generate

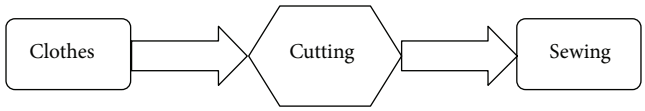

Figure 1: Flow chart of the cutting and sewing process of general apparel manufacturing industry.

a long-term profit. The apparel manufacturing process is a notable example. Each piece of cloth is processed using a cutting machine for the sewing process. Processes that allow jobs to be performed on identical machines to fill various purchase orders are called flexible production processes. Owing to the cost associated with the wait time for sewing, just-in-time capacity represents a bottleneck in the manufacturing process. Therefore, the fabrication of clothing requires an efficient scheduling multiprocessor to ensure timely market delivery. Figure 1 summarizes the limitation.

Wong et al. [5] developed a novel two-tier hierarchical model of the garment manufacturing scheduling system to solve flow shop scheduling problems. Their method had an excellent capacity for solving the flow shop scheduling problem with minimized earliness and tardiness penalties and minimized completion time in the cutting and sewing department. Test results indicated that the proposed scheduling model performed well for the apparel industry. Cheng et al. [6] developed an approximation algorithm with a worst-case ratio of four to solve a job shop scheduling problem using two batch-processing machines by minimizing makespan. Computational results demonstrate that their algorithm can efficiently solve this problem. Flexible flow shop scheduling problems have been extensively studied (Demir and Leyen [7] and Xiong et al. [2]). The related literature enriches this field (Pan et al. [8], Costantini et al. [9], Chiu et al. [10], Huang et al. [11], Choi [12], Hu et al. [13], Shen et al. [14], and Tan et al. [15]). Extending previous investigations, this work develops a novel algorithm for solving such production problems, and incorporates a feasible flexible design concept that uses multimachines to relax the limitations in the two-stage flow shop scheduling problem.

Wang and Liu [16] developed a heuristic method for solving flexible flow shop scheduling problems. It is based on a branch and bound (B&B) algorithm and uses four constructed heuristics to obtain initial upper bounds. Three dominance properties were used to enhance the performance of the proposed heuristic approach in solving the two-stage hybrid flow shop scheduling problem with critical and dedicated machines, thereby minimizing makespan. Computational results demonstrated that their method was effective. Uruk et al. [1] considered a two-machine flow shop scheduling problem with identical jobs. Each job involved three operations, of which the third operation (the flexible operation) was performed on either machine but could not be preempted. Solving the problem involved resolving the assignment of flexible operations to machines and determining processing times for each operation to minimize total manufacturing cost and makespan. Test results revealed that their proposed method using the *ε-constraint* approach reformulated the mathematical programming problem into a mixed integer programming problem and optimally solved either small or large versions of the problem in a reasonable time. Cheng et al. [17] solved a two-machine flow shop scheduling problem using a truncated learning function, in which the actual processing time of the job is a function of the job's position in a schedule and the learning truncation parameter. They developed a novel B&B and three crossover-based genetic algorithms (GAs) to find the optimal and approximate solutions that minimize makespan. Computational test results confirmed the performance of all proposed algorithms under various experimental conditions, demonstrating that the GAs can solve the problem efficiently. The literature enhances related study (Liu et al. [18], Dubois-Lacoste et al. [19], Choi et al. [20], Pan et al. [21], Yeung et al. [22], Pan et al. [8, 21], Mirabi [23], Yeung et al. [24], and Yao and Liu [25]). The two-stage flow shop problem clearly warrants further study, as revealed by the popularity of two-stage flexible flow shop scheduling problems in the literature.

For solving scheduling problems with time windows, this work develops an effective method that prevents unexpected delays and associated large losses during production. Based on customer responses, Solomon and Desrosiers [26] suggested that time windows can be classified as either hard or soft. The nanopattern sapphire substrate is an example of a hard time window. Waiting time before production at early arrival causes significant costs; additionally, etching should be performed during the hard time window. A soft time window is more flexible than a hard window. Therefore, early transfers are permitted outside the time window when the costs are paid in reduced customer satisfaction and related penalties. By using a simple algorithm that was based on the unique characteristics of earliness and tardiness, Huang et al. [3] solved the problem of a parallel machine production system with a common time window. Their algorithm provided the best solution. Liou et al. [4] utilized an encoding scheme-based hybrid algorithm (HA) to solve the two-machine flow shop group scheduling problem (GSP) with sequence-dependent setup and removal times, including job transportation times between machines. The objective was to minimize total completion time. Their algorithm solved the problem more efficiently than did a PSO algorithm or a GA algorithm. In practice, due windows are an important issue in real-time production. Accordingly, by imposing the due windows constraint, this work attempts to minimize weighted earliness and tardiness costs using the just-in-time concept. An effective ant colony optimization (EACO) algorithm is also developed to solve flexible two-stage flexible flow shop scheduling problems to minimize earliness and tardiness. Ant colony optimization and PSO are also compared to obtain better solutions to the above problems and to help enterprises increase profit and reduce overhead costs.

The remainder of this paper is organized as follows. Section 2 formulates the two FFSPs. Section 3 presents the proposed EACO for solving the FFSP as well as the basic PSO. The effect of parameter settings is investigated using the experiment that is designed in Section 4; computation and comparison results are also provided. Finally, Section 5 draws conclusions.

2. Problem Definition

The designated two-stage flexible flow shop scheduling problem of this study is $\mathrm{FF}_2(m_1, m_2) \mid d_i = (d_i^L, d_i^U), Sp_{ijk} \mid w_1 \cdot \sum_{i=1}^{n} E_i + w_2 \cdot \sum_{i=1}^{n} T_i + w_3 \cdot \sum_{i=1}^{n} C_{\max}$ and $\mathrm{FF}_2(m_1, m_2)$ represents a two-stage flexible flow shop environment involving k machines; d_i is the due window; Sp_{ijk} represents different setup times on each machine; $w_1 \cdot \sum_{i=1}^{n} E_i + w_2 \cdot \sum_{i=1}^{n} T_i + w_3 \cdot \sum_{i=1}^{n} C_{\max}$ indicates that the objective is to minimize the sum of weighted earliness, lateness, and makespan.

2.1. Notation. Consider the following.

n = Total number of jobs

m = Total number of machines

J_i = Job of number i

J_{ij} = The jth operation of J_i

M_k = Machine of number k

$J_{ijk} = J_{ij}$ is processed on M_k

Op_i = Number of operations to finish J_i

St_{ijk} = Starting time of J_{ijk}

Sp_{ijk} = Sequence dependent setup time of J_{ij} on M_k

ps_{ijk} = Processing time of J_{ijk}

d_i^L = Lower due window of J_i

d_i^U = Upper due window of J_i

$Z_{ijpqk} = \begin{cases} 1, & \text{if } J_{ij} \text{ is processed before } J_{pq} \text{ on } M_k \\ 0, & \text{if } J_{ij} \text{ is processed after } J_{pq} \text{ on } M_k \end{cases}$

F_i = Completion time of J_i

F_{ijk} = Completion time of J_{ijk}

E_i = Earliness time of J_i

T_i = Tardiness time of J_i

w_1 = Cost coefficient of earliness

w_2 = Cost coefficient of lateness

w_3 = Cost coefficient of makespan.

2.2. Mathematical Model

(1) Objective function:

$$\operatorname{Min} w_1 \cdot \sum_{i=1}^{n} E_i + w_2 \cdot \sum_{i=1}^{n} T_i + w_3 \cdot \sum_{i=1}^{n} C_{\max}; \quad i = 1, 2, \ldots, n. \tag{1}$$

Equation (1): this model minimizes the sum of weighted earliness, tardiness, and makespan.

Subject to:

(2) Earliness:

$$E_i = \operatorname{Max}\left(d_i^L - F_i, 0\right); \quad i = 1, 2, \ldots, n;\ L = 1, 2, \ldots, n. \tag{2}$$

Equation (2): earliness is the largest of $(F_i - d_i^U)$ and 0, which calculates the earliness of job i, and calculated as the due date minus completion time of job i.

(3) Tardiness:

$$T_i = \operatorname{Max}\left(F_i - d_i^U, 0\right); \quad i = 1, 2, \ldots, n;\ U = 1, 2, \ldots, n. \tag{3}$$

Equation (3): lateness is the largest of $(F_i - d_i^U)$ and 0, which calculates the tardiness of job i, and calculated as completion time of job i minus the due date.

(4) Maximum completion time and sequence constraints:

$$F_i = \operatorname{Max}\left(F_{ijk}\right); \quad i = 1, 2, \ldots, n;\ j = 1, 2, \ldots, Op_i;\ k = 1, 2, \ldots, m. \tag{4}$$

Equation (4): completion time of J_i is the maximum completion time on each machine:

$$F_{ijk} = St_{ijk} + Sp_{ijk} + ps_{ijk}; \quad i = 1, 2, \ldots, n;\ j = 1, 2, \ldots, Op_i;\ k = 1, 2, \ldots, m. \tag{5}$$

Equation (5): completion time of J_{ijk} is the sum of its starting time, setup time, and processing time:

$$F_{i(j-1)} \le St_{ij}; \quad i = 1, 2, \ldots, n;\ j = 1, 2, \ldots, Op_i. \tag{6}$$

Equation (6): as for job J_i, the starting time of the jth operation exceeds the completion time of the $(j-1)$th operation:

$$F_{ijk} \times Z_{ijpqk} \le St_{pqk}; \quad i = 1, 2, \ldots, n;\ j = 1, 2, \ldots, \mathrm{O}p_i;\ k = 1, 2, \ldots, m;\ p = 0, 1, \ldots, n;\ q = 1, 2, \ldots, O_{p_0};\ i \ne p. \tag{7}$$

Equation (7): when $Z_{ijpqk} = 1$, J_{ijk} is processed before J_{pqk}, and thus the starting time of J_{pqk} is larger than the completion time of J_{ijk}; meanwhile, when $Z_{ijpqk} = 0$, J_{pqk} is processed before J_{ijk}, and the starting time of J_{pqk} exceeds 0:

$$F_{pqk} - \left(F_{pqk} + 1\right) \times Z_{ijpqk} \le St_{ijk}; \quad i = 1, 2, \ldots, n;\ j = 1, 2, \ldots, \mathrm{O}p_i;\ k = 1, 2, \ldots, m;\ p = 0, 1, \ldots, n;\ q = 1, 2, \ldots, O_{p_0};\ i \ne p. \tag{8}$$

Equation (8): when $Z_{ijpqk} = 1$, J_{ijk} is processed before J_{pqk}, and thus the starting time of J_{ijk} is larger than 0; meanwhile, when $Z_{ijpqk} = 1$, J_{pqk} is processed before J_{ijk}, and thus the starting time of J_{ijk} is larger than the completion time of J_{pqk}.

(5) Makespan:

$$C_{\max} = \operatorname{Max}\left(F_i\right); \quad i = 1, 2, \ldots, n. \tag{9}$$

Equation (9): makespan is the maximum completion time of J_i.

3. Algorithm

3.1. EACO Algorithm Process. The proposed EACO algorithm modifies the state transition rule to find operations with the shortest time window and conducts it to be weighted to a definite local pheromone parameter ($[\vartheta(r,s)]$). Therefore, the ants find the next node quickly and this ant system can be used to solve problems. This adaptation can schedule the proper operation in each scheduling sequence. Artificial ants can identify the fitness of the subsequently feasible nodes while deciding the shortest routes. Therefore, in the search process, the weight of a node increases when it is chosen by many ants that are searching for routes. Following ants have a much higher probability of choosing this feasible node.

The EACO algorithm utilizes similar state transition rule into original ACO (Dorigo et al. [27]) to update global pheromone. Equations (10) and (11) present state transition rule of EACO:

$$S = \begin{cases} \arg\max_{u \in J_k(r)} \left\{ [\kappa(r,u)] \left[\psi(r,u)\right]^{\beta} \left[\vartheta(r,u)\right] \right\}, & q \le Q \\ P_k(r,s), & q > Q, \end{cases} \tag{10}$$

$$P_k(r,s) = \begin{cases} \left([\kappa(r,s)]^{\alpha} \cdot \left[\psi(r,s)\right]^{\beta} \cdot [\vartheta(r,s)]^{\nu^{-1}} \right) \\ \quad \times \left(\sum_{u \in J_k(r)} [\kappa(r,u)]^{\alpha} \cdot \left[\psi(r,u)\right]^{\beta} \right. \\ \qquad \left. \cdot [\vartheta(r,u)]^{\nu^{-1}} \right)^{-1}, & S \in J_k(r) \\ 0, & \text{otherwise.} \end{cases} \tag{11}$$

In (10) and (11), κ represents strongest trail of pheromone and ψ and ν represent visibility and the shortest time window weight coefficient, that is, $\text{Min}\{(d_i^r - d_i^q)/2\}$, respectively. Among them, visibility is the reciprocal of processing time. In state transition rule, first we set a value Q and then a randomly generated number q. If $q \le Q$, an ant at node r will apply exploitation rule, and choose its next node $S \in J_k(r)$ based on (10). $J_k(r)$ is the set of all feasible nodes while ant k is at node r. If $q > Q$, an ant will choose exploration rule and move to a feasible node according to probability distribution of every node.

Except when required by state transition rule, the shortest time window function also utilizes the global pheromone update rule to update pheromones on each node according to the following:

$$\vartheta(r,s)^{\text{new}} = (1-\omega) \times \vartheta(r,s)^{\text{old}} + \omega \cdot \Delta\vartheta(r,s), \tag{12}$$

$$\Delta\vartheta(r,s) = \begin{cases} \dfrac{1}{L_{GB}}, & \text{if } (r,s) \text{ is the current best route} \\ 0, & \text{otherwise.} \end{cases} \tag{13}$$

ω is global pheromone evaporation coefficient, and L_{GB} is the best solution in each iteration.

Updating the global pheromone values after each iteration reinforces the visibility of the best route, increasing the probability that a node is properly chosen to pursue the optimal solution, and the solutions converge as the amount of pheromones on the nodes on the best route increases.

Although the state transition rule and global pheromone update rule are used, the EACO computation is just the same as the ACO computation. The algorithm firstly releases a specified number of ants in one iteration, searches for the best route, and memorizes which nodes are on the best route using the global pheromone update rule. The algorithm terminates when it satisfies a stop criterion, such as a certain number of iterations or convergence.

3.2. PSO Algorithm Process. The PSO algorithm is based on the way in which birds find food. Individual birds do not know the location of food in a particular search space. However, they move in a direction for a distance that is determined by their experiences and those of the other birds that they are following. The PSO algorithm treats each bird as a particle in which the leader is the nearest particle to food, representing the optimal solution.

Kennedy and Eberhart [28] developed a novel PSO algorithm that combines local and global search. Their algorithm has been successfully applied to numerous engineering problems (Liou et al. [4]).

Kennedy and Eberhart [29] developed a binary PSO (BPSO) algorithm by constructing a discrete space to improve upon PSO's former research limitation in a continuous space. Binary particle swarm optimization using binary notation yields various discretized solutions and solves 0-1-style problems. Particle swarm optimization can increase the feasibility of solving the scheduling problems in every field.

Traditional PSO is based on iterated solutions that are found by randomly generated particles. The solution procedure consists of the following three steps: initialization, searching for and updating the optimal solution, and search termination.

Step 1 (perform initialization). In a multidimensional space, particle coordinates and velocity parameters are randomly generated. Fitness values are calculated to optimize the position for the initial coordinate of the population. Every particle then inherits the best coordinates.

Step 2 (search for and update the optimal solution). The position of every particle is modified based on the following equations:

$$\begin{aligned} V_p^t = I \times V_p^{t-1} &+ c_1 \times \text{rand}_1 \times \left(\text{PL}_p^{t-1} - X_p^{t-1} \right) \\ &+ c_2 \times \text{rand}_2 \times \left(\text{PG}^t - X_p^{t-1} \right), \end{aligned} \tag{14}$$

$$X_p^t = X_p^{t-1} + V_p^t. \tag{15}$$

If $V_p^t > V_{\max}$, then $V_p^t = V_{\max}$; if $V_p^t > -V_{\max}$, then $V_p^t = -V_{\max}$.

In (14), I represents inertial weight, V_p^{t-1} is former round velocity, c_1 and c_2 are individual and population reference

Table 1: Due window parameter settings.

Data type	DR	TF
I	0.6	0.2
II	1.0	0.2
III	0.6	0.5
IV	1.0	0.5

weight, rand_1 and rand_2 are uniform distributions between $U[0, 1]$, PL_p^{t-1} denotes former local coordinates of individual particle, X_p^{t-1} indicates former best coordinates of all particles, and PG^t denotes former local coordinates of population particles. In (15), the result of former coordinates and new velocity $(X_p^{t-1} + V_p^t)$ can generate best new coordinates (X_p^t). According to Newton's law of motion, the acceleration of a body is parallel and directly proportional to the net force, while being inversely proportional to the mass. PSO transforms into three products (14). First, the product of inertial weight and former round velocity represents the inertial affection. The gap deducted from the individual and population best coordinate multiplies acceleration learning constant and then forms two velocities. Finally, all particles obtain a new velocity and modify their position by using vector addition to summarize all affections.

After the positions of all particles are modified positions using (15), the fitness value is recalculated. If the optimal coordinates of individual or population particles are found, the algorithm updates; otherwise, computations of the former coordinates continue.

Step 3 (stop searching). The algorithm repeats Step 2 to obtain better fitness value until the stop criterion is satisfied. Following completion of the computations, the best solution for the objective function is obtained.

4. Data Test and Analysis

This study considers the two-stage flexible flow shop problem. All assumptions regarding job numbers and machines used in the simulation refer to the categories for solving the overlapping shop problem proposed by Huang and Yang [30]. In small scale data testing, the number of machines is set to $U[1, 5]$ by solving problems involving 3 to 8 jobs, respectively. In the large scale data test, the number of jobs is 50, 100, 150, and 200 and the number of machines is $U[6, 10]$. All job processing times are randomly generated by $U[1, 50]$.

As for the issue of due window, this study adopts the formula derived by Zheng et al. [31], which uses tardy factor (TF) and due date range (DR) to set the due window range between $[(1 - \text{TF} - \text{DR}/2) \sum_{i=1}^{m} P_i(1 + (M - 1) * 0.2, (1 - \text{TF} + \text{DR}/2) \sum_{i=1}^{m} P_i(1 + (M - 1) * 0.2)]$ and where the following types are used for simulation test. P_i represents processing time and M means number of machines. TF and DR factors generate various due windows for comparison. Table 1 lists the parameters.

In the weighting of parameters for measuring solutions whether they meet stop criterion, this work incorporates decision processes that are used by enterprises to increase the weight of the cost of inventory to reduce stock pressure or increase the cost of tardiness for important orders and simultaneously to minimize makespan. This study thus adopts all possible conditions not just using unique weight cost but divides weighted costs into three categories $w_1 = 0.25$, $w_2 = 0.50$, $w_3 = 0.25$; $w_1 = 0.50$, $w_2 = 0.25$, $w_3 = 0.25$; and $w_1 = 0.25$, $w_2 = 0.25$, $w_3 = 0.50$ for simulation testing.

The settings of the parameters in the algorithm significantly influence the execution results. This work optimizes the parameter settings following a literature review and a pretest adjustment: $V_{\max} = 99$, $I = 1$, $c_1 = 2$, $c_2 = 2$,$nL = 100$, and $nP = 10$. IP is coded in Lingo 12 and all heuristics are coded in C++ programming language. All experiments are conducted on a PC with an Intel Core i5 CPU 760, 2.81 GHz, and 3.46 GB RAM. Computational results are analyzed and discussed.

4.1. Effectiveness Testing. To test the efficiency of the EACO algorithm, the EACO algorithm is used to solve a distinct problem to verify its effectiveness and compare it with the effectiveness of other methods. The formulae for measuring effectiveness in small problems are as follows.

Improvement ratio of effectiveness:

Effectiveness of EACO in contrast to PSO

$$= \left(1 - \frac{\text{Outcome}_{\text{EACO}} - \text{Outcome}_{\text{IP}}}{\text{Outcome}_{\text{PSO}} - \text{Outcome}_{\text{IP}}}\right) \times 100\%.$$

Effectiveness of EACO in contrast to ACO

$$= \left(1 - \frac{\text{Outcome}_{\text{EACO}} - \text{Outcome}_{\text{IP}}}{\text{Outcome}_{\text{ACO}} - \text{Outcome}_{\text{IP}}}\right) \times 100\%. \tag{16}$$

To demonstrate how the EACO algorithm outperforms PSO and ACO algorithms for small problems, 30 test datasets were calculated 30 times to compare computational times, mean solutions, and percentage difference among IP and the EACO, PSO, and ACO algorithms. With respect to the solving of the large problem, preliminary test results demonstrated that IP computational time increased exponentially with the number of jobs and stages (Huang et al. [3]). Hence, this work tests only jobs and machines with (0.25, 0.50, 0.25) in a large problem without IP. Table 2 lists computational results.

This study also tests the weight cost conditions of (0.50, 0.25, 0.25) and (0.25, 0.25, 0.50). Table 3 summarizes all of the results.

The EACO algorithm is better than the PSO and ACO algorithms in finding the mean solutions to 30 problems. Under distinct conditions of due window and cost weighting, the total improvements in contrast to PSO and ACO over mean solutions are calculated to be 25.26% and 30.94%, respectively. Accordingly, the improvements in contrast to PSO and ACO over mean solutions that are achieved by

Table 2: Effectiveness test results for small problems ($w_1 = 0.25,\ w_2 = 0.50,\ w_3 = 0.25$).

Data type	n	IP		ACO		PSO		EACO	
		Average	CPU time (second)	Average	CPU time (second)	Average	CPU time (second)	Average	CPU time (second)
I	3	**71.50**	5.01	**81.31**	0.81	**80.19**	0.82	**82.42**	0.81
	4	**146.20**	56.14	**161.26**	1.03	**160.21**	1.02	**162.06**	1.05
	5	**271.60**	262.32	**293.54**	1.86	**293.34**	1.84	**289.47**	1.85
	6	**375.90**	402.16	**391.27**	0.85	**391.62**	0.83	**388.21**	0.84
	7	**497.10**	555.08	**526.29**	1.95	**517.79**	1.92	**514.38**	1.93
	8	**584.60**	832.31	**639.62**	1.99	**641.48**	1.97	**632.25**	1.98
II	3	**132.90**	5.12	**145.24**	0.84	**147.49**	0.83	**143.01**	0.83
	4	**247.30**	55.43	**262.43**	1.12	**265.18**	1.03	**260.32**	1.05
	5	**389.60**	264.66	**406.25**	1.86	**417.52**	1.84	**405.31**	1.83
	6	**514.50**	305.92	**579.33**	0.85	**587.26**	0.83	**570.07**	0.83
	7	**712.60**	552.41	**851.24**	1.94	**840.41**	1.92	**837.11**	1.92
	8	**806.10**	753.85	**928.29**	2.02	**919.07**	2.24	**911.05**	2.28
III	3	**79.40**	5.24	**96.14**	0.84	**96.87**	0.82	**95.02**	0.83
	4	**181.80**	53.75	**202.15**	1.13	**201.17**	1.1	**196.18**	1.11
	5	**332.40**	241.22	**389.52**	1.86	**378.62**	1.84	**361.17**	1.85
	6	**403.80**	345.19	**496.15**	0.87	**509.19**	0.83	**496.72**	0.83
	7	**512.90**	564.32	**570.46**	1.97	**568.18**	1.92	**565.23**	1.92
	8	**613.70**	613.93	**695.35**	2.08	**677.29**	2.12	**667.21**	2.06
IV	3	**141.60**	3.97	**157.17**	0.84	**155.32**	0.83	**150.47**	0.83
	4	**259.30**	51.32	**298.74**	1.12	**302.72**	1.03	**285.39**	1.05
	5	**402.50**	246.73	**532.29**	1.86	**496.05**	1.83	**471.35**	1.84
	6	**571.20**	365.08	**657.24**	0.88	**662.48**	0.84	**642.98**	0.85
	7	**718.60**	526.04	**786.28**	1.98	**775.43**	1.92	**771.36**	1.92
	8	**813.50**	728.46	**892.19**	2.01	**921.26**	1.98	**881.75**	2.02

the EACO algorithm in solving small problems exceed those obtained using the PSO and ACO algorithms.

4.2. Robustness Testing. To examine further whether the EACO algorithm is more robust than the PCO and ACO algorithms, this work determines the improvement in robustness for small and large problems.

Improvement ratio of robustness:

Robustness test result for small problems,

$$\text{in contrast to PSO} = \left(1 - \frac{\text{Outcome}_{\sigma^2(\text{EACO})}}{\text{Outcome}_{\sigma^2(\text{PSO})}}\right) \times 100\%. \tag{17}$$

Robustness test result for small problems,

$$\text{in contrast to ACO} = \left(1 - \frac{\text{Outcome}_{\sigma^2(\text{EACO})}}{\text{Outcome}_{\sigma^2(\text{ACO})}}\right) \times 100\%. \tag{18}$$

A randomly generated problem is tested 30 times using the EACO, PSO, and ACO algorithms under three cost conditions. The IP is presented once to compare the deviation of solutions with that of the PSO and ACO algorithms. This work calculates the worst solution, best solution, mean solution, standard deviations, and CPU time (in seconds) for each scenario. Table 4 summarizes those results.

This study also tests the weight cost conditions of (0.50, 0.25, 0.25) and (0.25, 0.25, 0.50). Table 5 summarizes those results.

The test results demonstrate that improvement ratios of robustness are excellent and outstanding in each test set within 2.3 s. The total improvements from using the EACO algorithm in the robustness ratios in contrast to the PSO and ACO algorithms are 26.93% and 28.75%, respectively. These ratios demonstrate that the EACO algorithm is more robust than the PSO and ACO algorithms in solving small two-stage flexible flow shop scheduling problems.

4.3. Large Problems. In real scheduling problems, the complexity of amount of jobs is multiplied by various orders. Therefore, following the small problem testing, large problems are tested. Pretesting demonstrates that the CPU time that is required to solve IPs increases exponentially with the total number of jobs. The CPU time for IP is significantly large, so the optimal solution is unidentified preventing the optimal solution from being found. Thus, the IP solution is omitted in tests that involve large problems. This work also compares the solving capacity of the EACO algorithm with

Table 3: Effectiveness test results for small problems contrast to PSO and ACO.

Data type	n	$w_1 = 0.25, w_2 = 0.50, w_3 = 0.25$		$w_1 = 0.50, w_2 = 0.25, w_3 = 0.25$		$w_1 = 0.25, w_2 = 0.25, w_3 = 0.50$	
		EACO versus PSO	EACO versus ACO	EACO versus PSO	EACO versus ACO	EACO versus PSO	EACO versus ACO
I	3	−25.66%	−11.31%	34.26%	33.64%	−21.21%	−10.33%
	4	−13.20%	−5.31%	18.51%	21.27%	−9.64%	12.88%
	5	17.80%	18.55%	46.28%	37.25%	44.39%	37.15%
	6	21.69%	19.91%	25.41%	38.25%	38.19%	29.65%
	7	16.48%	40.80%	56.13%	48.26%	27.74%	28.62%
	8	16.23%	13.40%	31.26%	27.05%	30.25%	43.27%
II	3	30.71%	18.07%	42.52%	30.13%	18.22%	37.67%
	4	27.18%	13.95%	13.29%	10.47%	27.02%	32.01%
	5	43.73%	5.65%	33.15%	53.24%	29.41%	45.84%
	6	23.63%	14.28%	43.19%	38.53%	18.46%	52.61%
	7	2.58%	10.19%	32.03%	41.21%	52.25%	65.48%
	8	7.10%	14.11%	18.24%	30.23%	29.24%	39.47%
III	3	10.59%	6.69%	35.41%	40.62%	50.14%	2.02%
	4	25.76%	29.34%	27.56%	32.26%	21.52%	23.97%
	5	37.75%	49.63%	31.26%	42.25%	43.25%	58.25%
	6	11.83%	−0.62%	28.15%	22.56%	−15.59%	8.03%
	7	5.34%	9.09%	−21.13%	61.51%	34.28%	44.96%
	8	15.85%	34.46%	29.24%	42.87%	29.16%	26.47%
IV	3	35.35%	43.03%	39.15%	65.02%	59.38%	75.43%
	4	39.91%	33.85%	−10.67%	35.38%	24.91%	35.17%
	5	26.40%	46.95%	53.55%	56.19%	33.28%	32.32%
	6	21.36%	16.57%	30.57%	32.57%	61.45%	65.16%
	7	7.16%	22.04%	25.73%	28.82%	26.41%	47.29%
	8	36.66%	13.27%	27.23%	33.53%	33.72%	34.84%
Avg. improvement		18.43%	19.02%	28.76%	37.63%	28.59%	36.18%
Total improvement (EACO versus PSO)				25.26%			
Total improvement (EACO versus ACO)					30.94%		

those of the PSO and ACO algorithms when they are applied to large problems. Table 6 lists the weight condition of (0.25, 0.50, 0.25).

This study also tests the weight cost conditions of (0.50, 0.25, 0.25) and (0.25, 0.25, 0.50). Table 7 summarizes those results.

For large problems, test results for all test problems indicate that the EACO algorithm is more robust than the PSO or ACO algorithm. The total improvements in robustness are 27.31% and 39.54%, respectively. Thus, the EACO algorithm is more robust than the PSO or ACO algorithm for large two-stage flexible flow shop scheduling problems and it exhibits excellent stability.

4.4. Discussion. Table 8 indicates that the EACO algorithm is 25.26% and 30.94% more effective than the PSO and ACO algorithms, respectively, for small problems. The EACO algorithm is, on average, 26.93% and 28.75% more robust than the PSO and ACO algorithms, respectively, for small problems, demonstrating that the EACO algorithm is excellent at solving such scheduling problems.

Owing to the limitations of IP testing, this work cannot compare the effectiveness of IP with that of other algorithms. The EACO algorithm provides mean improvements in robustness of 27.31% and 39.54% over the PSO and ACO algorithms in solving large problems, respectively, verifying that the EACO algorithm has an excellent capacity to solve such scheduling problems.

5. Summary

A good fashion and textiles operational control systems management can efficiently reduce the manufacturing duration, support the just-in-time shipping of garments, and

Table 4: Robustness test results for small problems ($w_1 = 0.25$, $w_2 = 0.50$, $w_3 = 0.25$).

Data type	n	IP		ACO			PSO			EACO		
		Optimal	CPU time	Best (worst)	Avg. (Standard)	CPU time	Best (worst)	Avg. (Standard)	CPU time	Best (worst)	Avg. (Standard)	CPU time
I	3	79.20	5	80.10	85.07	0.8	80.30	85.02	0.8	80.20	85.36	0.81
				116.30	**0.25**		**115.60**	**0.24**		**115.90**	**0.22**	
	4	148.50	52	157.20	162.3	1.02	158.50	161.96	1	158.90	159.6	1
				177.10	**1.92**		**179.10**	**2.09**		**181.20**	**1.61**	
	5	274.50	246	282.00	292.37	2.01	289.30	292.44	2	291.50	285.25	2
				313.60	**5.91**		**327.90**	**6.05**		**346.50**	**4.69**	
	6	379.00	413	389.20	397.23	0.85	381.20	399.33	0.83	389.20	405.23	0.83
				481.50	**9.42**		**490.70**	**8.23**		**482.70**	**9.17**	
	7	498.10	582	515.00	529.33	1.97	504.20	524.76	1.93	514.20	534.72	1.93
				586.10	**15.47**		**635.60**	**13.41**		**595.50**	**16.4**	
	8	587.90	849	626.40	641.03	2.01	614.70	649.74	2	604.30	657.78	2
				721.80	**23.81**		**736.00**	**23.08**		**725.20**	**21.23**	
II	3	138.90	6	143.90	149.07	0.84	141.80	148.02	0.83	141.20	148.15	0.83
				161.20	**0.23**		**178.40**	**0.29**		**176.90**	**0.19**	
	4	249.60	59	254.30	261.93	1.21	252.60	269.36	1.21	259.60	271.24	1.21
				283.00	**1.39**		**292.30**	**1.28**		**297.80**	**1.52**	
	5	389.20	248	392.10	409.73	1.98	390.10	401.8	1.94	397.50	405.04	1.94
				436.80	**4.71**		**452.70**	**5.97**		**476.20**	**4.92**	
	6	528.90	321	569.70	581.27	0.95	558.90	589.07	0.93	558.90	581.98	0.93
				670.00	**10.44**		**669.40**	**10.24**		**682.40**	**9.19**	
	7	736.00	572	846.40	862.73	1.9	831.50	858.39	1.9	846.40	861.6	1.9
				927.60	**18.73**		**937.00**	**19.59**		**952.40**	**17.63**	
	8	832.60	751	915.80	931.17	2.03	903.10	923.07	2.01	901.20	936.52	2.01
				1087.60	**25.79**		**1071.80**	**23.52**		**1069.10**	**22.64**	
III	3	80.80	5	89.10	98.08	0.86	102.80	108.27	0.85	101.80	102.96	0.85
				112.20	**0.37**		**116.00**	**0.46**		**118.80**	**0.32**	
	4	192.70	52	203.60	210.93	1.1	200.60	208.04	1.03	197.20	206.77	1.05
				234.50	**5.03**		**227.20**	**5.49**		**229.10**	**4.64**	
	5	319.90	256	381.10	396.9	2	399.30	411.67	2	370.10	387.72	2
				431.00	**10.93**		**440.30**	**11.25**		**411.50**	**9.32**	
	6	410.80	361	461.30	501.23	0.85	452.80	511.63	0.83	450.80	495.91	0.83
				612.20	**12.42**		**624.80**	**15.37**		**631.30**	**11.21**	
	7	523.20	589	561.40	591.17	1.99	568.40	578.09	1.92	573.90	576.45	1.92
				658.00	**18.12**		**691.20**	**20.83**		**683.90**	**17.17**	
	8	638.60	625	683.50	709.9	2.01	674.50	721.59	2	679.10	712.77	2
				768.90	**25.97**		**777.30**	**24.12**		**760.30**	**22.81**	
IV	3	150.20	4	161.20	168.55	0.84	160.00	170.3	0.83	160.20	169.27	0.83
				181.60	**0.4**		**181.60**	**0.38**		**180.70**	**0.29**	
	4	268.30	50	289.20	303.67	1.12	291.30	313.09	1.11	298.30	302.9	1.11
				341.00	**3.84**		**350.50**	**3.39**		**349.80**	**3.24**	
	5	415.70	259	487.30	528.57	1.86	478.40	533.3	1.84	472.90	524.31	1.84
				554.00	**10.6**		**547.60**	**12.39**		**567.20**	**9.38**	
	6	579.00	392	617.30	669.87	0.82	609.00	678.24	0.81	595.00	680.11	0.81
				780.00	**13.34**		**798.70**	**14.3**		**780.20**	**11.22**	
	7	724.70	554	786.50	795.16	1.95	778.50	788.74	1.94	765.00	780.89	1.94
				1092.40	**18.93**		**1099.40**	**19.92**		**1096.40**	**16.8**	
	8	828.20	714	872.00	897.83	2.21	881.10	890.08	1.99	887.50	892.52	1.99
				1127.80	**23.38**		**1132.50**	**27.46**		**1199.00**	**20.64**	

Table 5: Improvement ratio of robustness using EACO in small problems, in contrast to PSO and ACO.

Data type	n	$w_1 = 0.25, w_2 = 0.50, w_3 = 0.25$		$w_1 = 0.50, w_2 = 0.25, w_3 = 0.25$		$w_1 = 0.25, w_2 = 0.25, w_3 = 0.50$	
		EACO versus PSO	EACO versus ACO	EACO versus PSO	EACO versus ACO	EACO versus PSO	EACO versus ACO
I	3	15.97%	22.56%	18.27%	25.16%	41.01%	43.16%
	4	40.66%	29.68%	51.41%	25.68%	29.16%	38.48%
	5	39.91%	37.02%	32.01%	34.42%	32.07%	45.64%
	6	−24.15%	5.24%	40.11%	53.42%	34.42%	15.83%
	7	−49.57%	−12.38%	6.82%	23.57%	−24.21%	2.86%
	8	15.39%	20.50%	32.18%	31.06%	40.25%	49.47%
II	3	57.07%	31.76%	32.65%	30.56%	22.37%	34.54%
	4	−41.02%	−19.58%	34.95%	45.87%	21.47%	36.01%
	5	32.08%	−9.12%	41.53%	42.48%	32.49%	71.26%
	6	19.46%	22.51%	−30.23%	31.75%	52.51%	46.51%
	7	19.01%	11.40%	25.07%	−18.23%	25.04%	31.86%
	8	7.34%	22.94%	27.65%	15.26%	48.47%	37.61%
III	3	51.61%	25.20%	21.21%	21.02%	−31.26%	45.67%
	4	28.57%	14.91%	37.25%	16.21%	61.42%	35.07%
	5	31.37%	27.29%	32.47%	22.06%	51.37%	47.33%
	6	46.81%	18.54%	−22.09%	45.19%	9.53%	50.76%
	7	32.05%	10.21%	31.20%	31.63%	−25.76%	8.62%
	8	10.57%	22.86%	54.19%	43.68%	54.79%	38.65%
IV	3	41.76%	47.44%	40.71%	23.64%	22.92%	21.48%
	4	8.65%	28.81%	−23.82%	0.00%	37.68%	43.64%
	5	42.69%	21.69%	52.08%	29.69%	60.42%	48.92%
	6	38.44%	29.26%	44.32%	48.13%	50.62%	52.39%
	7	28.87%	21.24%	18.11%	23.02%	43.16%	31.01%
	8	43.50%	22.07%	51.36%	37.19%	62.25%	58.42%
Avg. improvement		22.38%	18.83%	27.06%	28.44%	31.34%	38.97%
Total improvement (EACO versus PSO)				26.93%			
Total improvement (EACO versus ACO)					28.75%		

strengthen competitiveness of enterprises. This study is to propose a novel algorithm to solve the cutting and sewing processes in a traditional flow shop scheduling problem that comprises just-in-time and costs to minimize earliness, tardiness, and makespan. Since the problem is NP hard, exact-solution methods cannot satisfy the agile requirements for solving the complex problems.

Numerous investigations demonstrate that the ACO and PSO are effective for solving scheduling problems. Hence, this study aims at the characteristic of cutting and sewing processes constructing an EACO algorithm to significantly solve the FFSP in the fashion and textiles industry. These heuristics cannot ensure getting the optimal solution; however, the proposed method must test both effectiveness and robust performance to solve the attempted problems. The results of data testing show that the proposed EACO algorithm is superior to the basic ACO and PSO algorithms. By using the EACO algorithm, this study obtains excellent results and provides the production managers and scheduling operators with an alternative way to deal with the attempted problems.

6. Conclusions

In the manufacturing of garments, the garments must be cut before they are sewn, and the cutting process consuming time requires that time be taken in waiting for patterns. Producers depend only on manual scheduling when planning production activities, likely causing excessive makespan and overhead costs. This work applies the EACO algorithm to solve this scheduling problem efficiently.

Following careful testing, this work solves the two-stage flexible flow shop scheduling problem with due window

Table 6: Robustness test results for large problems ($w_1 = 0.25,\ w_2 = 0.50,\ w_3 = 0.25$).

Data type	n	ACO			PSO			EACO		
		Best (worst)	Avg. (Std.)	CPU time	Best (worst)	Avg. (Std.)	CPU time	Best (worst)	Avg. (Std.)	CPU time
I	50	2142	2245.73	9.06	2102.7	2181.07	9.05	2014.3	2093.8	10.03
		2334.1	**48.18**		**2271.6**	**50.16**		**2199**	**42.04**	
	100	3862	4052.17	12.73	3787.5	3945.02	12.7	3460.3	3704.16	12.7
		4230.2	**91.21**		**4105.2**	**76.25**		**4798.7**	**65.04**	
	150	6342.6	6962.7	23.26	6361.4	6680.35	23.2	5952.3	6311.41	23.2
		7395	**227.54**		**7041.1**	**167.62**		**6813.5**	**124.69**	
	200	10252.2	10725.04	34.05	9749.65	10232.95	34.02	9865.2	10677.51	34.04
		11382.5	**257.62**		**10952.65**	**246.41**		**11411.4**	**206.77**	
II	50	2412.3	2537.45	9.07	2375.5	2592.08	9.01	2357	2392.8	9.03
		2604.8	**59.18**		**2792.1**	**45.79**		**2891.8**	**62.34**	
	100	4225.2	4411.61	12.53	4057.9	4437.43	12.5	3972.2	4407.94	12.51
		5587.1	**94.75**		**5682**	**82.08**		**5254.6**	**72.64**	
	150	7119	7588.2	23.16	7084	7451.48	22.1	7387.9	7801.52	23.4
		8024.3	**214.03**		**8726.1**	**141.49**		**8427.1**	**152.71**	
	200	9526	10216.47	34.03	9431.2	9863.23	34.15	9321.7	9931.53	34.41
		10852.3	**387.62**		**10927.6**	**196.42**		**11072**	**146.28**	
III	50	2502	2606.72	11.1	2488.1	2587.94	11.9	2391.1	2517.68	11.02
		2746.7	**32.09**		**2645**	**43.11**		**2775.6**	**46.01**	
	100	4371.5	4913.59	15.63	4368	4897.42	14.6	4826.2	4822.51	15.6
		5838	**117.24**		**5679.3**	**73.69**		**5701**	**62.76**	
	150	6362	6862.47	23.06	6272.4	6980.27	23.5	6116.2	6742.05	23.8
		7478.1	**137.09**		**7595.8**	**151.58**		**7692.6**	**118.46**	
	200	11687	11897.02	45.08	10725.1	11956.01	46.02	11218.5	11723.49	45.03
		12213.5	**259.73**		**12405.6**	**176.11**		**12871.8**	**135.83**	
IV	50	2757.5	2872.46	9.06	2601.8	2712.39	9.87	2605.2	2807.25	9.53
		2987.4	**52.12**		**2893.5**	**59.16**		**2971**	**41.23**	
	100	4752	4982.57	12.15	4718.3	4854.81	12.33	4519.1	4851.33	12.04
		5207.2	**107.92**		**5426**	**86.25**		**5762.7**	**70.29**	
	150	6852.1	7611.53	23.22	6927	7528.47	23.17	6803.2	7482.16	23.03
		7963.5	**145.98**		**7853.7**	**179.63**		**7625**	**101.32**	
	200	10714	11375.54	34.05	10521.5	11421.79	34.27	10817.6	11302.58	34.68
		12729.1	**352.14**		**11937.3**	**241.36**		**12771**	**195.47**	

constraints. The objective function minimizes total weighted earliness, tardiness, and makespan. Previous studies have demonstrated that PSO and ACO algorithms can solve scheduling problems, but that they have limited effectiveness and robustness. This work demonstrates that the two-stage flexible flow shop scheduling problem with due window constraints is an NP-hard problem.

In this work, a novel EACO algorithm, based on the ACO algorithm, is utilized to solve the two-stage flexible flow shop scheduling problem. The EACO algorithm is 25.26% and 30.94% more effective for solving small problems than the PSO and ACO algorithms, respectively. EACO is much more effective, on average, than the other two algorithms in solving large problems. Furthermore, average improvements in robustness in solving small problems are 26.93% and 28.75%, respectively, confirming that the EACO algorithm significantly outperforms the PSO and ACO algorithms in both effectiveness and efficiency. Enterprises can exploit the advantages of the proposed algorithm to generate profits and reduce overhead manufacturing costs.

This work offers IP solutions for the best solutions within a satisfactory time and the EACO algorithms provide solutions for quick response to market within a short time. Computational results prove that the EACO algorithm has

Table 7: Improvement ratio of robustness using EACO in large problems, in contrast to PSO and ACO.

Data type	n	$w_1 = 0.25, w_2 = 0.50, w_3 = 0.25$		$w_1 = 0.50, w_2 = 0.25, w_3 = 0.25$		$w_1 = 0.25, w_2 = 0.25, w_3 = 0.50$	
		EACO versus PSO	EACO versus ACO	EACO versus PSO	EACO versus ACO	EACO versus PSO	EACO versus ACO
I	50	29.76%	23.86%	22.07%	24.06%	31.27%	60.87%
	100	27.24%	49.15%	29.02%	42.48%	31.38%	67.51%
	150	44.66%	69.97%	−33.19%	61.75%	54.63%	42.09%
	200	29.59%	35.58%	32.19%	37.13%	49.23%	32.83%
II	50	−85.35%	−10.96%	27.56%	39.49%	30.72%	−62.49%
	100	21.68%	41.22%	30.12%	−49.03%	24.75%	52.66%
	150	−16.49%	49.09%	35.25%	72.19%	32.24%	43.84%
	200	44.54%	85.76%	41.11%	82.72%	48.05%	61.25%
III	50	−13.91%	−105.57%	27.26%	52.42%	38.25%	45.34%
	100	27.46%	71.34%	32.23%	74.15%	−52.87%	51.94%
	150	38.93%	25.33%	32.46%	60.02%	37.59%	55.42%
	200	40.51%	72.65%	49.28%	77.19%	41.32%	40.89%
IV	50	51.43%	37.42%	−21.54%	34.26%	42.17%	58.04%
	100	33.58%	57.58%	32.27%	53.66%	36.26%	78.54%
	150	68.18%	51.83%	38.64%	67.81%	37.31%	−56.45%
	200	34.41%	69.19%	31.33%	−76.82%	46.13%	48.73%
Avg. improvement		23.51%	38.97%	25.38%	40.84%	33.03%	38.81%
Total improvement (EACO versus PSO)				27.31%			
Total improvement (EACO versus ACO)					39.54%		

Table 8: Total test results for small and large problems.

		Avg. improvement ratio of effectiveness	Avg. improvement ratio of robustness	F-test	t-test
Small problems	EACO versus PSO	25.26%	26.93%	*	*
	EACO versus ACO	30.94%	28.75%	*	*
Large problems	EACO versus PSO	—	27.31%	*	*
	EACO versus ACO	—	39.54%	*	*

*Significant difference (all conditions are tested by $^{*}P < 0.1$; $^{**}P < 0.05$; $^{***}P < 0.01$).

a higher solving capacity than the common ACO and PSO algorithms. The proposed method reduces waiting and tardiness costs, helping enterprises simultaneously shorten makespan, increase profits, and lower overhead costs.

Acknowledgments

The authors thank the Editor and anonymous reviewers for their valuable comments and suggestions for the improvement of this paper. They would like to thank the National Science Council, Taiwan, for supporting this research under Contract no. NSC 100-2221-E-030-005.

References

[1] Z. Uruk, H. Gultekin, and M. S. Akturk, "Two-machine flowshop scheduling with flexible operations and controllable processing times," *Computers and Operations Research*, vol. 40, no. 2, pp. 639–653, 2013.

[2] J. Xiong, L. N. Xing, and Y. W. Chen, "Robust scheduling for multi-objective flexible job-shop problems with random machine breakdowns," *International Journal of Production Economics*, vol. 141, no. 1, pp. 112–126, 2013.

[3] R. H. Huang, C. L. Yang, and H. T. Huang, "Parallel machine scheduling with common due window," *Journal of the Operational Research Society*, vol. 61, pp. 640–646, 2010.

[4] C. D. Liou, Y. C. Hsieh, and Y. Y. Chen, "A new encoding scheme-based hybrid algorithm for minimising two-machine flow-shop group scheduling problem," *International Journal of Systems Science*, vol. 44, no. 1, pp. 77–93, 2013.

[5] W. K. Wong, C. K. Chan, and W. H. Ip, "A hybrid flowshop scheduling model for apparel manufacture," *International Journal of Clothing Science and Technology*, vol. 13, no. 2, pp. 115–131, 2001.

[6] B. Cheng, S. L. Yang, and Y. Ma, "Minimising makespan for two batch-processing machines with non-identical job sizes in job shop," *International Journal of Systems Science*, vol. 43, no. 12, pp. 2185–2192, 2012.

[7] Y. Demir and S. K. leyen, "Evaluation of mathematical models for flexible job-shop scheduling problems," *Applied Mathematical Modelling*, vol. 37, no. 3, pp. 977–988, 2013.

[8] Q. K. Pan, M. F. Tasgetiren, P. N. Suganthan, and T. J. Chua, "A discrete artificial bee colony algorithm for the lot-streaming flow shop scheduling problem," *Information Sciences*, vol. 181, no. 12, pp. 2455–2468, 2011.

[9] S. Costantini, G. de Gasperis, A. Provetti, and P. Tsintza, "A heuristic approach to proposal-based negotiation: with applications in fashion supply chain management," *Mathematical Problems in Engineering*. In press.

[10] C. H. Chiu, T. M. Choi, H. T. Yeung, and Y. Zhao, "Sales rebate contracts in fashion supply chains," *Mathematical Problems in Engineering*, vol. 2013, Article ID 908408, 19 pages, 2013.

[11] M. Huang, X. Wang, Fuqiang Lu, and B. I. Hualing, "A coordination of risk management for supply chains organized as virtual enterprises," *Mathematical Problems in Engineering*. In press.

[12] T. M. Choi, "Optimal return service charging policy for fashion mass customization program," *Service Science*, 2013.

[13] Z. H. Hu, Y. X. Zhao, and T. M. Choi, "Vehicle routing problem for fashion supply chains with cross-docking," *Mathematical Problems in Engineering*. In press.

[14] B. Shen, T. M. Choi, Y. Wang, and C. K. Y. Lo, "The coordination of fashion supply chains with a risk averse supplier under the markdown money policy," *IEEE Transactions on Systems, Man, and Cybernetics—Systems*, vol. 43, no. 2, pp. 266–276, 2013.

[15] Y. Tan, J. B. Guan, and H. R. Karimi, "The impact of subsidy policy on total factor productivity: an empirical analysis for China's cotton production," *Mathematical Problems in Engineering*, vol. 2013, Article ID 248537, 8 pages, 2013.

[16] S. J. Wang and M. Liu, "A heuristic method for two-stage hybrid flow shop with dedicated machines," *Computers and Operations Research*, vol. 40, no. 1, pp. 438–450, 2013.

[17] T. C. E. Cheng, C. C. Wu, J. C. Chen, W. H. Wu, and S. R. Cheng, "Two-machine owshop scheduling with a truncated learning function to minimize the makespan," *International Journal of Production Economics*, vol. 141, no. 1, pp. 79–86, 2013.

[18] H. Liu, L. Gao, and Q. Pan, "A hybrid particle swarm optimization with estimation of distribution algorithm for solving permutation flowshop scheduling problem," *Expert Systems with Applications*, vol. 38, no. 4, pp. 4348–4360, 2011.

[19] J. Dubois-Lacoste, M. López-Ibáñez, and T. Stützle, "A hybrid TP+PLS algorithm for bi-objective flow-shop scheduling problems," *Computers and Operations Research*, vol. 38, no. 8, pp. 1219–1236, 2011.

[20] T. M. Choi, W. K. Yeung, and T. C. E. Cheng, "Scheduling and coordination of multi-suppliers single-warehouse-operator single-manufacturer supply chains with variable production rates and storage costs," *International Journal of Production Research*, 2012.

[21] Q. K. Pan, P. N. Suganthan, J. J. Liang, and M. F. Tasgetiren, "A local-best harmony search algorithm with dynamic sub-harmony memories for lot-streaming flow shop scheduling problem," *Expert Systems with Applications*, vol. 38, no. 4, pp. 3252–3259, 2011.

[22] W. K. Yeung, T. M. Choi, and T. C. E. Cheng, "Optimal scheduling of a single-supplier single-manufacturer supply chain with common due windows," *IEEE Transactions on Automatic Control*, vol. 55, no. 12, pp. 2767–2777, 2010.

[23] M. Mirabi, "Ant colony optimization technique for the sequence-dependent flowshop scheduling problem," *International Journal of Advanced Manufacturing Technology*, vol. 55, no. 1–4, pp. 317–326, 2011.

[24] W. K. Yeung, T. M. Choi, and T. C. E. Cheng, "Supply chain scheduling and coordination with dual delivery modes and inventory storage cost," *International Journal of Production Economics*, vol. 132, no. 2, pp. 223–229, 2011.

[25] J. Yao and L. Liu, "Optimization analysis of supply chain scheduling in mass customization," *International Journal of Production Economics*, vol. 117, no. 1, pp. 197–211, 2009.

[26] M. M. Solomon and J. Desrosiers, "Time window constrained routing and scheduling problems," *Transportation Science*, vol. 22, no. 1, pp. 1–13, 1988.

[27] C. A. Dorigo, M. Dorigo, V. Manjezzo, and M. Trubian, "Ant system for job-shop scheduling," *Belgian Journal of Operations Research*, vol. 34, pp. 39–53, 1994.

[28] J. Kennedy and R. Eberhart, "Particle swarm optimization," *IEEE International Conference on Neural Networks*, vol. 4, no. 11-12, pp. 1942–1948, 1995.

[29] J. Kennedy and R. C. Eberhart, "A discrete binary version of the particle swarm algorithm," in *Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics*, vol. 5, pp. 4104–4108, October 1997.

[30] R. H. Huang and C. L. Yang, "Solving a multi-objective overlapping flow-shop scheduling," *International Journal of Advanced Manufacturing Technology*, vol. 42, no. 9-10, pp. 955–962, 2009.

[31] W. X. Zheng, H. Nagasawa, and N. Nishiyama, "Single-machine scheduling for minimizing total cost with identical, asymmetrical earliness and tardiness penalties," *International Journal of Production Research*, vol. 31, no. 7, pp. 1611–1620, 1993.