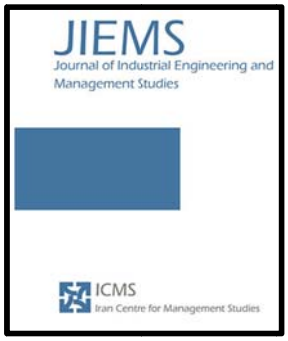
JIEMS
Journal of Industrial Engineering and Management Studies

Vol. 1, No. 1, pp. 1 - 19

www.jiems.icms.ac.ir



A cloud-based simulated annealing algorithm for order acceptance problem with weighted tardiness penalties in permutation flow shop scheduling

M.M. Asgari Tehrani[1] , M. Zandieh[1,*]

Abstract
Make-to-order is a production strategy in which manufacturing starts only after a customer's order is received; in other words, it is a pull-type supply chain operation since manufacturing is carried out as soon as the demand is confirmed. This paper studies the order acceptance problem with weighted tardiness penalties in permutation flow shop scheduling with MTO production strategy, the objective function of which is to maximize the total net profit of the accepted orders. The problem is formulated as an integer-programming (IP) model, and a cloud-based simulated annealing (CSA) algorithm is developed to solve the problem. Based on the number of candidate orders the firm receives, fifteen problems are generated. Each problem is regarded as an experiment, which is conducted five times to compare the efficiency of the proposed CSA algorithm to the one of simulated annealing (SA) algorithm previously suggested for the problem. The experimental results testify to the improvement in objective function values yielded by CSA algorithm in comparison with the ones produced by the formerly proposed SA algorithm.

Keywords: Permutation flow shop scheduling, order acceptance, weighted tardiness, cloud-based simulated annealing algorithm, make-to-order production strategy.

Received: March 2014-08

Revised: May 2014-14

Accepted: July 2014-19

1 Introduction

The objective function of the classical flow-shop scheduling problem is to minimize make-span where n Jobs (or Orders) are processed through m Stages. At any time, each machine processes at most one job, and each job is processed on at most one machine. If all jobs are processed with the same sequence on all machines, the process is called permutation flow shop scheduling (Potts et al. 1991). Most of the previously proposed models for flow shop scheduling problem assume that all jobs have to be processed through the stages without being declined; in contrast, firms

* Corresponding Author.
[1] Department of Industrial Management, Faculty of Management and Accounting, Shahid Beheshti University, G. C., Tehran, Iran.

applying make-to-order (MTO) manufacturing strategy, due to limited resources or capacities, may have to reject some orders. From the viewpoint of supply chain management, it has been proven that firms satisfying due dates promised to the customers and shortening lead times will have a competitive advantage. Flow shop scheduling problem with order acceptance decision, the objective function of which is to maximize the total net profit of the accepted orders, has become the epicenter of attention during the past years. Slotnick and Morton (1996, 2007), Ghosh (Ghosh, 1997), Lewis and Slotnick (2002), Rom and Slotnick (2009), and Pourbabai (1989) are among those addressed the flow shop scheduling problem with order acceptance decision and predetermined process times in a single machine (or workstation) environment. It is crystal clear that a production line often consists of multiple stages where the process time of each job in one stage may be different from the one in others. As an illustration, Toyota Motor Corporation's vehicle production system is a multistage lean manufacturing system or a just-in-time (JIT) system that has been established based on many years of continuous improvements with the objective of making and delivering the ordered vehicles in the quickest and most efficient way. Xiao et al. studied the order acceptance problem with weighted tardiness penalties in permutation flow shop scheduling problem, the process of choosing the selected orders and simultaneously scheduling them on a multistage production line to maximize the total net profit of the accepted orders (Xiao, 2012).

In this paper, we address the order acceptance problem with weighted tardiness in permutation flow shop scheduling. The problem is formulated as an integer-programming (IP) model, and a cloud-based simulated annealing (CSA) algorithm is developed to solve the problem. Based on the number of candidate orders the firm receives, fifteen problems with different scales are generated. Each problem is considered an experiment, which is conducted five times to compare the efficiency of the proposed CSA algorithm with the one of a formerly suggested simulated annealing (SA) algorithm for the problem. The experimental results validate the improvement in objective function values yielded by CSA algorithm in comparison with the ones produced by the previously proposed SA algorithm.

2 Literature Review

The publication of optimal two-stage and three-stage production schedules with set-up times by Johnson (2010) aroused considerable interest in the flow-shop scheduling problem. Ribas et al. presented an extensive review of the recently published papers on hybrid flow shop (HFS) scheduling problems. The papers were classified first according to the HFS characteristics and production limitations and second according to the solution approach proposed. Sviridenko (2004) proposed two approximation algorithms for the permutation flow shop problem with make-span objective function. The first algorithm had an absolute performance guarantee, and the second one was an approximation algorithm. Ladhari and Haouari (2005) considered the classical permutation flow shop problem, the scheduling of n jobs through m machines to minimize the make-span. This problem was known to be NP-hard. They presented extensive computational results on both random instances, with up to 8000 operations, and well-known benchmarks with up to 2000 operations, denoting the proposed algorithm solved large-scale instances in moderate CPU time.

Cheng et al. (2001) addressed the three-machine permutation flow-shop scheduling problem with release times to minimize the maximum completion time. Two dominance rules were applied to generating initial schedules, directing the search strategy and decomposing the problem into smaller ones. The proposed branch-and-bound algorithm integrated an adaptive branching rule with a fuzzy search strategy to narrow the search tree and lead the search to an optimal solution as early as possible. Ladhari and Haouari (2005) and Cheng et al. (2001) drew a conclusion that even though the non-permutation schedule provides shorter make-spans than the permutation one, the permutation schedule is a common practice to address the flow-shop scheduling

problem. In contrast to most of the previously published papers assuming that the process time of each job in each stage is predetermined, Niu et al. (2008) and Chanas and Kasperski (2001) addressed the scheduling problem using fuzzy process times and fuzzy due dates.

De et al. (1993) explored a single-machine scheduling problem using random processing times and deadline to select a subset of the jobs and sequence the selected jobs to maximize the expected profit. They presumed an exponentially distributed deadline and did not allow preemption. They described several solution properties, presented dynamic programming (DP) algorithms, and proposed a polynomial time approximation scheme (1993). Based on group technology and JIT manufacturing concepts, Pourbabai (1992) proposed an optimal selection of orders in a JIT manufacturing environment. Two loading models for optimal utilization of the processing capabilities of an integrated manufacturing system were developed to integrate and utilize the available information from both the bill of materials and the process plans. Gupta et al. (1992) addressed the simultaneous selection of a subset of N projects and determination of an optimal sequence to maximize the net present value of the total return. The process consisted of establishing an optimal sequence of all projects, without being dependent on the particular subset of selected projects, and proposing an efficient polynomial DP method to solve the problem.

Slotnick and Morton (1996) maintained that one way of adjusting the workload in a manufacturing facility when available jobs exceed the current capacity was to select a subset of jobs with the objective of maximizing total net profit, revenues minus costs, using weighted lateness as a criterion for time-related penalties. They developed an optimal algorithm and two heuristic procedures. Lewis and Slotnick (2002) elaborated on this problem by examining the profitability of job selection decisions over a number of periods when current orders exceed the capacity with the objective of maximizing profit (net revenue of processing costs minus weighted lateness costs per-job). Besides, they assumed that rejecting a job would adversely affect the future orders from that customer; in other words, they addressed the trade-offs between accepting or rejecting job orders and ensuring timeliness with money-back guarantees.

Ebben et al. (2005) argued that order acceptance and production planning are functionally separated. As a result, order acceptance decisions are made without considering the actual workload in the production system, or by only considering the aggregate workload. They addressed the significance of a good workload-based order acceptance method in over-demanded job shop environments and studied approaches integrating order acceptance with resource capacity loading. Yang and Geunes (2007) examined single-resource scheduling when candidate jobs may be accepted or rejected. Their solution approaches sought to maximize the profitability of the resulting schedule under job-specific tardiness costs and reducible processing times. They presented an algorithm to maximize schedule profit for a given sequence of jobs, along with two heuristic approaches to generate good job sequences. Torabzadeh and Zandieh (2010) addressed a two-stage assembly flow shop problem with m Machines in the first Stage to minimize the weighted sum of the make-span and the mean completion time for n available Jobs. Inasmuch as the problem was NP-hard, they proposed a CSA algorithm to solve it.

Above-mentioned papers fail to address the multistage flow-shop scheduling problem with order acceptance decision at the same time. To the best of our knowledge, there are only two relevant works:

Xiao et al. (2012) studied the order acceptance problem with weighted tardiness penalties in permutation flow shop scheduling, the process of choosing the selected orders and simultaneously scheduling them on a multistage production line with the aim of maximizing the total net profit of the accepted orders. A heuristic algorithm named simulated annealing based on partial optimization (SABPO) was developed for solving the IP model and obtaining near-optimal solutions.

Roundy et al. (2005) proposed a capacity-driven acceptance of customer orders for a multistage batch manufacturing system. When a new order comes in, they look for a feasible schedule to

accommodate it and all of the already accepted orders. They may use multiple production batches to fill the new order; hence, lot-sizing decisions are made as well. They developed a mixed-integer linear programming (MILP) formulation of the order insertion problem, which bypassed detailed Gantt chart manipulations, but guaranteed feasibility.
The prevailing approach to solving the flow-shop scheduling problem with order acceptance decision is to choose a subset of orders from the selected orders and schedule them by taking into account such criteria as total net profit, make-span, and total net present value, among others. Ghosh (1997) proved that one-machine flow-shop scheduling problem with order acceptance decision is NP-hard; consequently, exact algorithms are applied only to small-sized problems. Meta-heuristic algorithms such as genetic algorithm (Rom and Slotnick (2009)), SA (Xiao et al. (2012), Oguz et al. (2010), and Ivanescu et al. (2002, 2006)), CSA (Zandieh and Torabzadeh (2010)), and tabu search (Cesaret et al. (2012)), among others are used for solving medium-sized and large-sized problems.
In this paper, we address the order acceptance problem with weighted tardiness penalties in permutation flow shop scheduling. The problem is formulated as an IP model, and a CSA algorithm is developed to solve the problem. The rest of the paper is organized as the following. In Section 3, we describe the permutation flow-shop scheduling problem with order acceptance and weighted tardiness and formulate it as an IP model. In Section 4, the proposed CSA algorithm and the SABPO algorithm presented by Xiao et al. (2012) are elaborated. In Section 5, computational study on fifteen different problems, along with experimental results are presented. In Section 6, conclusions are drawn based on the results.

3 Problem definition and formulation

MTO is a production strategy in which manufacturing starts only after a customer's order is received. Firms applying MTO, due to limited resources or capacities, may have to reject some orders; as a result, the order acceptance problem with weighted tardiness is of paramount significance to them. The problem is defined as choosing the selected orders and simultaneously scheduling them on a multistage production line to maximize the total net profit of the accepted orders. As Xiao et al. discussed earlier, the production line, comprising a single processing line with multiple stages, is continuously available, and n Orders are processed through m Stages. At the same time, a number of candidate orders from the outside are waiting to be selected. The attributes of each candidate order are a due date, revenue, a tardiness penalty weight (used when an order is delivered later than its due date), and a number of deterministic process times for each stage. The tardiness penalty is proportional to the length of the delay. Finally, the orders are independent from each other, and rejecting (or accepting) an order will not negatively (or positively) impact on other orders. Table 1 shows the notations used in this paper:

Table 1: Notations used for formulating the problem

i	Index of orders, $i = 1,2,\dots,n$
I	Set of orders, $I = \{i \mid 1 \le i \le n\}$
j	Index of stages, $j = 1,2,\dots,m$
k	Index of ranking positions, $k = 1,2,\dots,n$
o_k	The ID of an order in kth position
P_i	Revenue of order i
d_i	Due date of order i
w_i	Tardiness penalty weight of order i
a_{ij}	Process time of order i at stage j
t_{kj}	Completion time of order in kth position at stage j

c_{ij}	Completion time of order i at Stage j

In permutation flow-shop scheduling problem, the processing of Order i at Stage j can be started immediately if and only if:
Its processing at Stage $(j-1)$ has been finished.
The processing of higher-ranked order at Stage j has been completed.
It is crystal clear that there is a unified processing sequence of orders through the stages in a permutation flow-shop system. Table 2 depicts a schematic diagram of a permutation flow-shop system with n Orders and m Stages:

Table 2: Schematic diagram of a permutation flow-shop system with n orders and m stages

		m Stages for Processing Orders			
		Stage 1	Stage 2	...	Stage m
n Ranking positions	Rank 1st	O_1	O_1	O_1	O_1
	Rank 2nd	O_2	O_2	O_2	O_2
	.	.	.	.	.
	.	.	.	.	.
	.	.	.	.	.
	Rank nth	O_n	O_n	...	O_n

Two integer decision variables are defined for the problem. First, the acceptance or rejection variable $\left(Y=\{y_i\}\right)$ and second, the sequence position variable $\left(S=\{s_i\}\right)$. The IP model of the permutation flow-shop scheduling problem with order acceptance decision and weighted tardiness is denoted below:

$$Maximize \quad Z=f(Y,S)=\sum_{i=1}^{n} y_i\left[p_i-w_i\max\left(0,c_{im}-d_i\right)\right] \tag{1}$$

$$Subject \ to: \tag{2}$$

$$y_i\in\{0,1\},\quad \forall i \tag{3}$$

$$1\le s_i\le n,\quad \forall i \tag{4}$$

$$s_i\neq s_{i'},\quad \forall i\neq i' \tag{5}$$

$$t_{kj}=\begin{cases} y_i a_{i1} & \forall j=1,\quad k=1,\quad i=o_1 \\ t_{k-1,1}+y_i a_{i1} & \forall j=1,\quad k>1,\quad i=o_k \\ y_i c_{i,j-1}+y_i a_{ij} & \forall k=1,\quad j>1,\quad i=o_1 \\ \max\left\{y_i c_{i,j-1},t_{k-1,j}\right\}+y_i a_{ij} & \forall k>1,\quad j>1,\quad i=o_k \end{cases} \tag{6}$$

$$c_{ij}=t_{kj},\quad \forall k\ge 1,\quad j\ge 1,\quad i=o_k \tag{7}$$

$$s_i=k,\quad \forall i=o_k$$

4 Proposed meta-heuristic algorithm

In this section, we first review the partial optimization strategy and the SABPO algorithm proposed by Xiao et al. (2012). We then present the basic concepts of cloud theory, which paves the way for introducing our proposed CSA algorithm for the permutation flow shop scheduling problem with order acceptance decision and weighted tardiness penalties.

4.1 Partial optimization (PO) strategy

Since the objective function $\left(f(Y,S)\right)$ includes two independent decision variables, the prevailing approach to addressing the problem is to employ a PO strategy, which in turn optimizes one of the variables while fixes the other. Although this strategy has its merits, it only accepts solutions being better than the incumbent solution, which is prone to be trapped in a local optimum. Besides, the quality of the solution is dependent on the initial solution and which variable is optimized first. The following are the steps of PO strategy:

- Find an initial solution as the incumbent solution.
- Fix S and find Y to maximize $f(Y,S)$.
- Fix Y and find S to maximize $f(Y,S)$.
- Iterate the second and third steps until $f(Y,S)$ cannot be improved anymore.

4.2 Simulated annealing (SA)

The SA algorithm, first proposed by Kirkpatrick et al. (1983), is a random search to find the optimal solution in stochastic combinatorial optimization problems. It is characterized by allowing hill climbing moves to escape the local optima and find global optimal solutions if the cooling schedule is slow enough. These approaches are based on the physical concepts of increasing temperature to reach a high value followed by a gradual cooling process and finally reaching to a state of a minimum potential energy. This improvement mechanism consists of two phases:

- Heating phase: an initial solution is set as the incumbent solution. The algorithm repeatedly changes the incumbent solution, and the maximum deviation of the objective function between two neighboring solutions is used as the initial temperature for the next phase.
- Cooling phase: new solutions are repeatedly generated based on the incumbent solution as the temperature is decreased according to a cooling schedule. These solutions are evaluated according to the Metropolis rule, introduced by Metropolis et al. (1953), to determine whether or not a new solution should be accepted as the new incumbent solution. This phase continues until the computational temperature drops to a predetermined threshold. The Metropolis rule used in SA for a maximizing objective function is as the following:

$$P\left(X_{old} \to X_{new}\right) = \begin{cases} 1 & \Delta f > 0 \\ e^{\frac{\Delta f}{T}} & otherwise \end{cases} \qquad (8)$$

In equation (8), $\Delta f = f\left(X_{new}\right) - f\left(X_{old}\right)$ is the deviation of the objective function between two neighboring solutions, $P\left(X_{new} \to X_{old}\right)$ is the probability of accepting a new solution, and T is the computational temperature.

4.3 Simulated annealing based-on partial optimization (SABPO) algorithm

According to the Metropolis rule, the SABPO algorithm proposed by Xiao et al. (2012) will accept a worse solution with a probability that decreases as the temperature drops. As a result,

SABPO, unlike the PO strategy, is capable of jumping out of local optima to find a globally optimal solution. The parameters of SABPO algorithm are defined in Table 2. The SABPO algorithm proposed by Xiao et al. (2012) is elaborated in Figure 1:

Table 2: The parameters of SABPO algorithm

τ	Drop speed of the temperature, $\tau \in (0,1)$
$EndT$	Threshold temperature for terminating the algorithm
N	Number of changing the incumbent solution at each temperature level
R_Y	The accepting rate (the ratio of the number of improvements to the number of attempts at one temperature level) when optimizing Y
R_S	The accepting rate when optimizing S
P_Y	Number of total attempts for optimizing Y while S is fixed: $P_Y = N\left(0.2 + 0.6 \times \frac{R_Y}{R_Y + R_S}\right)$ (9)
P_S	Number of total attempts for optimizing S while Y is fixed: $P_S = N\left(0.2 + 0.6 \times \frac{R_S}{R_Y + R_S}\right)$ (10)
T_0	Initial temperature; maximum deviation of the objective function between two neighboring solutions

Inputs: $\tau, EndT, N$

Outputs: Optimal (Y,S)

Begin

%% Initialization

1) Randomly generate decision variables; $Y^0 = randi([0,1],1,n), S^0 = randsample(n,n)$

2) Set (Y^0,S^0) as the incumbent solution; $Y = Y^0, S = S^0$

3) Randomly change Y and S for N times. Set the Maximum Deviation of the Objective Function as the initial temperature (T_0).

4) $T = T_0, P_Y = 0.05 \times N, P_S = 0.05 \times N$

%% Main loop of SABPO algorithm

5) While $T > EndT$

%% Cooling phase (dropping the temperature)

6) $T = T \cdot \tau$

%% Optimizing Y

7) For $k = 1$ *to* P_Y

8) $Y_{new} = Function(Y_{old}), Z_{new} = Function(Y_{new}, S)$

9) Calculate the Metropolis probability (P) according to Equation (8).

10) Generate a random value drawn from standard uniform distribution $(0,1)$.

If $rand(1) > p$, Y_{new} is then accepted and saved. Otherwise it is rejected.

11) End

%% Optimizing S

12) For $k=1$ *to* P_S

13) $S_{new} = Function(S_{old}), Z_{new} = Function(Y, S_{new})$

14) Calculate the Metropolis probability (P) according to Equation (8).

15) Generate a random value drawn from standard uniform distribution $(0,1)$.

If $rand(1) > p$, S_{new} is then accepted and saved. Otherwise it is rejected.

16) End

17) Recalculate P_Y and P_S according to Equation (9) and Equation (10), respectively.

18) End

19) Return the incumbent solution (Y,S) and its objective function value.

End

Figure 1. The SABPO algorithm proposed by Xiao et al. [8]

4.4 Solution representation, functions, and operators

Table 3 displays one possible solution for a problem with ten orders $(n=10)$. It consists of two rows of numbers, the first one refers to the acceptance or rejection of the orders, and the second one denotes the processing sequence of orders through the stages; that is, orders 3, 6, and 7 are rejected, and orders 8 and 9, among others are the first two ones processed through the stages, respectively:

Table 3: Solution representation of a problem with ten orders $(n=10)$

y_i	1	1	0	1	1	0	0	1	1	1
s_i	8	9	2	10	3	7	4	1	5	6

Three main functions being described in Figure 1 are as the following:

- $Y_{new} = Function(Y_{old})$: This function randomly generates an integer between 1 and n, which specifies the corresponding gene of the Y_{old} chromosome to be changed. The content will become 1 if it is 0, and vice versa.
- $S_{new} = Function(S_{old})$: Three different operators are defined to generate a new order sequence (S_{new}). The swap operator randomly generates two integers between 1 and n, which specify the corresponding genes of the S_{old} chromosome to be changed. The genes will be mutually exchanged. Table 4 illustrates the procedure for getting a new order sequence by swap operator. The insert operator randomly generates two integers between 1 and n, which specify the corresponding genes of the S_{old} chromosome to be changed. One gene will be inserted into another gene; it is placed before that gene. Table 5 illustrates the procedure for getting a new order sequence by insert operator. The reverse operator randomly generates four integers between 1 and n, which specify the corresponding genes of the S_{old} chromosome to be changed. Each pair genes will be mutually exchanged. Table 7 illustrates the procedure for getting a new order sequence by reverse operator.

Table 4: Swap operator

6	3	7	8	5	1	2	4	9	10
$i1=2$ & $i2=9$									
6	9	7	8	5	1	2	4	3	10

Table 5: Insert operator

4	10	3	7	9	8	1	5	6	2
$i1=1 \quad \& \quad i2=8$									
5	4	10	3	7	9	8	1	6	2

Table 6: Reverse operator

6	1	4	8	7	3	9	10	2	5
$i1=1, \quad ii1=3, \quad ii2=7, \quad i2=10$									
5	1	9	8	7	3	4	10	2	6

- $Z_{new} = Function(Y_{new}, S_{new})$: This function computes the objective function of the IP model, denoted in the mathematical model, according to the constraints and is elaborated in Figure 2:

Inputs: Y, S
Outputs: Total Net Profit
Begin
1) Sets $S = \{s_i\}$ and $O = \{o_k\}$ are calculated according to Equation (7).
%% Constraints pertinent to Stage 1
2) $t_{11} = y_i a_{i1}, \quad i = o_1$
3) For $k = 2 \quad to \quad n$
4) $t_{k1} = t_{k-1,1} + y_i a_{1i}, \quad i = o_k$
5) End
%% Constraints pertinent to Stages 2 to m
6) For $j = 2 \quad to \quad m$
7) $t_{1j} = c_{i,j-1} + y_i a_{ij}, \quad i = o_1$
8) For $k = 2 \quad to \quad n$
9) $t_{kj} = y_i a_{kj} + \max\{y_i c_{i,j-1}, t_{k-1,j}\}, \quad i = o_k$
10) End
11) End
12) For $j = 1 \quad to \quad m$
13) For $k = 1 \quad to \quad n$
14) $c_{ij} = t_{kj}, \quad i = o_k$
15) End
16) End
17) Calculate total net profit according to Equation (1).
18) Return the total net profit.
End

Figure 2: Function for calculating total net profit

4.5 Basic concepts of cloud theory

The cloud theory utilizes natural language to transfer uncertainty between quality concept and quantity data representation. The randomly changing nature of the annealing temperature diversifies and expedites searching of the neighborhood as well as avoids being trapped in the local optimum (Deyi et al. 1995). Cloud theory is an expansion of membership function in fuzzy theory, which guarantees both characteristics of randomness and stability (Deyi & Yi 2005). It has many applications in such areas as intelligence control (Deyi et al. (1998) and Feizhou et al. (1999)), knowledge representation (Cheng et al. (2005) and Deyi et al. (2000)), and data mining (Kaichang et al. (1999), Kaichang et al. (1998), Shuliang et al. (2003), and Yingjun and Zhongying (2004)), among others.

Let F be the language value of domain d and mapping $C_F(X)$ as the following:

$$C_F(X): d \to [0,1], \quad \forall x \in d, \quad x \to C_F(X) \tag{11}$$

If the distribution of $C_F(X)$ is normal, it is called a normal cloud model. This map generates a group of random numbers, reflecting quantitative characteristics of $C_F(X)$, that are distinguished by the following elements: [30]

- Expectation (Ex): determines the center of the cloud.
- Entropy (En): specifies the range of the cloud.
- Super entropy (He): determines the degree of the dispersion of the cloud drops.

Figure 3 depicts the procedure of generating a cloud drop $\left(drop(x_i, d_0)\right)$ using $\{Ex, En, He\}$ (Deyi & Yi 2005):

Inputs: $\{Ex, En, He\}, n, d_0$
Outputs: $\{(x_1, d_0), (x_2, d_0), \rightleftharpoons, (x_n, d_0)\}$
Begin
1) For $i = 1$ *to* n
2) $En' = randn(En, He)$
3) $x_i = Ex \pm En' \sqrt{-2 \times \ln(d_0)}$
4) End
End

Figure 3. Generating a cloud drop

4.6 Proposed Cloud-based Simulated Annealing (CSA) algorithm

All the required parameters for introducing our proposed CSA algorithm have been previously defined in Table 2. A suffix 'cloud' is added to roughly all variables and parameters to differentiate this algorithm from SABPO. Our proposed CSA algorithm is elaborated in Figure 4. At high temperatures, drop dispersion increases, and the range of the annealing temperature widens; hence, it ensures the randomness of annealing process. However, the opposite is the case at low temperatures (drop dispersion decreases, and the range of the annealing temperature narrows), which ensures the stable tendency of annealing process (Torabzadeh & Zandieh, 2010). In this paper, when it comes to using P_Y and P_S to compare the efficiency of CSA

algorithm to the one of SABPO, these parameters are fixed at $0.05 \times N$, unlike the Equation (9) and Equation (10) proposed by Xiao et al. (2012). It will not make the results invalid inasmuch as the same condition is used for comparison. Besides, the procedure to use swap, insert, and reverse operators is different from the one offered by Xiao et al. (2012). All functions are calculated according to the rules described in section 4.4. The following show the mechanism for selecting the right operator for both algorithms, which will not question the validity of the comparison:

- ✓ Generate a random value (b) from standard uniform distribution $(0,1)$.
- ✓ The selection mechanism is: $\left\{ \begin{array}{lll} if & 0 < b \le 0.33 \rightarrow & SWAP \\ if & 0.33 < b \le 0.66 \rightarrow & INSERT \\ if & 0.66 < b < 1 \rightarrow & REVERSE \end{array} \right\}$

Inputs: $\lambda, EndT_{cloud}, N$

Outputs: Optimal (Y_{cloud}, S_{cloud})

Begin

1) Randomly generate decision variables; $Y_{cloud}^0 = randi([0,1],1,n), S_{cloud}^0 = randsample(n,n)$

2) Set $(Y_{cloud}^0, S_{cloud}^0)$ as the incumbent solution; $Y = Y_{cloud}^0, S = S_{cloud}^0$

3) Randomly change Y and S for N times. Set the Maximum Deviation of the Objective Function as the initial temperature (T_{cloud}^0).

4) $T_{cloud} = T_{cloud}^0, P_Y = 0.05 \times N, P_S = 0.05 \times N$

5) $Ex(0) = T_{cloud}^0$, $En(0) = 0.1 \times T_{cloud}^0$

%% Main loop of CSA algorithm

6) While $T_{cloud} > EndT_{cloud}$

%% Dropping the mean and the variance of the temperature

7) $Ex(\alpha) = Ex(0) \cdot \lambda^{\alpha}$

8) $En(\alpha) = En(0) \cdot \lambda^{\alpha}$

%% Optimizing Y

9) For $k = 1$ *to* P_Y

10) $Y_{cloud}^{new} = Function(Y_{cloud}^{old}), Z_{cloud}^{new} = Function(Y_{cloud}^{new}, S_{cloud})$

%% Generating a cloud drop

11) $T_{k\alpha} = Ex(\alpha) \pm 3(En(\alpha) + \varepsilon_k)$, $\varepsilon \sim rand(-1,+1)$

12) Calculate the Metropolis probability (P) according to Equation (8).

Use $T_{k\alpha}$ instead of T as denominator.

13) Generate a random value drawn from standard uniform distribution $(0,1)$.

If $rand(1) > p$, Y_{cloud}^{new} is then accepted and saved. Otherwise it is rejected.

14) End

%% Optimizing S

15) For $k=1$ *to* P_S

16) $S_{cloud}^{new} = Function\left(S_{cloud}^{old}\right), Z_{cloud}^{new} = Function\left(Y_{cloud}, S_{cloud}^{new}\right)$

17) Use the step 11 to generate a cloud drop.

18) Calculate the Metropolis probability (P) according to Equation (8).

Use $T_{k\alpha}$ instead of T as denominator.

19) Generate a random value drawn from standard uniform distribution $(0,1)$.

If $rand(1) > p$, S_{cloud}^{new} is then accepted and saved. Otherwise it is rejected.

20) End

21) Recalculate P_Y and P_S according to Equation (9) and Equation (10), respectively.

22) $T_{cloud} = T_{cloud}^{0} \times \lambda^{\alpha}$

23) End

24) Return the incumbent solution (Y,S) and its objective function value.

End

Figure 4: The proposed CSA algorithm

5 Computational study and experimental results

In this section, we carry out computational experiments to compare the efficiency of our proposed CSA algorithm to the one of SA algorithm formerly suggested by Xiao et al. [8] for order acceptance problem with weighted tardiness penalties in permutation flow shop scheduling.

5.1 Data settings

Based on the number of candidate orders a firm receives, fifteen different problems are generated. Each problem is regarded as an experiment and run five times. The number of stages (m) is fixed at 10. Other values are generated according to the rules introduced by Xiao et al. (2012), which are given in Table 7.

Table 7: Values for the parameters of both algorithms

Data values
$n = 25, 50, 75, 90, 100, 110, 125, 140, 150, 160, 175, 190, 200, 225, 250$
$m = 10$
Experiments: I, II, III, IV, V
$\tau = \lambda = 0.99$
$EndT = EndT_{cloud} = 0.1$
$N = 100 \times n$
$P_Y = P_S = 0.05 \times N$
Orders' revenues: random integer on $[1, 500]$
Orders' due dates: random integer on $[1, n \times m \times 40]$
Orders' weights: random value on $[0.5, 1.5]$
Orders' process times at different stages: random integer on $[1, 100]$

5.2 Experimental results and comparisons

We perform computational experiments on a laptop with a 2.9GHz Dual Core i7 (Turbo Boost up to 3.6GHz) and 8GB of RAM. Both SABPO and CSA algorithms are coded in MATLAB and run on Macintosh Operating System. The results are shown in Table 8. A number of observations are emerged from the results:

- In 60% of the experiments, CSA algorithm yields higher average optimal solution (OS) values than those of SABPO algorithm. In 33.3% of the experiments, both algorithms produce the same average OS values. Only for $n = 225$ (6.7% of the experiments), CSA algorithm yields a lower average OS value than that of SABPO algorithm.
- In 68% of the experiments, CSA algorithm gains the best solution sooner than the time of SABPO algorithm. In 32% of the experiments, SABPO algorithm reaches to the best solution in a shorter time in comparison to that of CSA algorithm.

Figure 5 shows how two algorithms reach the best solution for $n = 175, I$. It is crystal clear that CSA algorithm is superior to SABPO algorithm; the final best solution of CSA algorithm is considerably higher than that of SABPO algorithm.

- OS: Optimal Solution
- TGBS : Time to Get the Best Solution (in second)
- TTRA : Total Time of Running the Algorithm (in second)
- NRO : Number of Rejected Orders

Table 8: Experimental results

$n = 25$	SABPO					CSA				
	I	*II*	*III*	*IV*	*V*	*I*	*II*	*III*	*IV*	*V*
OS	7657.4	7657.4	7657.4	7657.4	7657.4	7657.4	7657.4	7657.4	7657.4	7657.4
TGBS	8	12	7	8	3	6	6	1	1	2
TTRA	116	117	114	115	114	152	153	154	153	155
NRO	1	1	1	1	1	1	1	1	1	1
$n = 50$	SABPO					CSA				
	I	*II*	*III*	*IV*	*V*	*I*	*II*	*III*	*IV*	*V*
OS	11815	11815	11815	11814	11815	11815	11815	11815	11815	11815
TGBS	113	122	116	300	34	76	33	47	40	63
TTRA	424	418	418	417	417	544	499	501	499	497
NRO	0	0	0	1	0	0	0	0	0	0
$n = 75$	SABPO					CSA				
	I	*II*	*III*	*IV*	*V*	*I*	*II*	*III*	*IV*	*V*
OS	17679	17679	17679	17679	17679	17679	17679	17679	17679	17679
TGBS	154	74	99	230	109	115	71	55	90	89
TTRA	883	936	945	881	948	995	1058	1066	1005	1049
NRO	2	2	2	2	2	2	2	2	2	2
$n = 90$	SABPO					CSA				
	I	*II*	*III*	*IV*	*V*	*I*	*II*	*III*	*IV*	*V*
OS	23991	23932	23991	23991	23991	23991	23991	23991	23991	23991
TGBS	566	97	1321	647	317	237	221	871	191	300
TTRA	1378	1332	1345	1429	1325	1514	1486	1538	1545	1500
NRO	1	2	1	1	1	1	1	1	1	1
$n = 100$	SABPO					CSA				
	I	*II*	*III*	*IV*	*V*	*I*	*II*	*III*	*IV*	*V*
OS	26500	26500	26500	26500	26500	26500	26500	26500	26500	26500

TGBS	240	319	304	355	177	153	140	182	133	99
TTRA	1603	1593	1661	1584	1598	1752	1747	1856	1756	1743
NRO	0	0	0	0	0	0	0	0	0	0
$n=110$	SABPO					CSA				
	I	*II*	*III*	*IV*	*V*	*I*	*II*	*III*	*IV*	*V*
OS	28721	28721	28672	28721	28721	28721	28721	28721	28721	28721
TGBS	6	218	204	1022	505	403	207	469	408	257
TTRA	2231	2010	2019	2066	1990	2225	2190	2222	2243	2157
NRO	0	0	1	0	0	0	0	0	0	0
$n=125$	SABPO					CSA				
	I	*II*	*III*	*IV*	*V*	*I*	*II*	*III*	*IV*	*V*
OS	33189	33189	33189	33175	33175	33189	33189	33189	33189	33189
TGBS	764	744	857	395	860	700	1612	897	680	480
TTRA	2769	2743	2761	2755	2794	2921	2974	3034	3031	2995
NRO	3	3	3	3	3	3	3	3	3	3
$n=140$	SABPO					CSA				
	I	*II*	*III*	*IV*	*V*	*I*	*II*	*III*	*IV*	*V*
OS	34259	34259	34259	34259	34259	34259	34259	34259	34259	34259
TGBS	909	499	1160	411	531	437	474	414	1075	773
TTRA	3221	3281	3339	3292	3215	3483	3586	3506	3477	3438
NRO	0	0	0	0	0	0	0	0	0	0
$n=150$	SABPO					CSA				
	I	*II*	*III*	*IV*	*V*	*I*	*II*	*III*	*IV*	*V*
OS	39464	39456	39464	39456	39464	39464	39464	39464	39464	39464
TGBS	1442	3846	2826	1743	2228	853	2276	1269	1620	2143
TTRA	3874	3908	3882	3924	4107	4106	4162	4343	4273	4257
NRO	1	2	1	2	1	1	1	1	1	1
$n=160$	SABPO					CSA				
	I	*II*	*III*	*IV*	*V*	*I*	*II*	*III*	*IV*	*V*
OS	39085	39196	39085	39145	39085	39085	39196	39196	39085	39196
TGBS	849	798	983	1330	240	624	880	521	754	1137
TTRA	4301	4305	4325	4340	4332	4638	4573	4658	4631	4740
NRO	1	0	1	0	1	1	0	0	1	0
$n=175$	SABPO					CSA				
	I	*II*	*III*	*IV*	*V*	*I*	*II*	*III*	*IV*	*V*
OS	42938	43317	43321	43321	43321	43321	43301	43277	43321	43321
TGBS	3159	4666	950	2198	3068	1240	5565	1986	2321	2398
TTRA	5580	5573	5650	5494	5478	5807	5987	5903	5750	5780
NRO	3	2	1	1	1	1	2	2	1	1
$n=190$	SABPO					CSA				
	I	*II*	*III*	*IV*	*V*	*I*	*II*	*III*	*IV*	*V*
OS	42920	42926	42934	42934	42934	42934	42934	42934	42934	42920
TGBS	5300	2721	4347	3241	4624	1643	1814	1187	1448	5300
TTRA	6268	6516	6315	6248	6413	6584	6669	6565	6597	6268
NRO	2	2	1	1	1	1	1	1	1	2
$n=200$	SABPO					CSA				
	I	*II*	*III*	*IV*	*V*	*I*	*II*	*III*	*IV*	*V*
OS	49419	49420	49422	49420	49418	49422	49422	49422	49422	49422

TGBS	6315	2708	4816	2715	3662	1114	4159	5781	5994	1925
TTRA	7136	7003	6917	6988	7073	7333	7374	7346	7303	7476
NRO	2	1	0	1	1	0	0	0	0	0
$n = 225$	SABPO					CSA				
	I	*II*	*III*	*IV*	*V*	*I*	*II*	*III*	*IV*	*V*
OS	55635	55586	55624	55607	55609	55630	55615	55631	55587	55595
TGBS	3154	6999	7686	5186	2587	3898	2261	4245	4583	7885
TTRA	9503	9269	9427	9403	9221	9729	9696	9705	9755	9751
NRO	0	2	1	2	1	1	1	1	4	2
$n = 250$	SABPO					CSA				
	I	*II*	*III*	*IV*	*V*	*I*	*II*	*III*	*IV*	*V*
OS	61943	61943	61943	61943	61943	61943	61943	61943	61943	61943
TGBS	2584	4315	10877	3429	6783	3273	5704	3088	6112	7295
TTRA	11480	11576	11945	11665	11853	12210	12067	12595	12278	12450
NRO	0	0	0	0	0	0	0	0	0	0

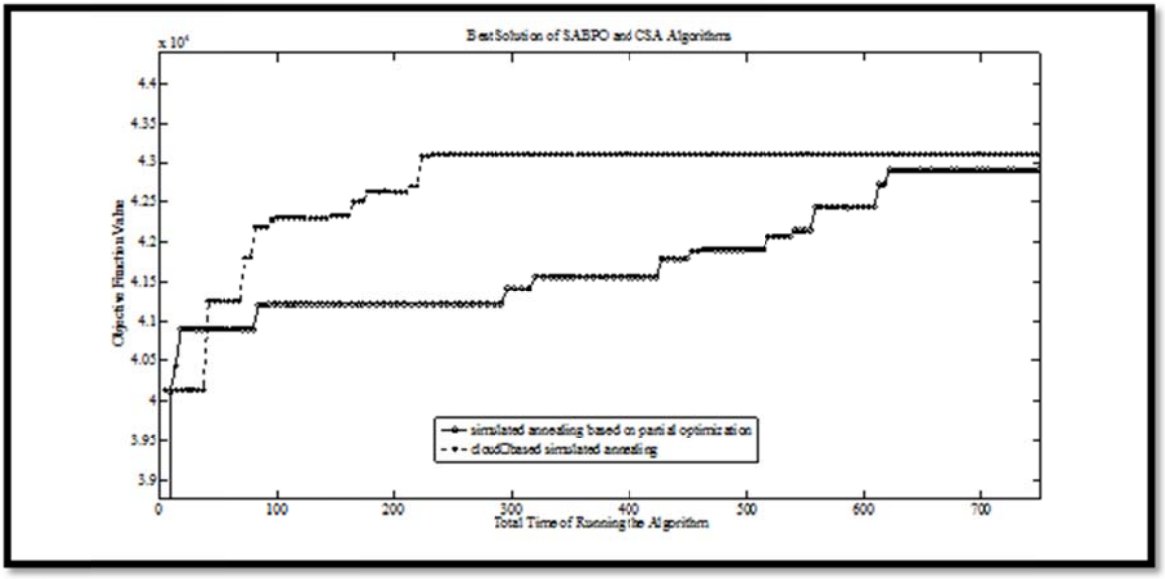


Figure 5: Best solution of SABPO and CSA algorithms for $n = 175, I$

For a better and more logical comparison of the efficiency of both algorithms, an index called relative percentage deviation (RPD) is calculated for all the experimental results according to Equation (12), where $A\lg_{sol}$ is the objective function value of a given algorithm, and Min_{sol} is the best value of objective function value between both algorithms. It is obvious that lower RPD values are preferable (Torabzadeh & Zandieh, 2010):

$$RPD = \frac{A\lg_{sol} - Min_{sol}}{Min_{sol}} \cdot 100\% \qquad (12)$$

In order to test the efficiency of our proposed CSA algorithm to the one of SA algorithm formerly suggested by Xiao et al. (2012), hypothesis testing of difference between two population means (independent samples and unknown population variances, t-test for objective

function (OF) values and computational times) is carried out according to Equation (13) and Equation (14):

$$\begin{cases} H_0: & \mu_{OF}(SABPO) = \mu_{OF}(CSA) \\ H_1: & \mu_{OF}(SABPO) \neq \mu_{OF}(CSA) \end{cases} \quad (13)$$

$$\begin{cases} H_0: & \mu_{Computational-Time}(SABPO) = \mu_{Computational-Time}(CSA) \\ H_1: & \mu_{Computational-Time}(SABPO) \neq \mu_{Computational-Time}(CSA) \end{cases} \quad (14)$$

This hypothesis testing is carried out by SPSS. Table 9 and Table 10 indicate the results of hypothesis testing for OF values. Since $significance\ \ level = 0.01 < 0.05$, the null hypothesis is rejected; in other words, the difference between two population means is significant and $\mu_{OF}(SABPO) \neq \mu_{OF}(CSA)$. Inasmuch as CSA's average RPD $\left(\overline{RPD_{CSA}} = 0.018688\right)$ is lower than that of SABPO's $\left(\overline{RPD_{SABPO}} = 0.083373\right)$, the CSA algorithm produces higher OF values than that of SABPO algorithm. Table 11 and Table 12 indicate the results of hypothesis testing for computational times. Since $significance\ \ level = 0.061 > 0.05$, we fail to reject the null hypothesis; that is, there is insufficient evidence to make a conclusion about the inequality of two population means and $\mu_{Computational-Time}(SABPO) = \mu_{Computational-Time}(CSA)$. Hence, we conclude that the CSA algorithm is not superior to SABPO algorithm when it comes to comparing the time to get the best solution. Figure 6 and Figure 7, showing 95% confidence interval plots for SABPO's and CSA's OF values and computational times, testify to the conclusions made based on SPSS outputs.

Table 9: Hypothesis testing for OF values

	N	Mean	Std. Deviation	Std. Error	95% Confidence Interval for Mean	
					Lower Bound	Upper Bound
SABPO	75	.0834	.2058	.0238	.0360	.1307
CSA	75	.0187	.0592	.0068	.0051	.0323
Total	150	.0510	.1544	.01260	.0261	.0759

Table 10: Analysis of Variance (ANOVA) for OF values

	Sum of Squares	Degree of freedom	Mean Square	F	Sig.
Between Groups	.157	1	.157	6.842	.010
Within Groups	3.394	148	.023		
Total	3.551	149			

Table 11: Hypothesis testing for computational times

	N	Mean	Std. Deviation	Std. Error	95% Confidence Interval for Mean	
					Lower Bound	Upper Bound
SABPO	75	566.4412	2204.2086	254.5201	59.2989	1073.5836
CSA	75	86.1645	114.4874	13.2199	59.8234	112.5057
Total	150	326.3029	1574.0162	128.5179	72.3498	580.2559

Table 12: ANOVA for computational times

	Sum of Squares	DF	Mean Square	F	Sig.
Between Groups	8649964.263	1	8649964.263	3.551	.061
Within Groups	360501574.313	148	2435821.448		

Total	369151538.576	149			

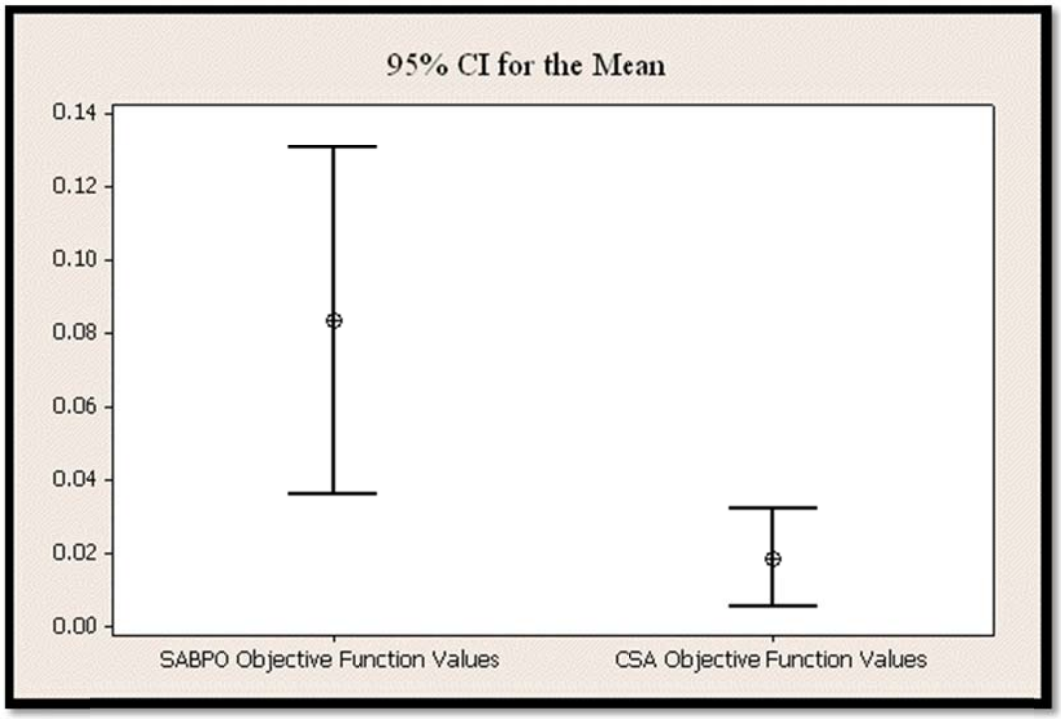


Figure 6: 95% Confidence interval for SABPO's and CSA's OF values

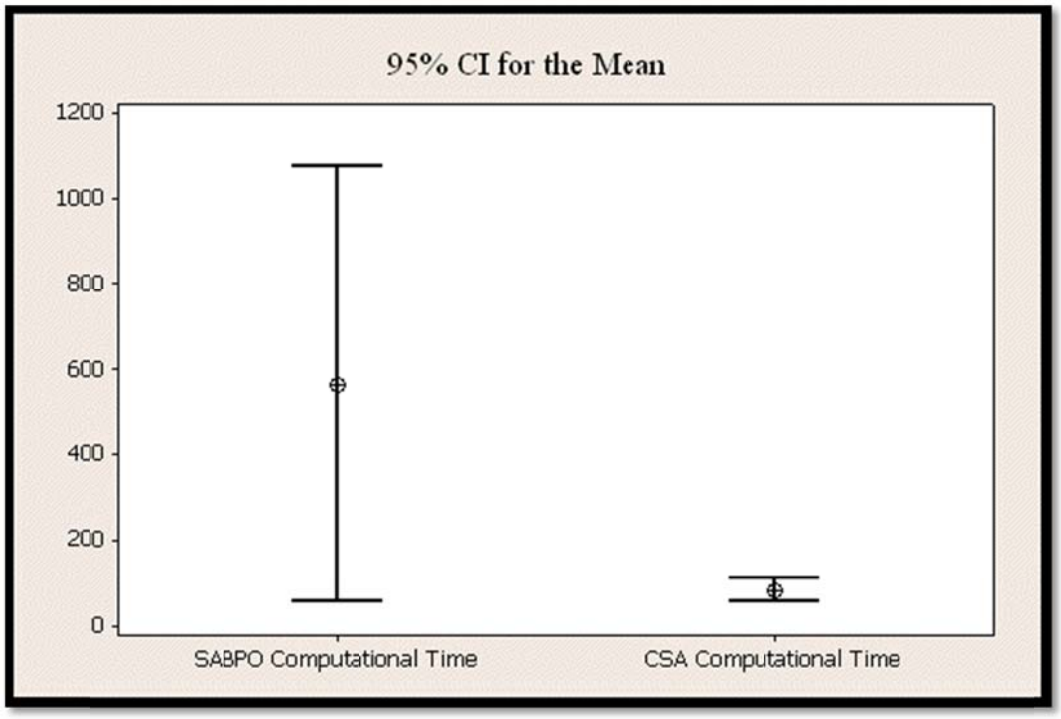


Figure 7: 95% Confidence interval for SABPO's and CSA's computational times

6 Conclusions and future work

In this paper, we address a multistage flow-shop scheduling problem with order acceptance decision and weighted tardiness penalties. The problem is formulated as an IP model, and a CSA

algorithm is developed to solve the problem. Computational study, which consists of generating fifteen different problems based on the number of candidate orders a firm receives, is carried out to evaluate the efficiency of the proposed algorithm. The results denote that our proposed CSA algorithm outweigh the formerly SABPO algorithm suggested by Xiao et al. (2012) in terms of producing better objective function values. Future studies may include developing a more efficient and effective algorithm for the problem or formulating the problem in a non-permutation flow-shop environment.

References

1. Deyi L., Kaichang D., Deren L., 2000, Knowledge representation and uncertainty reasoning in GIS based on cloud theory, Proceedings of the 9th international symposium on spatial data handling:3–14.
2. Cesaret B., Og˘uz C., Salman F.S., 2012, A Tabu Search algorithm for order acceptance and scheduling, Computers & Operations Research 39:1197–1205.
3. Chanas S., Kasperski A., 2001, Minimizing maximum lateness in a single machine scheduling problem with fuzzy processing times and fuzzy due dates, Engineering Applications of Artificial Intelligence 143:377–386.
4. Cheng J., Steiner G., Stephenson P., 2001, A computational study with a new algorithm for the three-machine permutation flow-shop problem with release times, European Journal of Operational Research 130:559–575.
5. Cheng T., Li Z., Deng M., Xu Z., 2005, Representing indeterminate spatial objects by cloud theory, Proceedings of the 4th international symposium on spatial data quality:70–77.
6. De P., Ghosh J.B., Wells C.E., 1993, Job selection and sequencing on a single-machine in a random environment, European Journal of Operational Research 70:425–431.
7. Deyi L., Cheung D., Xuemei S., Ng V., 1998, Uncertainty reasoning based on cloud models in controllers, Computers and Mathematics with Applications 35:99–123.
8. Deyi L., Haijun M., Xuemei S., 1995,Membership clouds and membership cloud generators, Journal of Computer Research and Development 32:15–20.
9. Deyi L., Yi D., 2005, Artificial intelligence with uncertainty, Chapman and Hall.
10. Du J., Leung J.Y.-T., 1990, Minimizing total tardiness on one machine is NP-hard, Mathematics of Operations Research 15:483–495.
11. Ebben M.J.R., Hans E.W., Olde Weghuis F.M., 2005, Workload based order acceptance in job shop environments, OR Spectrum 27:107–122.
12. Feizhou Z., Yuezu F., Chengzhi S., Deyi L., 1999, Intelligent control based membership cloud generators, Acta Aeronaut Astronaut Sinica, 20:89–92.
13. Ghosh J.B., 1997 Job selection in a heavily loaded shop, Computers & Operations Research 24:141–145.
14. Gupta S.K., Kyparisis J., Ip C. M., 1992, Note—Project selection and sequencing to maximize net present value of the total return, Management Science 38:751–752.
15. Iva˘nescu V.C., Fransoo J.C., Bertrand J.W.M., 2002, Make-span estimation and order acceptance in batch process industries when processing times are uncertain, OR Spectrum 24:467–495.
16. Iva˘nescu V.C., Fransoo J.C., Bertrand J.W.M., 2006, A hybrid policy for order acceptance in batch-process industries, OR Spectrum 28:199–222.
17. Johnson S.M., 1954 Optimal two-and three-stage production schedules with set up times included, Naval Research Logistics Quarterly 1:61–68.
18. Kaichang D., Deyi L., Deren L., 1998, Knowledge representation and discovery in spatial databases based on cloud theory, International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences, 32:544–551.
19. Kaichang D., Deyi L., Deren L., 1999, Cloud theory and its applications in spatial data mining knowledge discovery, Journal of Image and Graphics, 4:930–935.
20. Kirkpatrick S., Gelatt Jr. C.D., Vecchi M.P., 1983, Optimization by simulated annealing, Science 220: 671–680.

21. Ladhari T., HaouariM. , 2005, A computational study of the permutation flow shop problem based on a tight lower bound, Computers & Operations Research 32:1831–1847.
22. Lewis H.F., Slotnick S.A., 2002, Multi-period job selection: planning work loads to maximize profit, Computers & Operations Research 29:1081–1098.
23. Metropolis N., Rosenbluth A., Teller A., Teller E., 1953, Equation of state calculations by fast computing machines, The Journal of Chemical Physics 21:1087–1092.
24. Niu Q., Jiao B., Gu X., 2008, Particle swarm optimization combined with genetic operators for job shop scheduling problem with fuzzy processing time, Applied Mathematics and Computation 205:148–158.
25. Og˘uz C., Salman F.S., Yalçın Z.B., 2010, order acceptance and scheduling decisions in make-to-order systems, International Journal of Production Economics 125:200–211.
26. Potts C.N. , Shmoys D.B., Williamson D.P., 1991 Permutation vs. non-permutation flow shop schedules, Operations Research Letters 10: 281–284.
27. Pourbabai B., 1989, A short term production planning and scheduling model, Engineering Costs and Production Economics 18:159–167.
28. Pourbabai B., 1992, Optimal selection of orders in a just-in-time manufacturing environment: a loading model for a computer integrated manufacturing system, International Journal of Computer Integrated Manufacturing 5:38–44.
29. Ribas I., Leisten R., Framiñan J.M., 2010, Review and classification of hybrid flow-shop scheduling problems from a production system and a solutions procedure perspective, Computers & Operations Research 37:1439–1454.
30. Rom W.O., Slotnick S.A., 2009, Order acceptance using genetic algorithms, Computers & Operations Research 36:1758–1767.
31. Roundy R., Chen D., Chen P., Cakanyildirim M., Freimer M.B., Melkonian V., 2005, Capacity-driven acceptance of customer orders for a multi-stage batch manufacturing system: models and algorithms, IIE Transactions 37:1093–1105.
32. Shuliang W., Deren L., Wenzhong S., Deyi L., Xinzhou W., 2003, Cloud model-based spatial data mining, Geographic Information Sciences, 9:67–78.
33. Slotnick S.A., Morton T.E., 1996, Selecting jobs for a heavily loaded shop with lateness penalties, Computers & Operations Research 23: 131–140.
34. Slotnick S.A., Morton T.E., 2007, Order acceptance with weighted tardiness, Computers & Operations Research 34: 3029–3042.
35. Sviridenko M. I.,2004, A note on permutation flow shop problem, Annals of Operations Research 129:247–252.
36. Torabzadeh E., Zandieh M., 2010, Cloud-theory based simulated annealing approach for scheduling in the two-stage assembly flow shop, Advances in Engineering Software 41:1238-1243.
37. Xiao Y.Y., 2012, Ren-Qian Zhang, Qiu-Hong Zhao, Ikou Kaku, Permutation flow shop scheduling with order acceptance and weighted tardiness, Applied Mathematics and Computation 218:7911-7926.
38. Yang B., Geunes J., 2007, A single resource scheduling problem with job-selection flexibility, tardiness costs and controllable processing times, Computers & Industrial Engineering 53:420–432.
39. Yingjun W., Zhongying Z., 2004, Pattern mining for time series based on cloud theory pan-concept-tree, International conference on rough sets and current trends in computing, 1:618–623.