

Two new meta-heuristics for no-wait flexible flow shop scheduling problem with capacitated machines, mixed make-to-order and make-to-stock policy

Sana Abdollahpour¹ · Javad Rezaian¹

Published online: 24 May 2016
© Springer-Verlag Berlin Heidelberg 2016

Abstract This paper focuses on solving no-wait flexible flow shop scheduling problem with capacitated machines and mixed make-to-order and make-to-stock production management policy restrictions. The considered objective function is minimization of the sum of tardiness cost, weighted earliness cost, weighted rejection cost and weighted incomplete cost. Considering the literature, this problem is known as NP-hard. Hence, the cloudy-based simulated annealing (CSA) and artificial immune system are developed to solve the considered problem. Due to the fact that the parameters may influence the meta-heuristic algorithms, the parameters tuning is performed by Taguchi method. Finally, the performances of algorithms are evaluated by solving the randomly generated problems. Computational experiments show that the CSA algorithm obtains higher-quality solutions than another one.

Keywords No-wait flexible flow shop · Sequence-dependent setup times · Make-to-order · Make-to-stock · Cloudy-based simulated annealing · Artificial immune system

1 Introduction

Scheduling is an integral part of advanced manufacturing systems (Davendra et al. 2013). In the most manufacturing systems, it is required that for completion of a job, a set of processes needs to be performed serially (Javadi et al. 2008).

Communicated by A. Di Nola.

✉ Javad Rezaian
j_rezaian@ustmb.ac.ir

¹ Department of industrial engineering, Mazandaran University of Science and Technology, Babol, Iran

In scheduling, this system is called flow shop. Emergence of advanced manufacturing systems such as computer-aided design/computer-aided manufacturing (CAD/CAM), flexible manufacturing system (FMS) and computer-integrated manufacturing (CIM) has increased the importance of flow shop scheduling (Solimanpur et al. 2004).

The goal of flow shop scheduling problems is determination of a job sequence that optimizes one or more performance measures such as maximum completion time (makespan), total tardiness and work in process. In the flow shop environments, where the processing route is the same for all jobs, the problem is called permutation flow shop. On the other hand, where the jobs sequence may be different on each machine, the problem is known as non-permutation flow shop.

The general assumption in flow shop applications is that the sequencing of jobs relies on buffers, which are assigned in consecutive machines. However, in many scheduling environments, some or all jobs need to be proceeded continuously through all machines. This situation is commonly known as “no-wait” (Seido Nagano and Almeida da Silva 2012). A typical example of no-wait condition is steel-making production where the molten steel must be carried out between production stages continuously to reduce its energy loss. Since, in the no-wait flow shop environments, the jobs are not allowed to be idle between machines, the jobs sequence on the machines cannot be changed. So this problem is necessarily a permutation flow shop (Ruiz and Allahverdi 2007). These types of problems are reviewed in detail in Hall and Sriskandarajah (1996), Framinan and Nagano (2008) and Framinan et al. (2010).

From the viewpoint of production management policy, products can be produced according to make-to-stock (MTS), make-to-order (MTO) or mixed MTS/ MTO strategy in industrial environments. Under make-to-order policy, a pro-

duction order is released to the manufacturing facility only after a firm demand has been received (Hadj Youssef et al. 2004). Unlike, under make-to-stock policy, products are manufactured based on anticipation of future market demand and stored in warehouses.

In this study, a manufacturing system that can produce a number of specific products is considered. The jobs, which must be proceed in this system, can be divided into two categories: a set of MTS jobs and a number of MTO job sets (orders), which are released to system in a deterministic times and have certain due dates. Since the available time of each machine is limited, some of the MTO job sets may be rejected. Due to this situation, the goal is to find a sequence of jobs, which are either belong to accepted MTO sets or MTS jobs.

As an example, in a steel factory, production rate of each product is determined by studying on market behavior in past periods. In addition, some customers may release their orders to this factory. In this situation, to optimize the goals of system, decision makers determine which order should be accepted and which one will be rejected.

Due to the time restriction of machines and possibility of rejection one or more MTO job sets, in this study, objective function is included of three parts: costs of rejected orders, costs of earliness and tardiness of each order and the costs of incomplete orders. This objective function can adapt to the philosophy of just-in-time production, which emphasizes producing goods only when they are needed since jobs are scheduled to complete as close as possible to their due dates (Valente Jorge and Alves Rui 2005).

The no-wait flow shop is usually used to some of the important industries such as plastic industries, steel factories and chemical processes. On the other hand, the mixed MTO/MTS strategy can be used to every manufacturing environment which its goal is achieving to a balance between keeping market portion and on-time delivery to customer.

According to the literature, there is no research for combining mixed MTO/MTS policy and no-wait flow shop scheduling. To reduce the gap between scheduling theories and practical applications in industrial environments, two new meta-heuristic algorithms, cloud theory-based simulated annealing (CSA) and artificial immune system (AIS), are proposed for solving the considered problems.

Cloud theory is a model of the fuzzy theory, which relates to quality concepts and quantity data (Deyi et al. 1995). Cloud theory can help to generate a group of continuous temperatures close to a fixed temperature in simulated annealing. Generating random temperature based on cloud theory can preserve diversity and prevents algorithm from being trapped in a local optimum acceptability (Torabzadeh and Zandieh 2010).

Artificial immune system algorithm is a new population based on meta-heuristic approach, which is inspired of bio-

logical immune system (BIS). In comparison with other meta-heuristics, the AIS has the lower complexity, so it is simple to code.

The rest of the paper is organized as follows: In Sect. 2, a literature review of no-wait flow shop scheduling problem, MTO and MTS policies, cloud theory and artificial immune algorithm is presented. Section 3 describes the problem and presents mathematical model. In Sect. 4, the principles of meta-heuristic approaches are described. The computational results and evaluation of proposed algorithm performances are presented in Sect. 5. Finally, conclusions are provided in Sect. 6.

2 Literature review

A highly prominent class of job scheduling problems is identified by a no-wait production environment, in which there is no storage between the machines. Thus, jobs must be processed from start to finish, without any interruptions in the machines or between them (Jolai et al. 2013).

The no-wait flow shop scheduling problem with single objective was proved strongly NP-hard by Rock (Hadj Youssef et al. 2004) when the number of machines is more than two. Therefore, the efforts have been devoted to finding high-quality solutions in reasonable time by using the heuristic and meta-heuristic approaches.

Some heuristic approaches were proposed to solve the no-wait flow shop scheduling problem. For instance, Reddi and Rama-moorthy (1972), King and Spachis (1980) proposed their methods. In addition, in the 1990s, Gangadharan and Rajendran (1993) and Rajendran (1994) proposed two heuristic methods GAN-RAJ and RAJ, which were shown by experiments that these methods outperformed the previous heuristic approaches reported in the literature. In recent years, Seido Nagano et al. (2015) proposed a constructive heuristic methods named QUARTS that breaks the problem to quartets in order to minimize the total flow time. Liu et al. (2013) proposed six heuristics approaches for no-wait flow shops with total tardiness criterion and proved that the modified NEH algorithm (MNEH) is the best.

Several meta-heuristic algorithms were proposed for solving this problem. Qian et al. (2009) proposed an effective hybrid differential evolution (HDE) for the no-wait flow shop scheduling problem with the makespan restriction. Tavakkoli-Moghaddam et al. (2008) presented the immune algorithm approach for solving multi-objective no-wait flow shop. Pan et al. (2009) applied a novel differential evolution algorithm for bi-criteria no-wait flow shop scheduling problem. Wang et al. (2010) proposed a new accelerated tabu search algorithm for no-wait flow shop problem with maximum lateness criterion. Tseng and Lin (2011) presented a hybrid genetic algorithm for no-wait flow shop

with makespan as objective function. More recently, [Seido Nagano and Almeida da Silva \(2012\)](#) applied a new clustering search for this problem with total flowtime criterion. [Ch et al. \(2012\)](#) studied on no-wait flow shop manufacturing cell with sequence-dependent family setup times and presented a number of meta-heuristics for this problem. [Davendra et al. \(2013\)](#) also solved no-wait makespan flow shop by using discrete self-organizing migrating algorithm. [Arabameri and Salmasi \(2013\)](#) used several meta-heuristic approaches for solving $F_m |nwt, S_{ijk}| \sum W'_j E_j + W''_j T_j$. As newest studies, [Shabtay et al. \(2014\)](#) defined a combined robot selection and scheduling problem (RSSP) for a set of Q nonidentical robots characterized by different costs and job transfer and empty movement times. [Laha and Sapkal \(2014\)](#) presented a constructive heuristic based on the assumption that the priority of a job in the initial sequence is given by the sum of its processing times on the bottleneck machines to minimize total flowtime criterion for no-wait flow shop scheduling problem.

It is more realistic to assume that at least one stage has more than one machine in flow shop environments. So in recent years, the flexible flow shop has attracted wide attention in both of academic and industrial societies. As an example of the no-wait flexible flow shop, [Jolai et al. \(2009\)](#) proposed a genetic algorithm for no-wait flexible flow shop with due window and job rejection. [Jolai et al. \(2013\)](#) also applied bi-objective simulated annealing approaches for no-wait two-stage flow shop problems. [Wang and Liu \(2013\)](#) studied on this problem and proposed the genetic algorithm for solving it. [Ramezani et al. \(2013\)](#) considered the no-wait flow shop with uniform parallel machines in each stage with sequence-dependent setup time constraint. In addition, [Pang \(2013\)](#) presented a genetic algorithm (GA)-based heuristic approach to solve the problem of two-machine no-wait flow shop scheduling problems that the setup times of machines are class dependent, and the objective is to minimize the maximum lateness. [Liu and Feng \(2014\)](#) studied on two-machine no-wait flow shop scheduling problems in which the processing times of jobs are functions of their positions in the sequence and solved it by using the classic KM (Kuhn–Munkres) algorithm.

According to the above MTO and MTS definitions, an important problem in MTS systems is the high inventory holding costs. On the other hand, an important drawback in MTO policies is the challenge of products on-time delivery, so in recent years the combination strategies attracted wide attention in both academic and industrial fields. For example, [Adan and Wall \(1998\)](#), [Federguen and Katalan \(1999\)](#) and [Soman et al. \(2004\)](#).

Corresponding to the characteristics of producible products, the MTO and MTS strategy are implemented by different methods. As examples, [Eivazy et al. \(2009\)](#) used hybrid MTO/MTS policy for semiconductor manufacturing

systems. [Hadj Youssef et al. \(2004\)](#) studied on efficient scheduling rules in combined MTO/MTS strategy in manufacturing systems. [Zaerpour et al. \(2009\)](#) combined analytical hierarchy process (AHP) and technique for order performance by similarity to ideal solution (TOPSIS) methodologies.

Cloud theory is an expansion of membership function of fuzzy theory ([Deyi and Yi 2005](#)) so many scientists use its applications like intelligence control ([Deyi et al. 1998; Feizhou et al. 1999](#)), knowledge representation ([Cheng et al. 2005; Deyi et al. 2000](#)), data mining ([Di et al. 1999; Kaichang et al. 1998; Shuliang et al. 2003; Yingjun and Zhongying 2004](#)), spatial analysis ([Cheng et al. 2006; Haijun and Yu 2007](#)), target recognition ([Fang et al. 2007](#)), intelligent algorithm improvement ([Yunfang et al. 2005](#)) and so on ([Torabzadeh and Zandieh 2010](#)). As an example, [Torabzadeh and Zandieh \(2010\)](#) used cloudy simulated annealing approach for solving two-stage assembly flow shop and the cloudy theory obtained good solutions, so in this study cloudy simulated annealing with various sizes of solutions is proposed to solve the considered problem.

The field of immunological computation (IC) or artificial immune system (AIS) has been evolving steadily ([Nunes de castro and Von Zuben 1999](#)) since 1985. In recent years, several researchers have developed computational models of the immune system that attempt to capture some of its most remarkable features such as its self-organizing capability ([Coello Coello and Cruz Cort'es 2005](#)).

Recently, the AIS algorithm has been applied with success for different variants of scheduling problems, for examples [Lin and Ying \(2013\)](#) proposed revised version of the AIS algorithm for flow shop with limited buffer scheduling problem. [Zandieh et al. \(2006\)](#) applied clonal selection principle to solve flow shop with sequence-dependent setup times.

3 Problem description and mathematical model

In this section, at first the problem statement is described. Then, assumptions and mathematical model are presented.

The flexible no-wait flow shop with mixed MTO/MTS production management policy and time limitation of machines is defined as follows:

Let n be the number of product types ($J = 1, 2, \dots, n$) which each of them has a deterministic processing time in each stage (p_{js}). This system must be produced a set of jobs, which is called MTS jobs ($i = 1$). In addition, a given number of order sets ($i = 2, \dots, N$) are released to system as MTO sets. Both MTS and MTO sets are presented by $O_i = \{h_{1i}, \dots, h_{ni}\}$ where h_{ni} is the request number of product type n in order set i .

Due to the time limitation of machines, some of the MTO sets may be rejected and even some of the accepted orders may be incomplete.

From the above definition if $N = 5$ and $n = 3$ and the accepted order sets are $\{O_1, O_3, O_4\}$, the jobs must follow though the systems are $\{(h_{11}, h_{13}, h_{14}), (h_{21}, h_{23}, h_{24}), (h_{31}, h_{33}, h_{34})\}$ and $Z = (h_{11} + h_{13} + h_{14}) + (h_{21} + h_{23} + h_{24}) + (h_{31} + h_{33} + h_{34})$.

All accepted jobs must be produced in flexible no-wait flow shop system, which has m_s identical parallel machines at stages. Each machine is available at time zero to ca_{sk} and after that gets unavailable (for example maintenance).

In no-wait flow shop, each job may be delayed to satisfy the no-wait condition. So, some of available time of machines will be idled and some of accepted jobs may be incomplete.

After determination of accepted orders, all of them and MTS jobs are sequenced ($q = 1, \dots, Z$, where Z is the total of jobs will be scheduled) to flow through the manufacturing system. To follow the no-wait restriction, all of consecutive operations of each job must be done continuously.

The schematic definition of the considered problem is depicted in Fig. 1.

It is remarkable that the proposed nonlinear mathematical model in this study is a developed version of the presented mathematical model by Jolai et al. (2013).

Following assumptions are taken into account for the considered problem:

- All machines are available at time zero.
- In each stage, there is one machine at least, and one stage must have more than one machines at least.
- Each machine can process at most one job, and each job must be processed only by one machine at each stage.
- Machines of all stages are identical.
- A deterministic release date and due date are defined for each order. The release dates and due dates of make-to-stock jobs are equal to zero and a large number, respectively.
- If two consecutive jobs are from different types on a machine, the setup time must be considered.
- Travel times between stages are negligible.

The parameters, decision variables and the mathematical model are as follows:

Parameters:	
k : Index of machines at stage s	$s = 1, \dots, S, k = 1, \dots, m_s$
j : Index of product types that can be processed in manufacturing system	$j = 1, \dots, n$
i : Index of order set	$i = 1, \dots, N$
t : Index of job in each order set	$t = 1, \dots, \text{sum}_i, i = 1, \dots, N$
(sum_i : The summation of the number of jobs in each order set)	
q : Index of location of each job in sequence	$q = 1, \dots, Z$
R_i : Release date of order i (The MTS jobs are available at time zero)	$i = 2, \dots, N$
D_i : Due date of order i (The MTS jobs are delivered at the end of time horizon)	$i = 2, \dots, N$
Wt_i : The tardiness penalty of order i for each time unit of tardiness	$i = 2, \dots, N$
We_i : The earliness penalty of order i for each time unit of earliness	$i = 2, \dots, N$
Wr_i : The rejection cost of order i	$i = 2, \dots, N$
Wg_i : The incompleteness cost of order i for each job of it	$i = 1, \dots, N$
ca_{sk} : Time constraint of machine k on stage s	$s = 1, \dots, S, k = 1, \dots, m_s$
p_{js} : Processing times of job type j on stage s	$j = 1, \dots, n, s = 1, \dots, S$
h_{ji} : The number of job type j in order i	$i = 1, \dots, N, j = 1, \dots, n$
$s_{jj'sk}$: Setup time of job type j' when j' is processed immediately after job type j on stage s on machine k	$j, j' = 1, \dots, n, j \neq j', s = 1, \dots, S, k = 1, \dots, m_s$
Decision variables:	
x_{itq} : 1 if t th job from order i is located in position q of job sequence. 0 otherwise	$t = 1, \dots, \text{sum}_i, i = 1, \dots, N, q = 1, \dots, Z$
y_{qsk} : 1 if the job where is located in position q is processed on machine k at stage s . 0 otherwise	$q = 1, \dots, Z, s = 1, \dots, S, k = 1, \dots, m_s$
v_{qj} : 1 if the job where is located in position q is type of j . 0 otherwise	$q = 1, \dots, Z, j = 1, \dots, n$
O_i : 1 if order i is accepted. 0 otherwise.	$i = 1, \dots, N$
st_{qs} : Start time of the job where is located in position q at stage s	$q = 1, \dots, Z, s = 1, \dots, S$
c_{qs} : Completion time of the job where is located in position q at stage s	$q = 1, \dots, Z, s = 1, \dots, S$
de_q : Delay value of the job where is located in position q	$q = 1, \dots, Z$
av_{qs} : Available time of the job where is located in position q at stage s	$q = 1, \dots, Z, s = 1, \dots, S$
at_{qsk} : Available time of the job where is located in position q at stage s on machine k	$q = 1, \dots, Z, s = 1, \dots, S, k = 1, \dots, m_s$
g_q : 1 if the job where is located in position q is completed. 0 otherwise	$q = 1, \dots, Z$
nond_i : The number of incomplete jobs in order i	$i = 1, \dots, N$

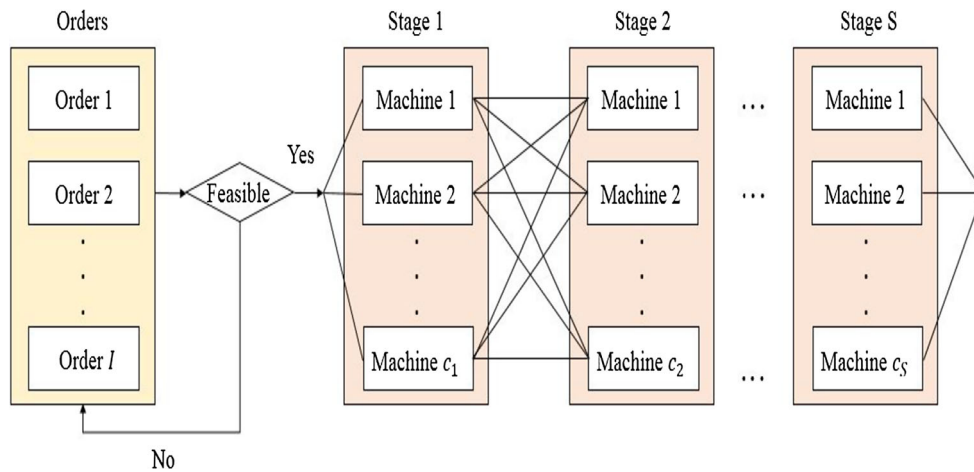


Fig. 1 Presentation of a flexible no-wait flow shop problem with capacity restriction and mixed MTO/MTS policy

The nonlinear mathematical model:

$$\begin{aligned} \min Z = & \left(\sum_{i=2}^N tardi_i \times O_i \times Wt_i \right) \\ & + \left(\sum_{i=2}^N earli_i \times O_i \times We_i \right) \\ & + \left(\sum_{i=2}^N Wr_i \times (1 - O_i) \right) \\ & + \left(\sum_{i=1}^N (\text{sum}_i - \text{nond}_i) \times O_i \times Wg_i \right) \end{aligned} \tag{1}$$

Subject to:

$$at_{qsk}y_{qsk} + \sum_{j=1}^n v_{qj}p_{js} \leq ca_{sk}, \quad q = 1, \dots, Z, \tag{2}$$

$$s = 1, \dots, S, k = 1, \dots, m_s \tag{3}$$

$$O_i \geq 1 \tag{4}$$

$$\text{sum}_i = \sum_{j=1}^n h_{ji}, \quad i = 1, \dots, N \tag{5}$$

$$Z = \sum_{i=1}^N \text{sum}_i O_i, \quad i = 1, \dots, N \tag{6}$$

$$\sum_{t=1}^{\text{sum}_i} x_{tiq} = 1, \quad q = 1, \dots, Z \text{ and } i = 1, \dots, N \tag{7}$$

$$\sum_{q=1}^Z x_{tiq} = 1, \quad t = 1, \dots, \text{sum}_i \text{ and } i = 1, \dots, N \tag{8}$$

$$\sum_{k=1}^{m_s} y_{qsk} \times at_{qsk} \leq av_{qs}, \quad q = 1, \dots, Z \text{ and } s = 1, \dots, S \tag{9}$$

$$\sum_{k=1}^{m_s} y_{qsk} \leq 1, \quad q = 1, \dots, Z \text{ and } s = 1, \dots, S \tag{10}$$

$$de_q = \max \left(\sum_{i=1}^N \sum_{t=1}^{\text{sum}_i} r_i O_i x_{tiq}, \right. \\ \left. (av_{qs} - av_{qs-1}), \dots, (av_{q2} - av_{q1}) \right), \\ q = 1, \dots, Z \tag{11}$$

$$st_{q1} = \max(de_q + av_{q1}), \quad q = 2, \dots, Z \tag{12}$$

$$st_{qs} = c_{qs-1}, \quad s = 2, \dots, S \text{ and } q = 2, \dots, Z \tag{13}$$

$$c_{qs} = st_{qs} + \sum_{j=1}^n v_{qj}p_{js}, \quad s = 1, \dots, S \tag{14}$$

$$at_{qks} = \max_{1 \leq q' \leq q-1} (C_{q's} y_{q'sk}), \quad s = 1, \dots, S, \\ q = 2, \dots, Z, k = 1, \dots, m_s \tag{15}$$

$$av_{qs} = \min_{1 \leq k \leq m_s} (at_{qks}), \quad s = 1, \dots, S \text{ and } \\ q = 2, \dots, Z \tag{16}$$

$$tardi_i = \max \left(0, \left(\max_{1 \leq q \leq Z} \left(\sum_{t=1}^{\text{sum}_i} x_{tiq} c_{qs} \right) - d_i \right) \right), \\ i = 2, \dots, N \text{ and } s = S \tag{17}$$

$$earli_i = \max \left(0, \left(d_i - \max_{1 \leq q \leq Z} \left(\sum_{t=1}^{\text{sum}_i} x_{tiq} c_{qs} \right) \right) \right), \\ i = 2, \dots, N \text{ and } s = S \tag{18}$$

$$S - \sum_{s=1}^S \sum_{k=1}^{m_s} y_{qsk} \geq (1 - g_q), \quad q = 1, \dots, Z \text{ and } \\ s = 1, \dots, S \tag{19}$$

$$\text{nond}_i = \text{sum}_i - \sum_{t=1}^{\text{sum}_i} \sum_{q=1}^Z g_q x_{tiq} \quad i = 1, \dots, N \tag{20}$$

The objective function, as presented in Eq. (1), is minimization of the total weighted earliness, tardiness, missed orders and incomplete orders. Constraint (2) ensures that a machine cannot be assigned to a given job unless its time limitation is satisfied by that job. Constraint (3) ensures that the MTS jobs are processed. Constraint (4) calculates the number of jobs which are requested by each order set. Constraint (5) determines the summation of jobs which are requested by accepted order sets. Constraints (6) and (7) determine the job sequence. Constraint (8) ensures that each job is allocated to a machine which will be available sooner than the other ones at each stage. Constraint (9) completes the process of assigning machines to jobs and ensures each job at each stage is allocated to one machine at most. Constraint (10) calculates the minimum delay on the first machine before the job processes are started in order to satisfy the no-wait condition. Constraints (11) and (12) compute the start time of jobs at each stage. Constraint (13) calculates the completion times of jobs at each stage. For each job, constraint (14) computes the available time of each machine. In addition, available time of each stage for each job is determined by constraint set (15). Since the goal of this problem is on-time delivery of orders, constraints (16) and (17) compute the earliness and tardiness of each order set. Constraint (18) counts the number of completed jobs in each order set. For a job, if a machine is assigned at each stage, so the job is called completed job. Constraint (19) computes the incomplete jobs of each accepted order set. If the *nound* value for each order set is greater than zero, the order set is known as incomplete order set.

We use the following example to illustrate how the above model works. Suppose there are 3 types of products ($n = 3$), two stage ($m_1 = 3, m_2 = 2$) and 4 order sets are released to system ($N = 4$). According to the previous agreement, the first-order set contains the MTS jobs and the other ones are MTO job sets. If we suppose that the third-order set is accepted ($O_3 = 1$) and the others are rejected, one possible sequence of accepted jobs is shown below (Q).

$$r_3 = 5$$

$$d_3 = 50$$

$$Wt_i = We_i = Wr_i = Wg_i = 5, i = 1, \dots, N$$

$$p = \begin{bmatrix} 24 \\ 53 \\ 45 \end{bmatrix}$$

$$O_1 = \{1, 2, 1\} \rightarrow \text{sum}_1 = 4 \rightarrow \begin{cases} \text{number} : \{1, 2, 3, 4\} \\ \text{type} : \{1, 2, 2, 3\} \\ \text{order} : \{1, 1, 1, 1\} \end{cases}$$

$$O_3 = \{1, 1, 1\} \rightarrow \text{sum}_3 = 3 \rightarrow \begin{cases} \text{number} : \{1, 2, 3\} \\ \text{type} : \{1, 2, 3\} \\ \text{order} : \{3, 3, 3\} \end{cases}$$

$$Z = \text{sum}_1 + \text{sum}_2 = 7$$

$$Q = \left. \begin{cases} q : \{1, 2, 3, 4, 5, 6, 7\} \\ t : \{1, 3, 3, 1, 4, 2, 2\} \\ \text{type} : \{1, 2, 3, 1, 3, 2, 2\} \\ i : \{3, 1, 3, 1, 1, 3, 1\} \end{cases} \right\} \rightarrow x_{131} = 1, v_{11} = 1$$

Based on sequence Q , the equations of the model are calculated in Table 1.

Based on the results of Table 1, it is clear that $earli_3 = 25, tardi_3 = 0$ and the objective function is 135. The Gantt chart of this example is shown in Fig. 2.

4 Solution approaches

In this section, the solution approaches are proposed. As well known, the meta-heuristic algorithms have significant role in tackling all category of combinatorial problems. Scheduling problems are kind of optimization problems, which was solved by meta-heuristic algorithms in recent years (Li et al. 2012).

As far as we know, the considered problem in this paper is not solved before this, and this study presents two newest meta-heuristic algorithms for solving it.

The cloudy simulated annealing (CSA) represents the single-point methods and the artificial immune system (AIS) as a population-based algorithm is applied to solve the considered problem.

In this part of paper, each algorithm is briefly described and then is implemented for the considered problem.

4.1 Cloud theory-based simulated annealing algorithm

4.1.1 Simulated annealing algorithm in general

Simulated annealing algorithm (SA) is a random optimization method that is based on Monte Carlo iterative strategy (Laarhoven and Aarts 1987). SA is inspired of the physical annealing process, which minimizes potential energy in a solid object. This algorithm starts with a primary solution (S_0) and primary temperature (T_0), and this improving mechanism consists of the following steps: T_0 is decreased according to specific cooling schedule function, and a new solution (S_1) is found in the neighborhood of the current solution (S_0). If the value of objective function ($f(S_1)$) is less than the value of objective function ($f(S_0)$) in minimization problems, the new solution will be accepted, otherwise it will be accepted with probability $p = e^{-\frac{\Delta}{T}}$ (Boltzmann distribution function), where $\Delta = \frac{f(S_1) - f(S_0)}{f(S_1)}$ and T is the current temperature. This process is continued until the termination criterion is met.

Table 1 A numerical example data

q	s	1	2	3
1		$sf_{11} = (5 + 0) = 5 \rightarrow c_{11} = 5 + 2 = 7$	$sf_{12} = 7 \rightarrow c_{12} = 7 + 4 = 11$	$de_1 = \max(5, 0) = 5$
2		$\begin{cases} at_{211} = \max(7 \times 1) = 7 \\ at_{212} = \max(7 \times 0) = 0 \\ at_{213} = \max(7 \times 0) = 0 \end{cases} \rightarrow av_{21} = \min(7, 0, 0) = 0$	$\begin{cases} at_{221} = 0 \\ at_{222} = 11 \end{cases} \rightarrow av_{21} = \min(7, 0, 0) = 0$	$de_2 = 0$
3		$sf_{21} = 0 \rightarrow c_{21} = 5$	$sf_{22} = 5 \rightarrow c_{22} = 5 + 3 = 8$	$de_3 = \max(5, (8 - 0)) = 8$
4		$\begin{cases} at_{311} = 7 \\ at_{312} = 5 \\ at_{313} = 0 \end{cases} \rightarrow av_{31} = 0$	$\begin{cases} at_{321} = 8 \\ at_{322} = 11 \end{cases} \rightarrow av_{31} = 8$	
5		$sf_{31} = 8 \rightarrow c_{31} = 12$	$sf_{32} = 12 \rightarrow c_{32} = 17$	
6		$\begin{cases} at_{411} = 7 \\ at_{412} = 5 \\ at_{413} = 12 \end{cases} \rightarrow av_{41} = 5$	$\begin{cases} at_{421} = 17 \\ at_{422} = 11 \end{cases} \rightarrow av_{41} = 11$	$de_4 = \max(0, (11 - 5)) = 6$
7		$sf_{41} = 11 \rightarrow c_{41} = 13$	$sf_{42} = 13 \rightarrow c_{42} = 17$	
8		$\begin{cases} at_{511} = 7 \\ at_{512} = 13 \\ at_{513} = 12 \end{cases} \rightarrow av_{51} = 7$	$\begin{cases} at_{521} = 17 \\ at_{522} = 17 \end{cases} \rightarrow av_{51} = 17$	$de_5 = \max(0, (17 - 7)) = 10$
9		$sf_{51} = 17 \rightarrow c_{51} = 21$	$sf_{52} = 21 \rightarrow c_{52} = 26$	
10		$\begin{cases} at_{611} = 21 \\ at_{612} = 13 \\ at_{613} = 12 \end{cases} \rightarrow av_{61} = 12$	$\begin{cases} at_{621} = 26 \\ at_{622} = 17 \end{cases} \rightarrow av_{61} = 17$	$de_6 = \max(5, (17 - 12)) = 5$
11		$sf_{61} = 17 \rightarrow c_{61} = 22$	$sf_{62} = 21 \rightarrow c_{62} = 25$	
12		$\begin{cases} at_{711} = 21 \\ at_{712} = 13 \\ at_{713} = 22 \end{cases} \rightarrow av_{71} = 13$	$\begin{cases} at_{721} = 26 \\ at_{722} = 25 \end{cases} \rightarrow av_{71} = 25$	$de_7 = \max(0, (25 - 13)) = 12$
13		$sf_{71} = 25 \rightarrow c_{71} = 30$	$sf_{72} = 30 \rightarrow c_{72} = 33$	

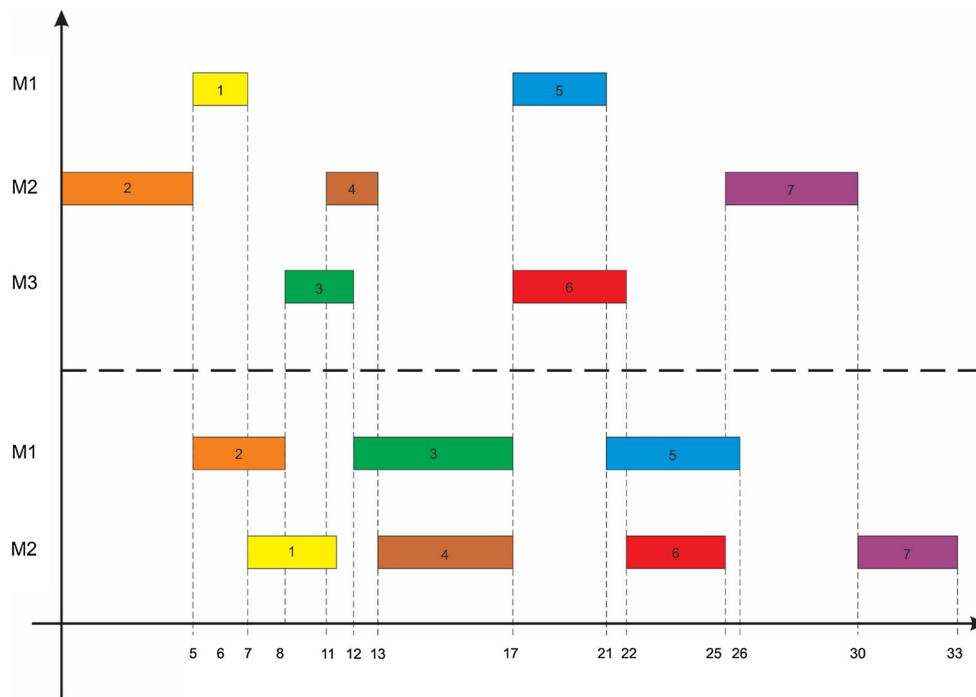


Fig. 2 The Gantt chart of numerical example

4.1.2 Basic concept of cloud theory

The cloud theory is a model that contains the transferring procedure of uncertainty between quality concept and quantity data representation by using natural language (Valente Jorge and Alves Rui 2005). Cloud theory is innovation and development of membership function of fuzzy theory (Di et al. 1999).

Let D be the language value of domain u and mapping $C_D(x) : u \rightarrow [0, 1], \forall x \in u, x \rightarrow C_D(x)$, if the distribution of $C_D(x)$ is normal, it is named a normal cloud model (Deyi et al. 1995).

Cloud theory generates a group of random numbers which have a same distribution, usually normal distribution. These numbers are determined by expectation E_x , entropy E_n and super entropy He . They reflect the quantitative characteristics of the generated cloud (Fig. 3).

A generator Y , condition cloud generator can generate a drop of cloud ($drop(x_i, u_0)$) with three characters E_x, E_n, He and a certain u_0 as follows (Deyi and Yi 2005):

1. Input: $\{E_x, E_n, He\}, n, u_0$
2. Output: $\{(x_1, u_0), (x_2, u_0), \dots, (x_n, u_0)\}$
3. For $i = 1$ to n
4. $E'_n = randn(E_n, He)$
5. $x_i = E_x \pm E'_n \sqrt{-2 \ln(u_0)}$
6. $drop(x_i, u_0)$
7. End

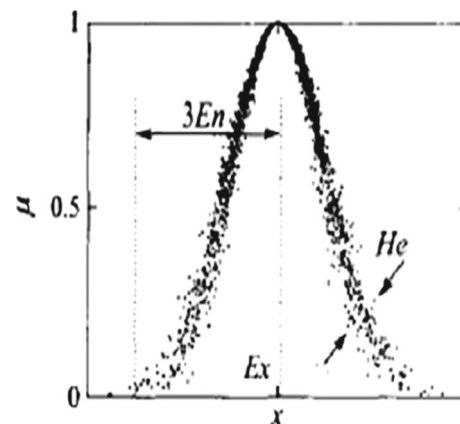


Fig. 3 Three digital characteristics of a normal cloud (Adan and Wall 1998)

Where $randn(E_n, He)$ will produce a random number with normal distribution which expectation is E_n and standard deviation is He (Deyi et al. 1995) (Fig. 3).

4.1.3 Cloud theory-based simulated annealing algorithm

Based on the SA concept, since the algorithm accepts deteriorate solutions easily, it has not good convergence rate with high temperature; on the other hand, SA accepts good solutions hardly with low temperature, so it may drop to local minimum trap. Therefore, it is necessary to find other cooling mechanism that improves searching ability and obtains better solutions.

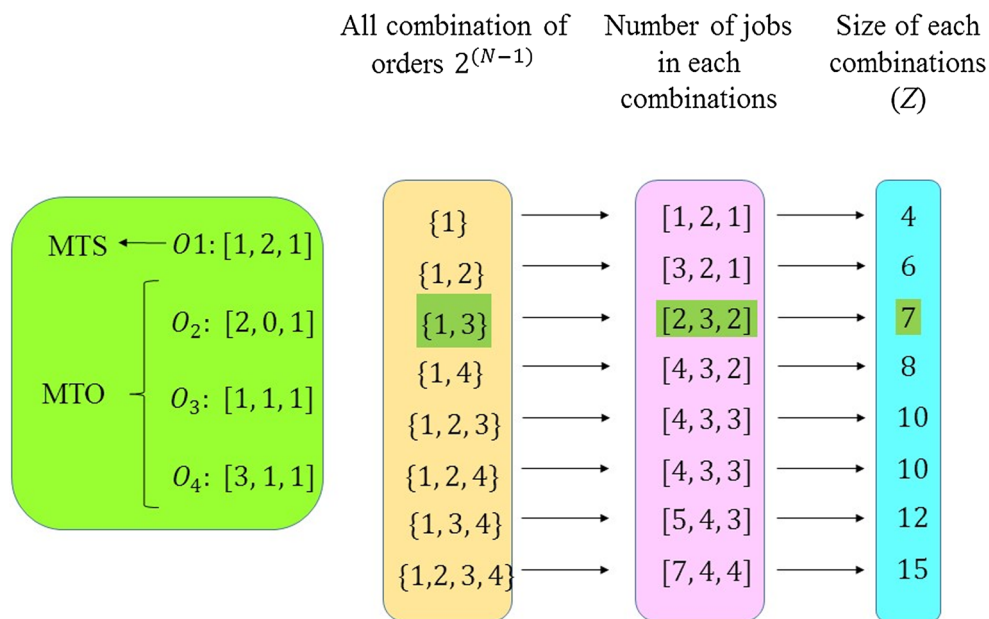


Fig. 4 The numerical example of how consider all of combinations of orders

In the cloud theory-based simulated annealing algorithm, random numbers by same normal distribution which are generated by Y generator normal cloud are used for continuous annealing process. The cloud theory has the characteristics of randomness and stable tendency, so the annealing temperature changes randomly, and it can preserve the diversity of searching and therefore avoid being trapped in local minimum (Valente Jorge and Alves Rui 2005).

At first, the initial temperature (T_0) has to be unified. Then, all possibility combination of orders as accepted orders are determined (2^{N-1}). It is noticeable that the make-to-stock jobs are always considered as the accepted orders. Therefore, CSA algorithm is applied for all of combination of accepted orders by the following manner:

T_0 is decreased according to specific cooling schedule. For each temperature which is generated according to the cloud theory function, a random permutation of accepted orders is determined as an initial solution (S_0). A new solution (S_1) is generated by the mutation methods, and then, the new solution will be accepted if the objective function value of new solution is better than the objective value of the current solution, otherwise the new solution will be accepted with certain probability. These steps are repeated until the termination conditions are satisfied.

In order to explain how the all combination of orders is considered, the following example is provided.

According to the previous example (Table 1), by considering a 4-order ($N = 4$) and 3-producible job ($n = 3$) problem, all combination of orders are shown in Fig. 4.

As shown in Fig. 4, O_1 represent the MTS jobs and other sets are referred to the MTO order sets. Since the MTS order

is always accepted, the number of all combination of orders as accepted orders is $2^3 = 8$.

According to Fig. 4, if only order 3 is accepted, order 3 and order 1 (MTS jobs) must be processed. So, the number of jobs is equal to $(1 + 1 + 1) + (1 + 2 + 1) = 7$.

Solution representation In order to represent the solution of the problem, a matrix with three rows is generated. The first row represents the number of each job in each accepted order. In the second row, the type of each job is determined and the third row includes the order numbers of jobs.

As an example of solution representation, the job sequence of previous example (Q in Table 1) is shown in Fig. 5.

In both of CSA and AIS algorithms, the solutions are represented similarly.

Neighborhood solution generation The purpose of generating neighborhood solutions is to find the better solutions from the neighborhood space of a solution. The neighborhood of S_0 is defined as the set of all solutions that can be generated by

1	3	3	1	4	2	2
1	2	3	1	3	2	2
3	1	3	1	1	3	1

Fig. 5 The numerical example of solution representation

Cloud theory-based simulated annealing algorithm:

1. Input: define all of the possibility combinations of accepted order and the make to stock jobs.
2. Select the first combination of orders and define a permutation job of it as initial solution (S_0).
3. **For** all order combinations.
4. Let $T_0 = 1, T_f = 0.000001$, determine the *iteration* value according to $iteration = size(S_0) * 10$;
5. $T = T_0, S = S_0, k = k_0$
6. **While** $T > T_f$
7. **Repeat**
8. $He = T; En = T; u = 1 - T;$
9. $E'_n = (E_n + He - \frac{1}{3rand(0,1)});$
10. $T' = E'_n \sqrt{-2\ln(u_0)};$
11. $L_1 =$ value of the objective function with current sequence;
12. **For 1: iteration**
13. $S' =$ mutated the current sequence;
14. $L_2 =$ value of the objective function with mutated sequence;
15. **If** $L_2 \leq L_1$ **then**
16. $S = S'$
17. **Else**
18. Accept S as new solution with probability $P = e^{\frac{-\Delta}{T}}$, where $\Delta = \left| \frac{L_2 - L_1}{L_1} \right|$ (Boltzmann distribution function);
19. **End if**
20. **End for**
21. **End repeat**
22. $k = k + 1;$
23. $T = T_0 \lambda^k;$
24. **End while**
25. Store the obtained best objective function value and its sequence for each of combination of the orders;
26. **End for**

Fig. 6 The pseudo-code of the proposed cloud theory-based simulated annealing algorithm

applying a specific operator to S_0 . In this study, two common insertion and swap operators are designed for generating the neighborhood solutions.

Insertion operator will be selected and eliminated a job randomly, and then, the solution is reproduced by reinserting the eliminated job in a new randomly selected location.

For swap operator, at first, two jobs of the S_0 are randomly selected. Then, they are swapped in their positions.

The advantage of CSA algorithm is neighborhood search (local search) ability. In each iteration (temperature) of algorithm, a set of temperature is produced by the cloud model. These temperatures will create an inner loop that its task is local search.

Annealing process In general SA, the annealing temperature is a fixed value at each step and the searching process is completed between neighbors (Framinan et al. 2010).

In proposed CSA algorithm, the temperature is updated by using an exponential function, which is shown in Eq. 20.

$$T = T_0 \lambda^k \quad k = 1, 2, \dots, 0 < \lambda < 1 \quad (20)$$

where λ is the annealing index and k is the step counter.

In addition by using Y generator and current temperature, a set of new temperatures which are distributed around the current temperature is generated. This set is called “cloud.”

Termination criterion In this study, the proposed CSA algorithm is terminated if T is lower than the pre-specified final temperature T_f .

The pseudo-code of the proposed CSA algorithm is shown in Fig. 6.

It is necessary to notice that the value of *iteration* is depended on the number of jobs in the current sequence; in other words, where the size of sequence is increased, the number of temperatures which are generated by Y generator is increased too. The *iteration* for each combination of accepted orders is calculated by Eq. 21.

$$iteration = size\ of\ considered\ combination \times 10 \quad (21)$$

4.2 Artificial immune system

The biological immune system (BIS) is a robust and adaptive system that its duty is defending the body from foreign elements that called pathogens. BIS is able to recognize both body cells and non-body cells. The immune defense mechanisms include either nonspecific (innate) or specific (acquired) (Khoo and Situmdrang 2003).

The algorithms that are inspired of BIS are called artificial immune systems (AISs). AIS includes three branches: clonal selection, negative selection and immune network. The clonal selection principles explain how the BIS recognizes the pathogens and can generate the best antibodies for eliminating them. For applying clonal selection method of AIS algorithm for the considered problem, all possibility combination of orders as accepted orders are determined. Then, AIS algorithm is applied for all combination of accepted orders by the following steps:

1. Generate a fixed number (pop size) of antibodies as a primarily antibody population.
2. Calculate affinity function values (based on objective function) for each antibody.
3. Select the best fixed number (n_c) of antibodies.
4. Clone each selected antibody according to its affinity value.
5. All antibodies in the clone population are mutated (based on their affinity values).
6. The worst n_c antibodies in the current population are replaced by the best n_c mutated antibodies.

These steps are repeated until the termination criterion is satisfied.

4.2.1 Initial population

The initial population consists of $popsize$ randomly generated solutions. The antibody representation of AIS algorithm is similar to the solution representation (4.1.3.1) of CSA algorithm (Fig. 5).

4.2.2 Affinity calculation

Each antibody has an affinity value, which is referred to its objective function value. The affinity value for each antibody is calculated by the ($-objective\ function$). From this value, it is clear that a solution with a higher affinity value is better than a solution with lower one.

4.2.3 Cloning phase

During the AIS algorithm, n_c antibodies with highest affinity values are selected for cloning step. Based on the idea of

Wang and Liu (2013), the clone number of each n_c selected antibody is calculated by $(n_c - k + 1)$, where k denotes the antibody with the k th highest affinity function value in the antibody population (Lin and Ying 2013). So the antibodies with higher affinity values have a higher number of clones.

4.2.4 Mutation phase

In proposed AIS algorithm, two mutation operators (swap and insertion) are considered. Based on the AIS algorithm concepts, for each antibody the replication number of mutation step (*iteration*) is increased where the k value of its antibody is increased.

4.2.5 Stopping criterion

In this study, the maximum number of generations (*maxgen*) is considered as the termination condition. Since in each combination of accepted orders the number of jobs is changed, the *maxgen* for each combination is calculated by Eq. 22.

$$maxgen = size\ of\ considered\ combination \times 10 \quad (22)$$

The pseudo-code of AIS algorithm is shown in Fig. 7.

5 Computational experiments

In this section, to examine the effectiveness of the proposed algorithms, after tuning parameters, their performances are evaluated by solving the randomly generated test problems and the results of this evaluation are described.

In addition, apart from what algorithm is better, some tests are performed to show the effects of management decisions to the performance measurements.

5.1 Test instance generation

Due to the unavailability of the standard test problems for the considered problem, to evaluate the effectiveness of the proposed algorithms, test problems are generated randomly.

The data that are required to generate the random problems are: the number of orders, the number of producible jobs, the maximum number of jobs which can be ordered by each order family, the number of stages, the number of machines in each stage, processing time of each job at each stage, time limitation of machines, release date and due date of each order family and weights of earliness, tardiness, rejection and incompleteness of each order family. Table 2 shows the generated problem. It is remarkable that in order to generate the parameter values, we used Ramezani et al. (2013) benchmark problems. These values are adopted for a flexible no-wait flow shop system such as a steel factory.

Artificial immune system algorithm:

1. Input: define all of the possibility combinations of accepted order and the make to stock jobs.
2. Select the first combination of orders and define a permutation job of it as initial solution (s_0).
3. **For** all order combinations
4. Generate the *popsize* of the s_0 permutation randomly as initial population, determine the *iteration* value according to size of s_0 ;
5. Calculate the affinity function of these antibodies;
6. **For** *maxgen*
7. Select the best n_c of the antibodies;
8. Make $(n_c - k + 1)$ copies of each antibodies (clone);
9. Mutate all of the cloned antibodies;
10. Calculate the affinity function value for all of the mutated antibodies;
11. The n_c of the worse antibodies in current population are replaced by the n_c of the best mutated antibodies;
12. **End for**
13. Store the obtained best affinity function value and its sequence for each of combination of the orders;
14. **End for**

Fig. 7 The pseudo-code of the proposed artificial immune system algorithm**Table 2** Problem parameters and their values

Parameters	Values
Number of family orders	Small: 1, 2 Large: 3, 5
Number of product able jobs	Small: 2, 4 Large: 3, 5
Maximum number of each producible job can be ordered by each order family	Small: 5, 10 Large: 25, 35
Number of stages	Small: 2, 4 Large: 2, 4
Number of machines	Small: U(2,4) Large: U(2,4)
Processing times	Small: U(10,50) Large: U(10,100)
Release dates	U(1,100)
Due dates	$\frac{\sum_{i=1}^{No.job} No.job_i \times \sum_{s=1}^{No.stage} p_{ij} + round\left(\frac{U(1, \sum_{s=1}^{No.stage} p_{ij})}{\sum_{s=1}^{No.stage} No.machines_s}\right)}{\sum_{i=1}^{No.job} No.job_i}$
Time limitation of machines	Small: U(50,100) Large: U(1000,10,000)
Weight of earliness	U(1,5)
Weight of tardiness	U(1,5)
Weight of rejection	U(30,100)
Weight of incompleteness	U(1,5)

It is necessary to notice that the defined formula for generating due dates in Table 2 is an adopted version of the presented formula by Jolai et al. (2013) for the same goal.

To conduct the experiments, one problem which is randomly generated is selected and is run five times to reduce the error (Table 5).

Table 3 CSA algorithm factors and their levels

Parameters	A	B	C
Name	γ	T_0	T_f
	0.8	1	0.0001
Levels	0.9	0.1	0.00001
	0.95	0.01	0.000001

Table 4 The orthogonal array L9

	A	B	C
1	1	1	1
2	1	2	2
3	1	3	3
4	2	1	2
5	2	2	3
6	2	3	1
7	3	1	3
8	3	2	1
9	3	3	2

5.2 Algorithm calibration

Parameters may influence on the algorithm performance. Considering the effect of parameters, in recent years the algorithm configuration is attracted wide attention. There are many ways to design experiments (Wang and Liu 2013), but the most frequency used approach is Taguchi method. In this study, the factors of CSA algorithm are γ, T_0, T_f , respectively, which their levels are given in Table 3. The considered orthogonal array with three factors and three levels of Taguchi method (L9) is presented in Table 4.

As shown in Fig. 8 and Table 5, the optimal levels of the factors are 1, 3 and 2, respectively.

Table 5 Average S/N ratio table

Level	A	B	C
1	-54.97200526	-54.84013317	-54.94168776
2	-54.94745099	-54.88471646	-54.95008187
3	-54.92747033	-55.12207695	-54.95515696

Table 6 Small-scale test problems

Problem	Number of families	Number of producible jobs	Maximum jobs can be ordered	Number of stages
1	1	2	5	2
2	1	2	5	4
3	1	2	10	2
4	1	2	10	4
5	1	4	5	2
6	1	4	5	4
7	1	4	10	2
8	1	4	10	4
9	2	2	5	2
10	2	2	5	4
11	2	2	10	2
12	2	2	10	4
13	2	4	5	2
14	2	4	5	4
15	2	4	10	2
16	2	4	10	4

Fig. 8 The mean S/N ratio plot for each level of CSA factors

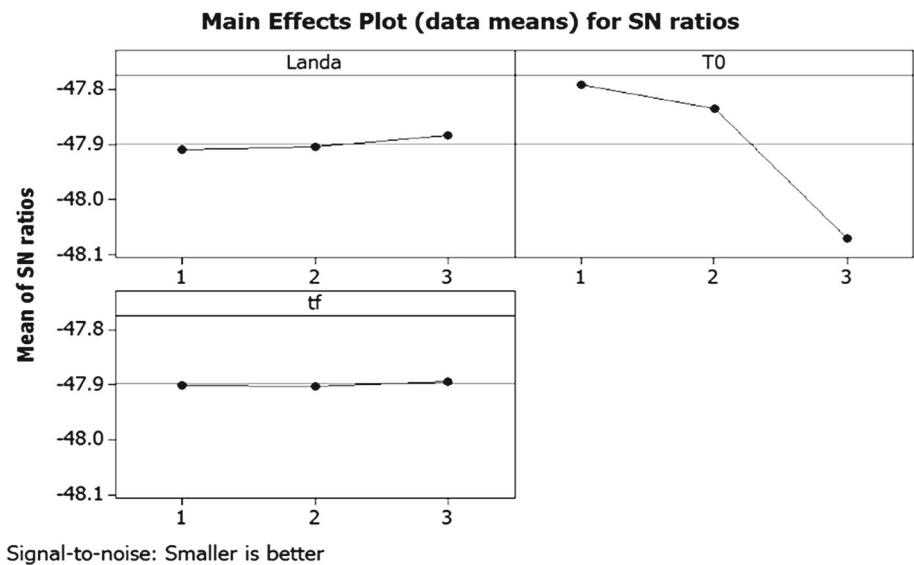


Table 7 Large-scale test problems

Problem	Number of families	Number of producible jobs	Maximum jobs can be ordered	Number of stages
1	3	3	25	2
2	3	3	25	4
3	3	3	35	2
4	3	3	35	4
5	3	5	25	2
6	3	5	25	4
7	3	5	35	2
8	3	5	35	4
9	5	3	25	2
10	5	3	25	4
11	5	3	35	2
12	5	3	35	4
13	5	5	25	2
14	5	5	25	4
15	5	5	35	2
16	5	5	35	4

5.3 The results

In this section, two evaluations are presented. At first, to examine the effectiveness of the proposed solution approaches (AIS and CSA), 32 benchmark problems are run by each algorithm. Then, to analysis the influences of each parameter, one of the test problems is selected and the sensitive of percentage of MTS/MTO and order

family (N)/*SUM* (number of orders/number of jobs in each order) on performance measurements are evaluated.

It is remarkable that in this study the algorithms are coded in MATLAB 2012 and the mathematical model is coded by Lingo 9. All of them are run on a PC with an Intel Core 4 Duo 2.5GHz CPU with 4GB of RAM under a Windows 8 environment. Based on suggested approach by Ramezani et al. (2013), algorithms are run under their termination condition and their average of computational time is saved. Since the average computational time of AIS is bigger than the average computational time of CSA, to compare the performances, AIS algorithm is run for each test problem 5 times and then the CSA algorithm is run at the average of these times. The relative percentage deviation (RPD) is used as the performance measurement. For each test problem, these algorithms are run 5 times for each instance. The best solution is denoted by $Best_{sol}$. So RPD is calculated by following formula:

$$RPD = \frac{Alg_{sol} - Best_{sol}}{Best_{sol}} \times 100 \quad (23)$$

Where Alg_{sol} is the objective function value obtained for a given algorithm and test problem. The average relative percentage error is defined according to Eq. 24.

$$ARPD = \frac{\sum_{i=1}^{Number\ of\ runs} RPD_i}{Number\ of\ runs} \quad (24)$$

The effectiveness of the algorithms is evaluated by solving 32 different problems. These problems are given in Tables 6 and 7. In addition, the solutions obtained for these problems

Table 8 The ARPD results of the small-scale problems

Problem	$Best_{sol}$	$Best_{AIS}$	AIS	$Best_{CSA}$	CSA	Time spent (s)	Lingo	Time spent of lingo (s)
1	50	50	0.356	50	0.004	0.4730794	4	3610
2	8	48	7.75	8	4.2	1.275652	14	4025
3	8.66	56	7.38	8.66	3.03	2.9911912	–	–
4	3.77	4	18.15	3.77	7.78	4.3755882	–	–
5	3.57	51.14	37.83	3.57	8.83	2.5293084	26.17	42,030
6	2.37	30.33	67.76	2.37	9.34	7.2843404	83.38	4218
7	9.5	45.57	6.42	9.5	3.32	15.364154	–	–
8	7.33	40	9.61	7.33	3.94	39.135422	–	–
9	25.9	25.9	0.50	56.4	2.01	1.4850138	–	–
10	64.5	75.5	0.75	64.5	0.35	8.8557552	–	–
11	61	91.33	1.27	61	0.59	25.572245	–	–
12	70	87.57	0.96	70	0.37	14.375928	–	–
13	65	121	1.24	65	0.43	24.645037	–	–
14	81	116	1.64	81	0.17	110.08146	–	–
15	59	97	1.52	59	0.39	115.061568	–	–
16	71	139	1.19	71	0.23	158.16730	–	–

Bold values are the best obtained solutions by the presented algorithms

Table 9 The ARPD results of the large-scale problems

Problem	<i>Best_{sol}</i>	<i>Best_{AIS}</i>	AIS	<i>Best_{CSA}</i>	CSA	Time spent
1	81	81	152.1975309	116	73.77777778	757.0815828
2	149	149	59.43624161	195	35.27516779	836.0687778
3	60	60	102.1333333	266	29.2	2161.440536
4	65	69	158.6461538	65	119.0153846	4359.864051
5	158	158	30.43037975	306	4.405063291	648.3888726
6	224	224	24.85714286	315	1.178571429	1169.933452
7	82	82	119.6097561	309	12	4715.734246
8	165	165	31.56363636	303	3.345454545	7535.91165
9	57	57	266.3859649	120	107.5087719	1341.900605
10	54	60	830.0740741	54	161.1851852	1765.831126
11	44	44	107.6363636	101	251.4545455	9413.971643
12	47	47	132.9361702	48	520.3404255	11634.75708
13	156	156	47.12820513	201	107.4358974	8469.997937
14	236	286	118.2033898	236	60.40677966	12917.0801
15	192	192	56.41666667	246	103.3333333	9019.580586
16	286	286	64.95104895	489	70.34965035	11228.37623

Bold values are the best obtained solutions by the presented algorithms

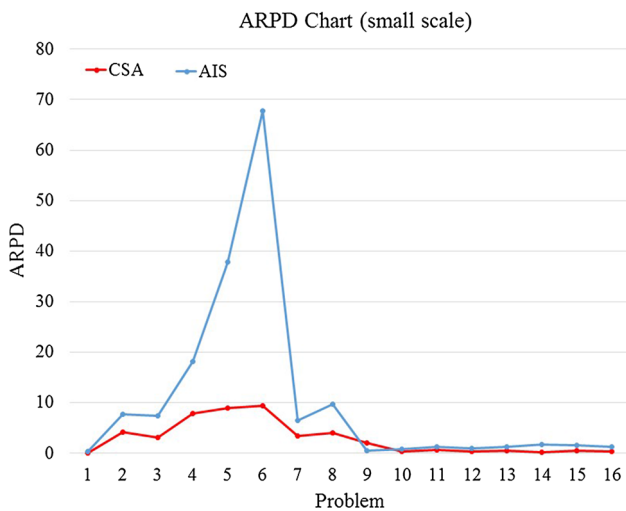


Fig. 9 The comparison between the ARPD of CSA and AIS algorithms in small-scale problems

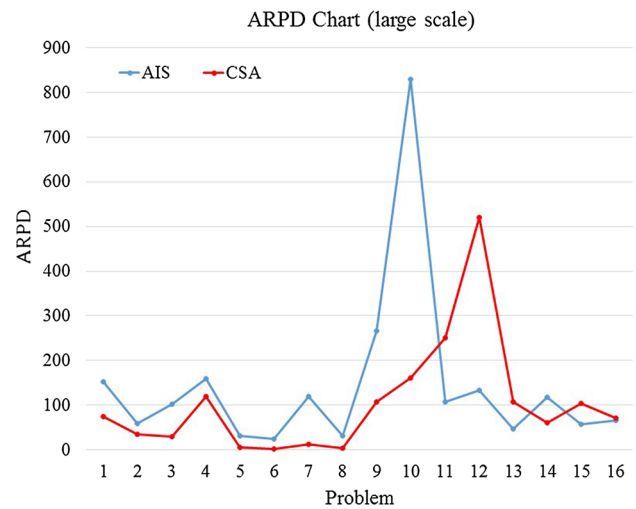


Fig. 10 The comparison between the ARPD of CSA and AIS algorithms in large-scale problems

by the proposed algorithms are summarized in Tables 8 and 9 and Figs. 9 and 10.

Table 8 and Fig. 9 show that for small-scale problems, CSA algorithm has been yielded the better results ($\frac{15}{16} \times 100 = 93.75\%$) than another algorithm.

Table 9 and Fig. 10 illustrate the results of experiments for the large-scale problems. As can be seen, CSA algorithm has been yielded the better results in $\frac{11}{16} \times 100 = 68.75\%$ of the problems. Also, AIS algorithm is obtained the better solutions in $\frac{5}{16} \times 100 = 31.25\%$ of the problems.

It is clear from Tables 8 and 9 the performance of CSA algorithm for small-scale problems is better than its perfor-

mance for large-scale problems. Unlikely, by increasing the size of the problems, the performance of AIS algorithm will be improved.

Tables 8 and 9 show that for 93.75% of small-scale problems the obtained *Best_{sol}* is equal to the *Best_{CSA}*. For 81.25% of the large-scale problems, the *Best_{sol}* of algorithms is equal to the *Best_{AIS}*. It means that for small-scale problems CSA performs better than AIS and for large-scale problems AIS performs better than CSA. In addition, Figs. 11 and 12 show that the proposed meta-heuristic algorithms are sensitive to the number of stages and the maximum number of jobs. Moreover, Fig. 13 demonstrates the convergence speed of algorithms in the same iteration. As shown in Fig. 13,

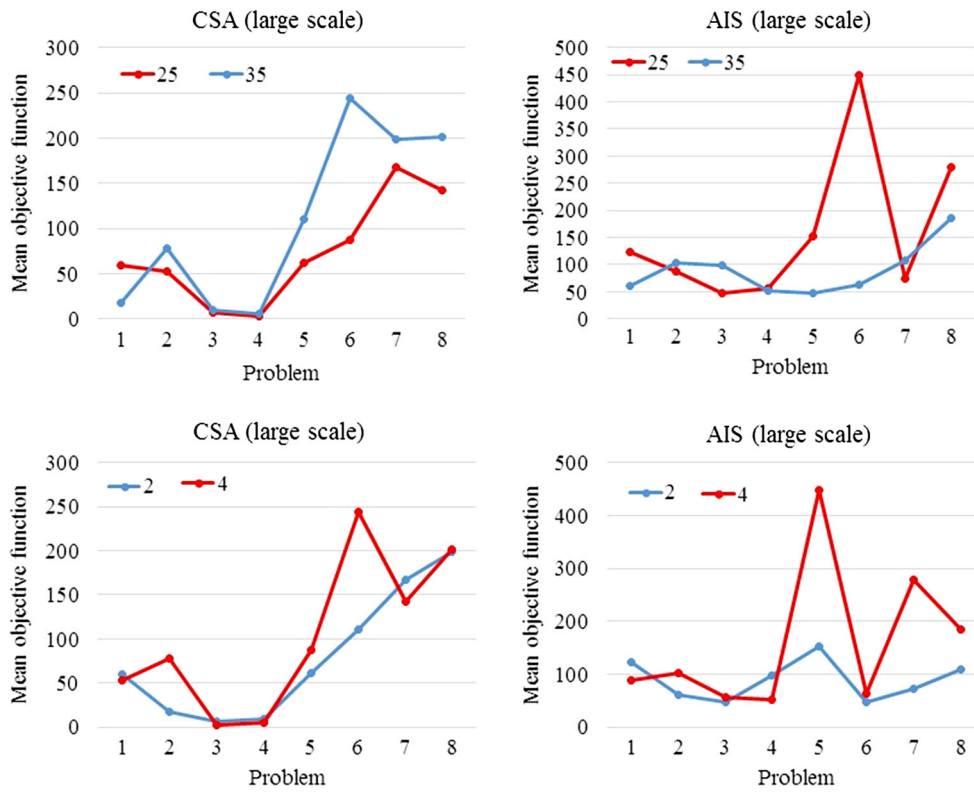


Fig. 11 The interaction between stage and performance of algorithms

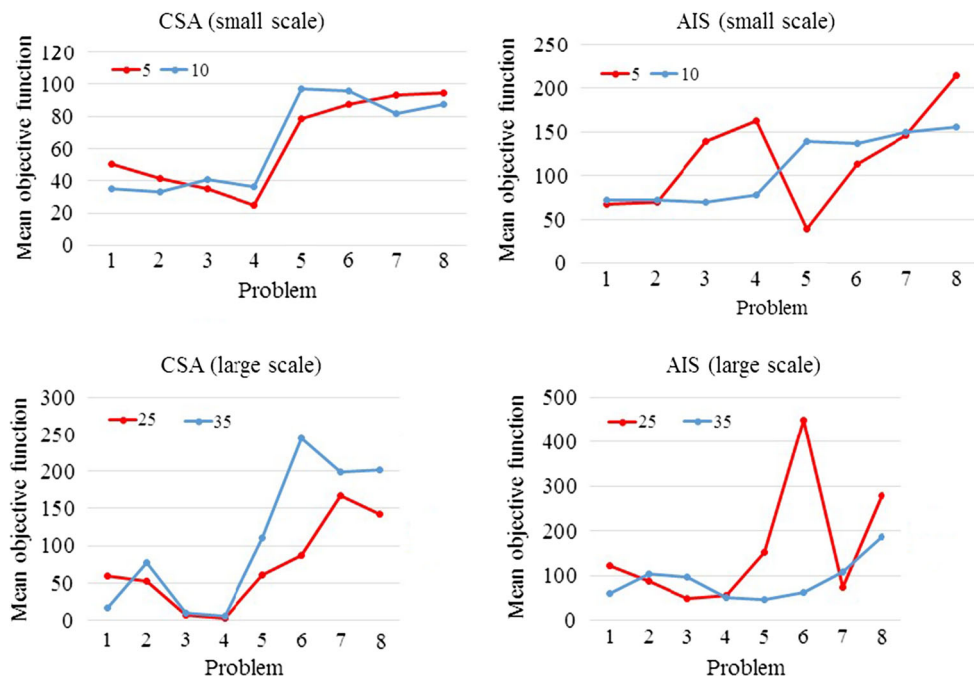


Fig. 12 The interaction between maximum number of jobs and performance of algorithms

AIS has higher convergence speed and is obtained the better solutions than the CSA algorithm in the same iteration number.

One of the important factors that affect decisions is the ratio of MTS/MTO. Generally, in order-based industries, the demand of orders from MTS policy is considered less than

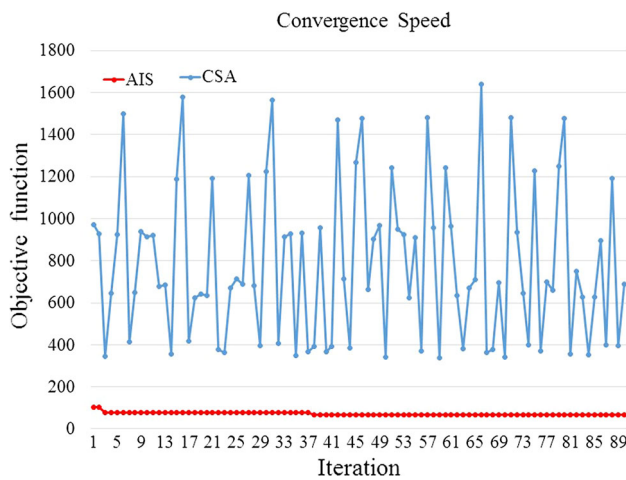


Fig. 13 The convergence speed of algorithms for problem 1 of large-scale problems

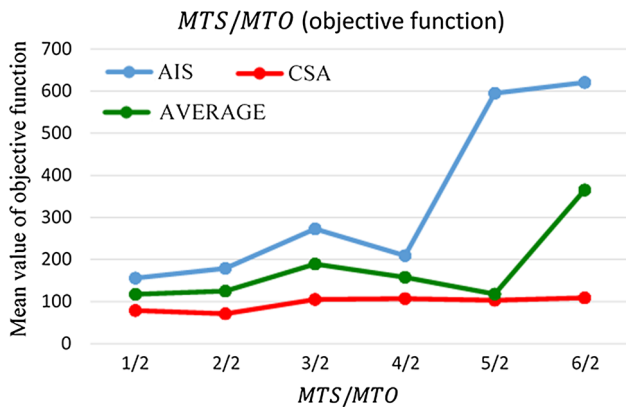


Fig. 14 The influence of MTS/MTO ratio on the objective function

MTO policy. Here, for analyzing the impact of MTS/MTO ration, a test problem is selected and the results are shown in Figs. 14 and 15.

According to Fig. 14, by increasing the ratio of MTS/MTO, because of limitation of machines capacity some of MTO jobs will be accepted and most of them should be rejected, and the value of objective function will be dropped.

Another factor that could be effect on performance measurements is the percentage of order family(N)/SUM. It will be asked that accepting more orders with less jobs is appropriate or accepting less orders with more jobs. This is one of the most important problems to achieve the balance between keeping market portion and on-time delivery of orders. Fig-ure 16 illustrates the selection of the second approach.

Figure 16 shows clearly that in the same number of orders when the number of jobs is increased, the objective function will be increased. On the other hand, in the same number of jobs, increasing the number of orders can be improved the objective function value. These results are predictable. Because, as mentioned above, the available time of machines is constant. So, when the number of orders is increased while

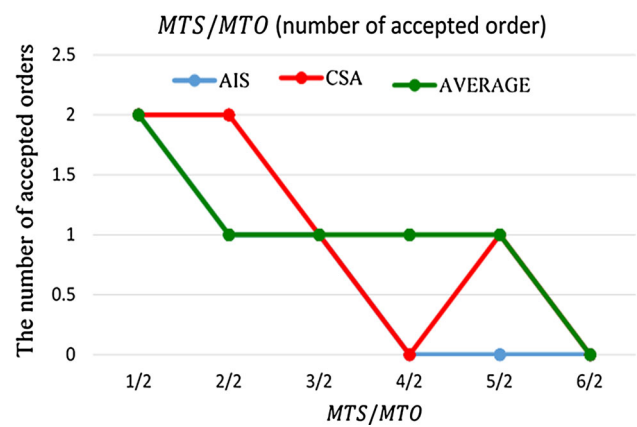


Fig. 15 The influence of MTS/MTO ration on the number of accepted order

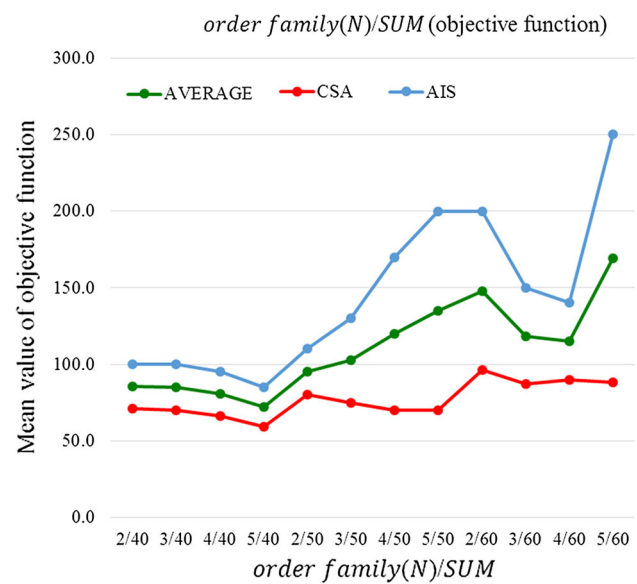


Fig. 16 The influence of order family(N)/SUM ratio on the objective function

the number of ordered jobs becomes constant or less, the number of accepted orders becomes increased and the objective function gets better. Unlikely, increasing the number of jobs when the number of orders becomes constant, the number of accepted orders becomes less, and in result, the objective function value gets worse.

Due to the above results, for every manufacturing system accepting more orders with less number of jobs may be better than accepting fewer orders with more number of jobs. It is obviously because increasing the accepted orders not only has more financial profit but also improves the company's image. It is remarkable note that this approach can be useful in the specified situation which the rejection costs of all orders are nearly equal to it.

It is important note that in Figs. 14, 15 and 16 the AVERAGE curve illustrates the average of obtained results by both algorithms.

6 Conclusion

This paper considered the no-wait hybrid flow shop with time limitation of machines and mixed MTO/MTS production management strategy. Due to the wide application of different variants of no-wait flow shop system in the various industries such as steel factories, plastic industries and chemical processes, studying on this system is very important. This study can be useful and valid for all above-considered industries. A new mathematical model is proposed, and as the solution approaches, two new meta-heuristics are proposed for this problem: the artificial immune system (AIS) as a population-based method and the cloudy-based simulated annealing (CSA) as a single-point approaches.

The results of experiments revealed that, however, CSA is obtained the better results but AIS has higher convergence speed.

In this study, all jobs are divided into MTO or MTS groups. The results show that increasing the MTO jobs decreases the MTS jobs production rate and this matter may reduce the market portion of company. In addition, increasing the production rate of MTS jobs causes the more orders are rejected, and in result their financial profits are lost and the worth of brand becomes less. So, to achieve balance between keeping market portion and increasing the financial and spiritual worth of company accepting the orders which have less jobs may be good approach.

This study can be useful for every industries which use flexible no-wait flow shop production line and produce MTO and MTS jobs separately or in a same time.

Several ideas deserve further investigation. First, the combined MTO/MTS policy is rarely studied, but it has great potential to develop this idea in other manufacturing systems. Second, in this study the MTO and MTS products have the same characteristics. But, in many modern industries the custom products may have different characteristics.

Compliance with ethical standards

Conflict of interest All authors declare that they have no conflicts of interest.

References

- Adan IJBF, Wall J (1998) Combining make to order and make to stock. *OR Spektrum* 20:73–81. doi:[10.1007/BF01539854](https://doi.org/10.1007/BF01539854)
- Arabameri S, Salmasi N (2013) Minimization of weighted earliness and tardiness for no-wait sequence-dependent setup times flowshop scheduling problem. *Comput Ind Eng* 64:902–916
- Ch YK, Lee ZJ, Lu CC, Lin SW (2012) Metaheuristics for scheduling a no-wait flowshop manufacturing cell with sequence-dependent family setups. *Int J Adv Manuf Technol* 58:671–682. doi:[10.1007/s00170-011-3419-y](https://doi.org/10.1007/s00170-011-3419-y)
- Cheng T, Li Z, Deng M, Xu Z (2005) Representing indeterminate spatial objects by cloud theory. In: *Proceedings of the 4th international symposium on spatial data quality*, pp 70–77
- Cheng T, Zhilin L, Deren L, Deyi L (2006) An integrated cloud model for measurement errors and fuzziness. In: *Proceedings of the 12th international symposium on spatial data handling*, pp 699–718
- Coello Coello CA, Cruz Cort'es N (2005) Solving multi objective optimization problems using an artificial immune system. *Genet Program Evol Mach* 6:163–190
- Davendra D, Zelinka I, Bialic-Davendra M, Senkerik R, Jasek R (2013) Discrete self-organising migrating algorithm for flow-shop scheduling with no-wait makespan. *Math Comput Model* 57:100–110
- Deyi L, Haijun M, Xuemei S (1995) Membership clouds and membership cloud generators. *J Comput Res Dev* 32(6):15–20
- Deyi L, Cheung D, Shi XM, Ng V (1998) Uncertainty reasoning based on cloud models in controllers. *Comput Math Appl* 35(3):99–123
- Deyi L, Di K, Deren L (2000) Knowledge representation and uncertainty reasoning in GIS based on cloud theory. In: *Proceedings of the 9th international symposium on spatial data handling*, vol 3a, pp 3–14
- Deyi L, Yi D (2005) *Artificial intelligence with uncertainty*. Chapman & Hall, London
- Deyi L, Yi D (2005) *Artificial intelligence with uncertainty*. Chapman & Hall, London
- Di K, Deyi L, Deren L (1999) Cloud theory and its applications in spatial data mining knowledge discovery. *J Image Graph* 4A(11):930–935
- Eivazy H, Rabbani M, Ebadian M (2009) A developed production control and scheduling model in the semiconductor manufacturing systems with hybrid make-to-stock/make-to-order products. *Int J Adv Manuf Technol* 45:968–986. doi:[10.1007/s00170-009-2028-5](https://doi.org/10.1007/s00170-009-2028-5)
- Fang W, Yanpeng L, Xiang L (2007) A new performance evaluation method for automatic target recognition based on forward cloud. In: *Proceedings of the Asia Simulation Conference*, pp 337–345
- Federgruen A, Katalan Z (1999) The impact of adding a make-to-order item to a make-to-order production system. *Manage Sci* 45(7):980–994. doi:[10.1287/mnsc.45.7.980](https://doi.org/10.1287/mnsc.45.7.980)
- Feizhou Z, Yuezu F, Chengzhi S, Deyi L (1999) Intelligent control based membership cloud generators. *Acta Aeronaut Astronaut Sin* 20(1):89–92
- Framinan JM, Nagano MS, Moccellini JV (2010) An efficient heuristic for total flow time minimisation in no-wait flow shops. *Int J Adv Manuf Technol* 46:1049–1057
- Framinan JM, Nagano MS (2008) Evaluating the performance for makespan minimisation in no-wait flow shop sequencing. *J Mater Process Technol* 197:1–9
- Gangadharan R, Rajendran C (1993) Heuristic algorithms for scheduling in the no-wait flow shop. *Int J Prod Econ* 32:285–290
- Hadj Youssef K, Van Delft C, Dallery Y (2004) Efficient scheduling rules in a combined make-to stock and make-to order manufacturing system. *Ann Oper Res* 126:103–134
- Haijun W, Yu D (2007) Spatial clustering method based on cloud model. *Proceedings of the fourth international conference on fuzzy systems and knowledge discovery*, vol 7, pp 272–276
- Hall NG, Sriskandarajah C (1996) A survey of machine scheduling problems with blocking and no-wait in process. *Oper Res* 44:510–525
- Javadi B, Saidi-Mehrabad M, Haji A, Mahdavi I, Jolai F, Mahvadi-Amiri N (2008) No-wait flow shop scheduling using fuzzy multi-objective linear programming. *J Franklin Inst* 345:452–467
- Jolai F, Sheikh S, Rabbani M, Karimi B (2009) A genetic algorithm for solving no-wait flexible flow lines with due window and job rejection. *Int J Adv Manuf Technol* 42:523–532. doi:[10.1007/s00170-008-1618-y](https://doi.org/10.1007/s00170-008-1618-y)

- Jolai F, Asefi H, Rabiee M, Ramezani P (2013) Bi-objective simulated annealing approaches for no-wait two-stage flexible flow shop scheduling problem. *Sci Iran E* 20(3):861–872
- Kaichang D, Deyi L, Deren L (1998) Knowledge representation and discovery in spatial databases based on cloud theory. *Int Arch Photogramm Remote Sens* 32(3/1):544–551
- Khoo LP, Situmdrang TD (2003) Solving the assembly configuration problem for modular products using an immune algorithm approach. *Int J Prod Res* 41(15):3419–3434
- King JR, Spachis AS (1980) Heuristics for flow shop scheduling. *Int J Prod Res* 18:343–357
- Laarhoven V, Aarts E (1987) *Simulated annealing: theory and applications*. Kluwer Academic, Norwell 186pp
- Laha D, Sapkal SU (2014) An improved heuristic to minimize total flow time for scheduling in the m-machine no-wait flow shop. *Comput Ind Eng* 67:36–43
- Li X, Yalaoui F, Amodeo L, Chehade H (2012) Metaheuristics and exact methods to solve a multiobjective parallel machines scheduling problem. *J Intell Manuf* 23(4):1179–1194
- Lin SW, Ying KC (2013) Minimizing makespan in a blocking flow shop using a revised artificial immune system algorithm. *Omega* 41:383–389
- Liu G, Song S, Wu C (2013) Some heuristics for no-wait flowshops with total tardiness criterion. *Comput Oper Res* 40:521–525
- Liu Y, Feng Z (2014) Two-machine no-wait flowshop scheduling with learning effect and convex resource-dependent processing times. *Comput Ind Eng* 75:170–175
- Lv P, Yuan L, Zhang J (2009) Cloud theory-based simulated annealing algorithm and application. *Eng Appl Artif Intell* 22:742–749
- Nunes de castro L, Von Zuben FJ (1999) *Artificial immune systems. Part I: Basic theory and applications*, Technical Report TR-DCA 01/99. FEEC/UNICAMP, Brazil
- Pan QK, Wang L, Qian B (2009) A novel differential evolution algorithm for bi-criteria no-wait flow shop scheduling problems. *Comput Oper Res* 36:2498–2511
- Pang KW (2013) A genetic algorithm based heuristic for two machine no-wait flow shop scheduling problems with class setup times that minimizes maximum lateness. *Int J Prod Econ* 141:127–136
- Qian B, Wang L, Hu R, Huang DX, Wang X (2009) A DE-based approach to no-wait flow-shop scheduling. *Comput Ind Eng* 57:787–805
- Rajendran C (1994) A no-wait flow shop scheduling heuristic to minimize makespan. *J Oper Res Soc* 45:472–478
- Ramezani P, Rabiee M, Jolai F (2013) No-wait flexible flowshop with uniform parallel machines and sequence-dependent setup times: a hybrid meta-heuristic approach. *J Intell Manuf*. doi:10.1007/s10845-013-0830-2
- Reddi SS, Rama-moorthy CV (1972) On the flow shop sequencing problems with no-wait intermediate queues. *Oper Res Q* 23:323–331
- Ruiz R, Allahverdi A (2007) No-wait flow shop with separate setup times to minimize maximum lateness. *Int. J. Adv. Manuf. Technol.* 35:551–565
- Seido Nagano M, Hissashi Miyata H, Castro Araújo D (2015) A constructive heuristic for total flowtime minimization in a no-wait flowshop with sequence-dependent setup times. *J Manuf Syst* 36:224–230
- Seido Nagano M, Almeida da Silva A (2012) A new evolutionary clustering search for a no-wait flow shop problem with set-up times. *Eng Appl Artif Intell* 25:1114–1120
- Shabtay D, Arviv K, Stern H, Edan Y (2014) A combined robot selection and scheduling problem for flow-shops with no-wait restrictions. *Omega* 43:96–107
- Shuliang W, Deren L, Wenzhong S, Deyi L, Xinzhou W (2003) Cloud model-based spatial data mining. *Geogr Inf Sci* 9(2):67–78
- Solimanpur M, Vrat P, Shankar R (2004) A neuro-tabu search heuristic for flow shop scheduling problem. *Comput Oper Res* 31:2151–2164
- Soman CA, Van Donk DP, Gaalman G (2004) Combined make-to-order and make-to-stock in a food production system. *Int J Prod Econ* 90(2):223–235. doi:10.1016/S0925-5273(02)00376-6
- Tavakkoli-Moghaddam R, Rahimi-Vahed AR, Mirzaei AH (2008) Solving a multi-objective no-wait flow shop scheduling problem with an immune algorithm. *Int J Adv Manuf Technol* 36:969–981
- Torabzadeh E, Zandieh M (2010) Cloud theory-based simulated annealing approach for scheduling in the two-stage assembly flowshop. *Adv Eng Softw* 41:1238–1243
- Tseng LY, Lin YT (2011) A hybrid genetic algorithm for no-wait flow shop scheduling problem. *Eur J Oper Res* 210:39–47
- Valente Jorge MS, Alves Rui AFS (2005) An exact approach to early/tardy scheduling with release dates. *Comput Oper Res* 32:2905–2917
- Wang C, Li X, Wang Q (2010) Accelerated tabu search for no-wait flowshop scheduling problem with maximum lateness criterion. *Eur J Oper Res* 206:64–72
- Wang S, Liu M (2013) A genetic algorithm for two-stage no-wait hybrid flow shop. *Comput Oper Res* 40:1064–1072
- Yingjun W, Zhongying Z (2004) Pattern mining for time series based on cloud theory pan-concept-tree. In: *International conference on rough sets and current trends in computing*, vol 1, pp 618–623
- Yunfang Z, Chaohua D, Weirong C (2005) Adaptive probabilities of crossover and mutation in genetic algorithms based on cloud generators. *J Comput Inf Syst* 1(4):671–678
- Zaerpour N, Rabbani M, Gharehgozli AH, Tavakkoli-Moghaddam R (2009) A comprehensive decision making structure for partitioning of make-to-order, make-to-stock and hybrid products. *Soft Comput* 13:1035–1054. doi:10.1007/s00500-008-0377-x
- Zandieh M, Fatemi GHomi SMT, Moattar Husseini SM (2006) An immune algorithm approach to hybrid flow shops scheduling with sequence-dependent setup times. *Appl Math Comput* 180(1):111–127