Scientia Iranica E (2021) 28(3), 1853–1870

Sharif University of Technology
Scientia Iranica
Transactions E: Industrial Engineering
http://scientiairanica.sharif.edu

Flexible flow shop scheduling problem to minimize makespan with renewable resources

N. Abbaszadeh, E. Asadi-Gangraj*, and S. Emami

Department of Industrial Engineering, Babol Noshirvani University of Technology, Babol, Iran.

Received 18 May 2019; received in revised form 4 August 2019; accepted 2 September 2019

KEYWORDS
Flexible flow shop;
MILP model;
Renewable resources;
Particle Swarm Optimization (PSO);
Simulated annealing.

Abstract. This paper deals with a Flexible Flow Shop (FFS) scheduling problem with unrelated parallel machines and a renewable resource shared among the stages. The FFS scheduling problem is one of the most common manufacturing environments, in which there is more than a machine in at least one production stage. In such a system, to decrease the processing times, additional renewable resources are assigned to the jobs or machines, which can lead to a decrease in the total completion time. For this purpose, a Mixed Integer Linear Programming (MILP) model is proposed to minimize the maximum completion time (makespan) in an FFS environment. The proposed model is computationally intractable. Therefore, a Particle Swarm Optimization (PSO) algorithm, as well as a hybrid PSO and Simulated Annealing (SA) algorithm named SA-PSO, are developed to solve the model. Through numerical experiments on randomly generated test problems, the authors demonstrate that the hybrid SA-PSO algorithm outperforms the PSO, especially for large size test problems.

1. Introduction

Scheduling is a resource allocation process used in activities by considering operational limitations to optimize one or more objective functions. The effective allocation of resources to the activities leads to enhancement of the performance of the manufacturing and service systems, and is considered a necessity for survival in today's competitive market. In the current competitive market, organizations are faced with new changes every day. Therefore, they must utilize an appropriate scheduling program. It can lead to effective use of resources, decreasing costs, increasing efficiency in the production of goods and services, and to satisfy customer expectation [1].

The Flexible Flow Shop (FFS) scheduling problem is a developed form of the general flow shop problem where one or more unrelated parallel machines exist at different stages [2]. Since the FFS scheduling problem must also determine job allocation to the machines, it is more complex than the flow shop scheduling problem. In the FFS scheduling problem with unrelated parallel machines, the processing time of the jobs at each stage is different, and dependent on the type of machine. It is considered a difficult production environment due to its high complexity [3].

In the lean production philosophy, it is very important to break the processing bottleneck to enhance productivity [1]. One of the important issues for achieving this goal is to consider renewable resources to reduce processing time, especially in the bottleneck stages. It means that the processing time of a job on a particular machine is dependent on the number of allocated renewable resources. It can lead to reduction in processing time and to, finally, decrease job com-

*. *Corresponding author. Tel.: +98 1135501814*
E-mail address: e.asadi@nit.ac.ir (E. Asadi-Gangraj)

doi: 10.24200/sci.2019.53600.3325

pletion time. In the context of renewable resources, jobs require these resources to process besides the machines. After finishing its processing, the job returns the allocated resources and they can be used by other jobs on other machines. By allocating renewable resources, job processing time is reduced with respect to the number of allocated renewable resources.

In many real-world manufacturing systems, rather than a machine, renewable resources such as human resources and molds are available and shop-floor managers can assign them to the jobs/machines. Due to the complexity of the scheduling problems with renewable resources, most scheduling approaches neglect this concept or solve the scheduling problem and renewable resources assignment, separately. Assigning the renewable resource leads to speed-up the processing operations. It is a critical issue and can enhance the efficiency of the scheduling problems, and finally, the overall performance of the production line. As a result, this issue is considered of great importance in the classical shop scheduling problem. In other words, considering renewable resources and job scheduling, jointly, can lead to generating better solutions. In the simultaneous job scheduling and renewable resource assignment problems, job completion times are affected by two main decisions: job sequencing and scheduling and the optimal assignment of renewable resources to each machine at each stage.

There are many applications of the FFS scheduling problem with renewable resources in the real world, but little research has been undertaken regarding this problem. In this research, the FFS scheduling problem, with unrelated parallel machines and renewable resources, is conducted and an Mixed Integer Linear Programming (MILP) model is developed to minimize maximum completion time (makespan). The research problem is computationally intractable and strongly NP-hard, therefore, two metaheuristic algorithms, including Particle Swarm Optimization (PSO) and hybrid SA-PSO, are developed to solve the research problem.

The outline of the paper is organized as follows: The next section briefly reviews the literature. The proposed mathematical model is illustrated in Section 3. Section 4 represents the metaheuristic algorithms that are proposed to solve the problem. Section 5 gives the obtained results and, finally, in Section 6, discussion and some suggestions for future research are offered.

2. Literature reviews

2.1. Metaheuristic algorithms for FFS scheduling problems

Due to the NP-hardness of the FFS scheduling problem [4], many researchers have proposed different metaheuristic approaches for this problem. Zabihzadeh and Rezaeian [5] considered an FFS scheduling problem with release time and robotic transportation. They presented a Genetic Algorithm (GA) and Ant Colony Optimization (ACO) for this problem. Karimi et al. [6] suggested a GA to solve a multi-objective FFS scheduling problem. Shahvari and Logendran [7] considered a bi-objective FFS batch scheduling problem with machine-dependent and sequence-dependent family setup times with various assumptions such as machine availability constraints, ready time, and learning effect. They presented two stage-based metaheuristic algorithms based on local search and population-based structures. Almeder and Hartl [8] considered a stochastic FFS scheduling problem with limited buffers. They proposed a solution approach based on a variable neighborhood search. Besbes et al. [9] focused on the FFS scheduling problem by considering availability constraints for machines. They proposed a GA-based approximation algorithm to minimize makespan.

Sangsawang et al. [10] dealt with a two-stage reentrant FFS scheduling problem with blocking constraint and makespan minimization. They developed two hybrid metaheuristic algorithms based on PSO and GA. The obtained results demonstrate that the hybrid PSO algorithm generates superior results. Jolai et al. [11] dealt with a bi-objective no-wait two-stage FFS scheduling problem to minimize maximum tardiness and makespan. They proposed three bi-objective metaheuristic algorithms based on the Simulated Annealing (SA) algorithm. Akrami et al. [12] proposed GA and Tabu Search (TS) for joint economic lot sizing and scheduling problems in the FFS environment, with respect to limited intermediate buffers.

Marichelvam, Prabaharan et al. [13] focused on a FFS scheduling problem with makespan minimization. They proposed an improved cuckoo search algorithm for this problem. Choong et al. [14] proposed two hybrid algorithms based on PSO, SA, and TS for the FFS scheduling problem. Chung and Liao [15] considered an FFS scheduling problem. They proposed an immunoglobulin-based artificial immune system algorithm to minimize makespan, and Dios et al. [16] proposed some heuristic algorithms in the FFS environment to minimize makespan.

2.2. Scheduling problems with renewable resources

During recent years, various research has examined renewable resources in scheduling problems. Behnamian and Fatemi Ghomi [17] considered FFS scheduling problem with resource-dependent processing times. The selected objective function minimizes the total resource allocation costs and makespan. They proposed a hybrid metaheuristic algorithm based on GA

and a Variable Neighborhood Search (VNS). They compared the proposed algorithm with random initial population SA [18]. The obtained results demonstrated that the hybrid approach is very efficient for different test problems. Edis and Oguz [19] studied a parallel machine scheduling problem with additional flexible resources to speed-up the production process. They presented an Integer Programming (IP) model and IP-based constraint programming for this problem.

Yin et al. [20] focused on unrelated parallel machine scheduling problems, in which resource-dependent processing time and deteriorating jobs are considered, simultaneously. They proposed a polynomial approach to minimize a cost-related objective function. Su and Lien [1] dealt with a parallel machine scheduling problem with resource-dependent processing time. The selected objective function aims to minimize makespan. They firstly proposed a heuristic to minimize the makespan, and two procedures, RA1 and RA2, to optimally allocate the renewable resources. Finally, they combined CL with RA1 and RA2 to solve the problem.

Figielska [21] considered a two-stage flow shop scheduling problem with a parallel machine and renewable resources at the first stage and a single machine at the second stage. A novel heuristic algorithm was, thus, developed to minimize makespan. In Figielska [22], the previous research was extended and dealt with a two-stage flow shop scheduling problem with parallel machines at both stages. Four heuristic algorithms using linear programming were proposed to minimize makespan. Figielska [23] also provided three metaheuristic approaches, TS, SA, and GA, to solve the research problem, considered in Figielska [22].

Li et al. [24] considered the scheduling problem in a parallel machine environment with the identical machine and resource-dependent processing time, so that processing time is a linearly decreasing function of the number of allocated resources. They proposed a SA algorithm to achieve near-optimal solutions. The results showed that the proposed algorithm has good performance in solution quality and CPU time. Liu and Feng [25] focused on a two-stage no-wait flow shop scheduling problem with a cost-related objective function. They considered position-based and resource-dependent processing time, and decomposed the research problem into two subproblems; optimal resource allocation, and an optimal sequencing problem. Kellerer [26] presented an approximation algorithm for an identical parallel machine scheduling problem. Resource-dependent processing time was considered with the objective of makespan minimization.

Jun et al. [27] dealt with a single machine scheduling problem with different assumptions, such as resource-dependent processing time, learning effect, and serial batch production. A limited number of total resources were imposed into the model and the makespan was minimized. Thus, a hybrid algorithm based on a GS algorithm and TS was developed to achieve high-quality solutions. Wang and Wang [28] considered the single machine scheduling problem with deteriorating jobs and convex resource-dependent processing times. It was also shown that the research problem is polynomially solvable with the cost-related objective function.

Wei and Ji [29] focused on a single machine problem with time-dependent and resource-dependent processing time. Different cost-based objective functions were considered for the research problem. Wang et al. [30] dealt with resource-dependent processing time and learning effect in a single machine environment. Two different processing time functions were considered and a polynomial time algorithm was developed to achieve optimal solutions.

Wang and Cheng [31] considered a single machine environment with respect to resource-dependent release time and processing time, each of which is a linearly decreasing function of the allocated resources. The selected objective function is to minimize makespan and the total consumed resource cost. A heuristic approach was also proposed based on some optimal properties. Nguyen et al. [32] studied a parallel machine scheduling problem with non-renewable resources. A hybrid approach, based on a differential evolution algorithm, an iterated greedy search, a MILP model, and parallel computing, was proposed for this problem.

To facilitate those papers considered in the literature review, some of them with a background of renewable resources are categorized in Table 1, with respect to some aspects. They are categorized based on type of production environment, objective function, resources (renewable, non-renewable), and the solution method.

Regarding Table 1, most of the research is studied in simple environments such as single machine and parallel machine environments. Furthermore, only one piece of research considered the multi-stage FFS problem with non-renewable resources. The other research in this environment focuses on the two-stage FFS. However, none of the reviewed articles address renewable resources in the multi-stage FFS environment and, to the best of our knowledge, consideration of renewable resources is not undertaken in the multi-stage FFS scheduling problem. Moreover, metaheuristic approaches in the context of renewable resources are rarely studied. As a result, the main contributions of the present research are:

- Considering renewable resources in the FFS scheduling problem;

Table 1. Different aspects of the related researches.

Author	Year	Environment	Objective function (s)	Resources		Solution method
				Renewable	Non-renewable	
Wang and Cheng [31]	2005	Single machine	Makespan and total number of consumed resource cost	–	*	Heuristic
Wang et al. [30]	2010	Single machine	Costs	–	*	Polynomial time algorithm
Wei and Ji [29]	2012	Single machine	Costs	–	*	Polynomial time algorithm
Wang and Wang [28]	2013	Single machine	Costs	–	*	Polynomial time algorithm
Jun et al. [27]	2018	Single machine	Makespan	–	*	Hybrid Gravitational Search algorithm & TS
Kellerer [26]	2008	Parallel machine	Makespan	–	*	Approximation algorithm
Li et al. [24]	2011	Parallel machine	Makespan	–	*	SA
Edis and Oguz [19]	2009	Parallel machine	Makespan	*		IP-based constraint programming
Yin et al. [20]	2014	Parallel machine	Costs	–	*	Polynomial time algorithm
Nguyen et al. [32]	2019	Parallel machine	Total weight tardiness	–	*	Hybrid approach
Liu and Feng [25]	2014	Flow shop	Costs	–	*	Decomposition approach
Figielska [21]	2008	Two-stage flexible flow shop	Makespan	*	–	Heuristic algorithm
Figielska [22]	2010	Two-stage flexible flow shop	Makespan	*	–	LP-based heuristic algorithm
Figielska [23]	2011	Two-stage flexible flow shop	Makespan	*	–	TS, SA, and GA
Behnamian and Fatemi Ghomi [17]	2011	Flexible flow shop	Makespan and total resource allocation costs	–	*	Hybrid GA&VNS
Present research	2019	Flexible flow shop	Makespan	*	–	PSO, Hybrid SA&PSO

- Developing an MILP model for the FFS problem with renewable resources;
- Proposing two metaheuristic approaches for this problem;
- Developing a heuristic approach to assign the renewable resources.

3. Mathematical model of the research problem

This section is devoted to describing the studied problems more formally and introduces the assumptions, notations, and mathematical model.

3.1. Problem description

As mentioned above, an FFS scheduling problem is studied, in which a group of parallel machines is arranged into a number of stages in series. Assume that n different jobs require to be processed at different stages and all the jobs must be processed through all the stages. There are m^t unrelated parallel machines at stage t. Also, a number of renewable resources are considered in this research, which must be allocated to the machines at each stage. It is assumed that job processing times are dependent on the number of resources allocated to the jobs at each stage, which can lead to speeding up the processing of the jobs. The allocation can lead to decreasing the processing

times and, finally, the makespan. Regarding Gupta et al. [33], the normal processing times of the jobs decrease with respect to the number of allocated resources and reduction coefficient (see Eq. (8)).

3.2. Problem assumptions

The following assumptions are made in this research:

- There is no set-up time for the jobs and travel time between stages;
- Entire jobs and machines are available at zero time;
- There is no prerequisite constraint between the jobs, and they are independent of each other;
- There is no possibility of machine failure;
- There is an unlimited capacity for intermediate buffers;
- The machines are not the same at the stages (unrelated parallel machines);
- All programming parameters are deterministic;
- Each machine at each stage can handle only one job each time, and any job must be allocated to only one machine at each stage;
- The resources are renewable. This means that they can be used for different jobs and stages during the planning horizon.

3.3. Notation

The following notations are used to formulate the research problem.

Indices

t	Index of stages
i	Index of machines
j, j'	Index of jobs

Parameters

g	Number of stages
m^t	Number of unrelated parallel machines at stage t
n	Number of jobs
Np_{ij}^t	Normal processing time of job j at stage t on machine i
R^t	Number of renewable resources at stage t
a^t	Processing time reduction coefficient at stage t due to assigning renewable resources
U_{ij}^t	Maximum renewable resources that can be assigned to job j at stage t on machine i
M	A large positive number

Decision variables

p_{ij}^t	Modified processing time of job j on machine i at stage t after renewable resource allocation
r_{ij}^t	The number of renewable resources allocated to job j at stage t on machine i
x_{ij}^t	1 if job j is processed on machine i at stage t; 0 otherwise
$y_{ijj'}^t$	1 if job j' precedes job j on machine i at stage t; 0 otherwise
$\rho_{jj'}^t$	1 if completion time of job j is greater than or equal to the start time of job j' at stage t; 0 otherwise
C_j^t	Completion of job j at stage t
$C_{\max}$	Makespan

3.4. Mathematical model

$$\min Z = C_{\max}, \tag{1}$$

s.t.:

$$\sum_{i=1}^{m^t} x_{ij}^t = 1, \quad j = 1, 2, .., n; \quad t = 1, 2, ..., g, \tag{2}$$

$$c_j^1 \geq \sum_{i=1}^{m^1} p_{ij}^1 x_{ij}^1 \quad j = 1, 2, .., n; \quad t = 1, 2, ..., g, \tag{3}$$

$$c_j^t \geq c_j^{t-1} + \sum_{i=1}^{m^t} p_{ij}^t x_{ij}^t; \quad j = 1, 2, .., n \quad t = 2, ..., g, \tag{4}$$

$$c_{j'}^t + M\left(3 - y_{ijj'}^t - x_{ij}^t - x_{ij'}^t\right) \geq c_j^t + p_{ij'}^t x_{ij'}^t$$
$$i = 1, 2, .., m^t; \quad t = 1, 2, ..., g;$$
$$j, j' = 1, 2, .., n; \quad j \neq j', \tag{5}$$

$$c_j^t + M\left(2 + y_{ijj'}^t - x_{ij}^t - x_{ij'}^t\right) \geq c_{j'}^t + p_{ij}^t x_{ij}^t;$$
$$i = 1, 2, .., m^t; \quad t = 1, 2, ..., g;$$
$$j, j' = 1, 2, .., n \quad j \neq j', \tag{6}$$

$$c_j^g \leq C_{\max} \quad j = 1, 2, .., n, \tag{7}$$

$$p_{ij}^t = Np_{ij}^t - a^t r_{ij}^t;$$
$$i = 1, 2, .., m^t; \quad j = 1, 2, .., n; \quad t = 1, 2, ..., g, \tag{8}$$

$$r_{ij}^t \leq U_{ij}^t;$$
$$i = 1, 2, .., m^t \quad j = 1, 2, .., n \quad t = 1, 2, ..., g \tag{9}$$

$$M * \rho^t_{jj'} \geq C^t_j - \left(C^t_{j'} - \sum_{i=1}^{m^r} p^t_{ij'} x^t_{ij'} \right)$$
$$t = 1, 2, ..., g; \quad j, j' = 1, 2, .., n; \quad j \neq j', \tag{10}$$

$$\sum_{i=1}^{m^t} r^t_{ij} + \sum_{i=1}^{m^t} \sum_{j'=1}^{n} r^t_{ij'} \left(\rho^t_{jj'} + \rho^t_{j'j} - 1 \right) \leq R^t$$
$$t = 1, 2, ..., g; \quad j, j' = 1, 2, .., n; \quad j \neq j', \tag{11}$$

$$x^t_{ij}, y^t_{ijj'}, \rho^t_{jj'} \in \{0, 1\}$$
$$i = 1, 2, .., m^t; \quad t = 1, 2, ..., g;$$
$$j, j' = 1, 2, .., n; \quad j \neq j', \tag{12}$$

$$r^t_{ij} \in \{0, 1, 2, ...\}$$
$$i = 1, 2, .., m^t; \quad j = 1, 2, .., n; \quad t = 1, 2, ..., g, \tag{13}$$

$$C^t_j, p^t_{ij}, C_{\max} \geq 0$$
$$i = 1, 2, .., m^t; \quad j = 1, 2, .., n; \quad t = 1, 2, ..., g. \tag{14}$$

Expression (1) defines the maximum of completion time as the objective function. Constraint set (2) states that each machine must process only one job at every time. Constraint sets (3) and (4) determine the completion time of each job at the first and other stages, respectively. Constraint sets (5) and (6) are disjunctive constraints. They calculate the relation between the completion times of two jobs, which are processed on one machine at each stage. In one moment, at most, one of these two constraints is activated. If jobs j and j' are processed on machine i at stage t ($x^t_{ij} = x^t_{ij'} = 1$) and j is processed before j' ($y^t_{ijj'} = 1$), Constraint set (5) is activated. Conversely, if job j' is processed before job j ($y^t_{ijj'} = 0$), Constraint set (6) will be activated. Finally, if jobs j and j' are processed on different machines ($x^t_{ij} + x^t_{ij'} \leq 1$), both constraint sets are redundant. Constraint set (7) determines the makespan with respect to the completion times at the last stage. Constraint set (8) is incorporated into the model to calculate the modified processing time of each job after the resources allocation. Constraint set (9) represents the maximum renewable resources which can be allocated to each job at each stage. Constraint set (10) specifies the jobs that have overlap. If completion time of job j at stage t is greater than the start time of job j' ($\rho^t_{jj'} = 1$), and simultaneously, the competition time of job j' is greater than the start time of job j ($\rho^t_{j'j} = 1$), these jobs have overlap. Constraint set (11) shows that the total used resources for these jobs, which are processed simultaneously ($\rho^t_{jj'} + \rho^t_{j'j} = 2$), cannot exceed the maximum renewable resources at stage t. Finally, Constraint sets (12)–(14) show the range of the decision variables.

It is clear that the proposed mathematical model is nonlinear (with respect to Constraint sets (3)–(6), (10) and (11)). It is obvious that the nonlinear models are very time-consuming to achieve optimal solutions. Hence, an effort is made to change nonlinear terms into a linear form by substituting variable S^t_{ij} with $x^t_{ij} p^t_{ij}$ in Constraint sets (3)–(6) and (10), as well as $D^t_{ijj'}$ with $r^t_{ij'}(\rho^k_{jj'} + \rho^k_{j'j} - 1)$ in Constraint set (11). As a result, the linear formulation of these constraint sets are as follows:

$$S^t_{ij} + M(1 - x^t_{ij}) \geq p^t_{ij}$$
$$i = 1, 2, .., m^t; \quad j = 1, 2, .., n; \quad t = 1, 2, ..., g, \tag{15}$$

$$s^t_{ij} \leq M x^t_{ij}$$
$$i = 1, 2, .., m^t; \quad j = 1, 2, .., n; \quad t = 1, 2, ..., g, \tag{16}$$

$$s^t_{ij} \leq p^t_{ij}$$
$$i = 1, 2, .., m^t; \quad j = 1, 2, .., n; \quad t = 1, 2, ..., g, \tag{17}$$

$$D^t_{ijj'} + M(2 - \rho^t_{jj'} - \rho^t_{j'j}) \geq r^t_{ij'}$$
$$i = 1, 2, .., m^t; \quad t = 1, 2, ..., g;$$
$$j, j' = 1, 2, .., n; \quad j \neq j', \tag{18}$$

$$D^t_{ijj'} \leq M(\rho^t_{jj'} + \rho^t_{j'j} - 1)$$
$$i = 1, 2, .., m^t; \quad t = 1, 2, ..., g;$$
$$j, j' = 1, 2, .., n; \quad j \neq j', \tag{19}$$

$$D^t_{ijj'} \leq r^t_{ij'}$$
$$i = 1, 2, .., m^t; \quad t = 1, 2, ..., g;$$
$$j, j' = 1, 2, .., n \quad j \neq j'. \tag{20}$$

4. Methodology

Since the FFS scheduling problem with unrelated parallel machines is NP-hard, the research problem is also NP-hard in the strong sense; thus, achieving optimal solutions for medium to large-size problems is very time-consuming. As a result, two metaheuristic approaches based on PSO and SA are developed to minimize the makespan for the FFS scheduling problem with renewable resources. First, two proposed metaheuristic approaches are briefly described, and then, in order to calculate the objective function, a heuristic

approach is proposed to assign the jobs to machines, the sequence of jobs on each machine, and assign the renewable resources to the jobs at each stage.

Details of the proposed algorithms are described as follows.

4.1. PSO algorithm

The PSO algorithm is a population-based optimization technique which was introduced for the first time by Kennedy and Eberhart [34]. The main idea of this algorithm is based on simulation of animal social behavior, such as birds and fish which are living in a group [35]. It is assumed that a number of animals is seeking food in a random space. None of these animals have information regarding the location of the food and only instinctively feel their distance from it. Due to the relatively good performance in some scheduling problems, as well as the simple structure of the algorithm and the efficiency of its implementation, this algorithm is an effective approach to solve large-scale scheduling problems.

In the PSO algorithm, each particle moves around the solution space to obtain optimal/near-optimal solutions by updating its velocity and position based on two parts: cognitive and social. The following formulae are applied for updating the velocity and position of the particles at every iteration:

$$vel_i(k+1) = W \times vel_i(k) + C_1 \times r_1 \times (pbest_i - X_i(k)) + C_2 \times r_2 \times (gbest - X_i(k)), \quad (21)$$

$$X_i(k+1) = X_i(k) + vel_i(k+1), \quad (22)$$

where, W is called inertial weight and shows the impact of the previous velocity of the particle on its velocity in the next iteration. $vel_i(k)$ shows the velocity of particle i in iteration k, and $X_i(k)$ represents the position of particle i in the kth iteration. $pbest_i$ and $gbest$ are the best-known position vectors of particle i and the best location vector in the population for all the particles, respectively. Parameters C_1 and C_2 are acceleration coefficients with different constant values and which determine the influence of $pbest_i$ and $gbest$ on velocity, respectively. Two random numbers, r_1 and r_2, are incorporated in the structure of the PSO algorithm to add uncertainty.

4.2. Implementation of the PSO algorithm

4.2.1. Solution representation and jobs sequence

Solution representation in the form of a string of numbers, letters, or a combination of both is the first and perhaps one of the most important steps in applying and implementing metaheuristic algorithms. In the present research, the solution representation in the form of a string of numbers is a permutation of numbers in the interval of [1,n], so that, n indicates the number of jobs.

Index	1	2	3	4	5
Particle	0.85	0.64	0.12	0.53	0.26
Descending order	0.85	0.64	0.53	0.26	0.12
Jobs sequence	1	2	4	5	3

Figure 1. Random key method.

By considering the continuous space for the particles in the PSO algorithm, it needs to apply a heuristic approach to convert a particle in the continuous space to the one in the discrete space [36]. As a result, in this study, the Random Key (RK) method [37] is used to transform a particle in continuous space. In the RK method, the position of any particle in the RK virtual space (continuous space) is turned into a position in the problem space (discrete space). Consider 5 jobs, the initial sequence vector of jobs in the continuous space is (0.26, 0.53, 0.12, 0.64, 0.85), as represented in Figure 1, in a given iteration. Based on the RK method, the numbers in the sequence vector are arranged in descending order with an index related to each of these numbers. Figure 1 indicates how to achieve the jobs sequence in discrete space based on the vector in the continuous space. As can be seen, the corresponding sequence of the jobs is as (1, 2, 4, 5, 3).

4.2.2. Updating the particles

A multiplier χ was applied into the structure of Eq. (21). It leads to acceleration of the convergence process and enhances the overall performance of the PSO algorithm [38]. The desired value for χ is determined as follows:

$$C = C_1 + C_2 > 4, \quad (23)$$

$$\chi = \frac{2}{C - 2 + \sqrt{C^2 - 4C}}. \quad (24)$$

According to the above equation, the position and velocity of the particles will be updated based on the following formulae:

$$vel_i(k+1) = \chi \times \Big[W \times vel_i(k) + C_1 \times r_1 \times (pbest_i - X_i(k)) + C_1 \times r_1 \times (gbest - X_i(k))\Big], \quad (25)$$

$$X_i(k+1) = X_i(k) + vel_i(k+1). \quad (26)$$

Similar to Tadayon and Salmasi [39], Eq. (27) was applied to determine the value of W in each iteration. If W is set to a high value at the beginning of the procedure and is gradually reduced to a lower value, a better performance of the PSO algorithm can be obtained.

Initialization

Population size ($pop-size$); Maximum number of iterations ($MaxIt$); Maximum velocity (V_{max}); Learning factors (c_1 and c_2); Constriction coefficient (χ)

Generate the initial particles based on $pop-size$ and $t=1$

While $t \leq MaxIt$ **Do**

Calculate fitness value for every particle

Evaluate the initial particles to get the local best ($pbest$) and the global best ($gbest$):

If $F_k^t < F_k^{pbest} \rightarrow pbest_k(t) = X_k(t)$

If $F_k^{pbest} < F^{gbest} \rightarrow gbest(t) = pbest_k(t)$

Update the velocity and position of each particle

t=t+1

End While

Figure 2. The pseudo-code of the proposed Particle Swarm Optimization (PSO) algorithm.

$$W = W_{\max} - \frac{(W_{\max} - W_{\min}) \times iter}{MaxIt}, \tag{27}$$

in which, $W_{\max}$ and $W_{\min}$ are the upper and lower bounds for W, respectively, and $MaxIt$ is the total number of iterations that are accomplished in the PSO algorithm.

The pseudo-code of the PSO approach that is applied in this research is presented in Figure 2.

4.3. SA algorithm

The SA algorithm, or, in other words, the fusion/cooling algorithm was presented in the early 1980s by Kirkpatrick et al. [40]. During the SA process, a material is heated to a temperature higher than its melting temperature and then, its temperature is gradually lowered. The temperature reduction process is so slow that the material is, to some extent, in thermodynamic equilibrium. In other words, at any created temperature, the atoms can be replaced only to the extent of creating the greatest stability. This means that if the material is cooled even more slowly, the atoms will be able to release greater energy and locate in the direction of the greatest stability.

The SA algorithm is one of the first metaheuristic methods for searching neighborhood solutions that has an explicit strategy to avoid being trapped in local optimum solutions. In this approach, if the current solution has a better objective value than the last one, the current solution is accepted for the next iteration. Otherwise, it will be accepted as the current solution, with respect to the Boltzmann function, if:

$$\exp\left(\frac{f(X_k(t)) - f(X_k(t-1))}{T}\right) \geq P, \tag{28}$$

where, P is a random number in the interval $[0,1]$, $f(X_k(t))$ is the objective value of the solutions in the current iteration (t), $f(X_k(t-1))$ is the objective value of the current in the previous iteration and T is named as the temperature at which the current solution is evaluated. Note that T is a function of two input parameters: initial temperature and cooling rate [41].

4.4. Hybrid SA-PSO algorithm

Although the PSO algorithm has a relatively good performance in optimization problems, especially in scheduling problems, one of the major drawbacks of this algorithm is that it is easy to trap in the local optimum. Therefore, a combination of the PSO with other algorithms, such as SA, can solve this difficulty. As mentioned above, the SA algorithm is a well-known metaheuristic algorithms used to search for neighborhood solutions, such that it applies a high-performance strategy to prevent trapping in local optimum solutions. For this reason, in this research, a hybrid algorithm based on the PSO and SA, namely the SA-PSO algorithm, is proposed.

In the proposed procedure, if the *gbest* (*pbest*) of a particle has better performance (objective function), the new particle will be accepted, but if the *gbest* (*pbest*) is inferior, it may still be accepted with a positive probability, based on the Boltzmann function (Eq. (28)).

The pseudo-code of the hybrid SA-PSO algorithm is presented in Figure 3.

4.5. Calculation of the objective function

One of the most important aspects of metaheuristic approaches is to calculate the objective function based on the proposed solution representation. For this pur-

Initialization

Population size ($pop-size$); Maximum number of iterations ($MaxIt$); maximum velocity (V_{max}); Learning factors (c_1 and c_2); Constriction coefficient (χ); Initial temperature(T_0) ; Cooling rate (α)

Generate the initial particles based on $pop-size$ and t=1

While $t \leq MaxIt$ **Do**

Calculate fitness value for every particle

Evaluate the initial particles to get the local best ($pbest$) and the global best ($gbest$):

If $F_k^t < F_k^{pbest} \rightarrow pbest_k(t) = X_k(t)$

Else

Calculate $\Delta = F_k^t - F_k^{pbest}$

Generate a random number $r \in (0,1)$

If $r \leq (p = e^{-\frac{\Delta}{T}}) \rightarrow pbest_k(t) = X_k(t)$

If $F_k^{pbest} < F^{gbest} \rightarrow gbest(t) = pbest_k(t)$

Else

Calculate $\Delta = F_k^{pbest} - F^{gbest}$

Generate a random number r∈(0,1)

if $r \leq (p = e^{-\frac{\Delta}{T}}) \rightarrow gbest(t) = pbest_k(t)$

Update the velocity and position of each particle

Reduce the temperature $T_t = \alpha(T_{t-1})$

t=t+1

End While

Figure 3. The pseudo-code of the SA-PSO algorithm.

pose, a heuristic approach is proposed, here. Due to the FFS environment with unrelated parallel machines and renewable resources in this research, three decisions must be made to calculate the objective function:

1. Assign the jobs to each machine at each stage;
2. Determine the job sequence on each machine;
3. Allocate renewable resources to the machines.

The details of the proposed approach are discussed as below:

Step 1: Scheduling of the jobs in the first stage. By considering the obtained job sequences based on the RK method, the jobs are allocated to all the machines (available or unavailable) at the first stage, and completion times of the jobs are calculated. In this research, each job is virtually allocated to all the machines at the first stage, the machine with minimum completion time is selected and the job is really assigned to this machine.

Step 2: Scheduling of the jobs in other stages. For the second stage until the end, the jobs are sorted in an ascending order of the completion time at the previous stage. Afterwards, the jobs are allocated to all the machines (available or unavailable) and the machine with minimum completion time is selected to assign the job. By considering this procedure for entire stages, the completion time of the last job is calculated and considered as $C_{\max}$.

Step 3: Allocation of the renewable resources. As mentioned before, by allocating a fixed number of renewable resources to the machines, job processing time is reduced regarding the normal processing time and coefficient of processing time reduction. In this issue, resource allocation to the machines is conducted after the job assignment to the machines and job sequencing on the machines at each stage. First, scheduling of the jobs is performed by the normal processing time without considering any renewable resources. After that, the critical path on the Gantt chart, which determines $C_{\max}$, is identified and one renewable resource (if available) is assigned to the jobs on the critical path at each stage. Then, the new $C_{\max}$ is determined based on the new processing time. The above procedure is continued until entire renewable resources are assigned to the jobs.

A simple example: In order to illustrate how job scheduling is generated based on the proposed heuristic approach in the FFS problem with renewable resources, a simple example is considered, here. Suppose that

Table 2. The normal processing time of each job.

		Job 1	Job 2	Job 3	Job 4	Job 5
Stage 1	Machine 1	7	8	9	9	9
	Machine 2	6	4	7	3	7
Stage 2	Machine 1	10	7	2	2	2
	Machine 2	4	3	3	3	3
Stage 3	Machine 1	2	7	5	6	3
	Machine 2	8	4	9	4	4

there is a scheduling problem with five jobs; two, two, and one renewable resources in stages 1, 2, and 3, respectively, two machines at each stage, and $a^t = 1; t = 1,2,3$. The processing times of the jobs are produced randomly in the interval [2,10]. Table 2 shows the normal processing time of any job on different machines at any stage.

It is assumed that the proposed metaheuristic approach generates a vector for the jobs sequence as $J = (0.5, 0.2, 0.3, 0.8, 0.9)$ in a given iteration. The equivalent sequence vector generated by the RK method should be: $Seq = (5, 4, 1, 3, 2)$.

As mentioned above, in order to assign the jobs to each machine at any stage, a job in the sequence is assigned to all the available and unavailable machines at each stage and a machine with the earliest completion time is selected. As a result, the Gantt chart of the generated solution based on the proposed procedure (Steps 1 and 2 in the heuristic approach) is shown in Figure 4.

With regard to Figure 4, the initial makespan (without renewable resources assignment) equals 25.

In Step 3 of the heuristic approach, the renewable resources must be assigned to the jobs to reduce the processing times and as a result, reduce the makespan. As can be seen in Figure 3, the critical path on the $C_{\max}$ is jobs 4 and 3 on machine 1 at the first stage, job 3 on machine 1 at the second stage, and finally, job 3 on machine 1 at the last stage. Regarding the number of renewable resources at each stage and Constraint set (10), the processing time of the jobs on the critical path is reduced by one unit. As a result, the Gantt chart of the new sequence is changed as follows.

As a result, the new $C_{\max}$ with one renewable resource is 24.

In iteration 2, there is one renewable resource for the first and second stages; therefore, only jobs on the critical path are considered at theses stages. Therefore, regarding Figure 5, the critical path is jobs 5, 1, and 2 on machine 2 at the first stage, and job 2 on machine 2 at the second stage. By reducing one unit of the processing time of the jobs on the critical path at the first and second stages, the new Gantt chart is shown in Figure 6.

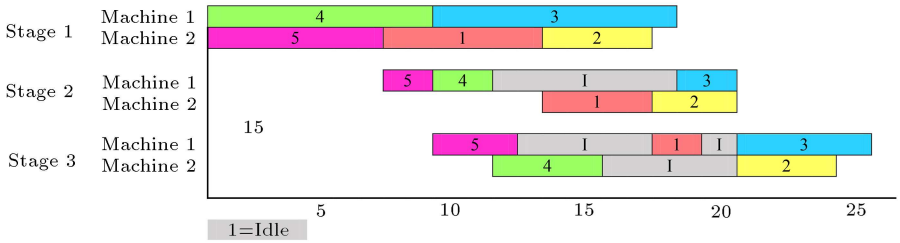

Figure 4. The Gantt chart with the normal processing time.

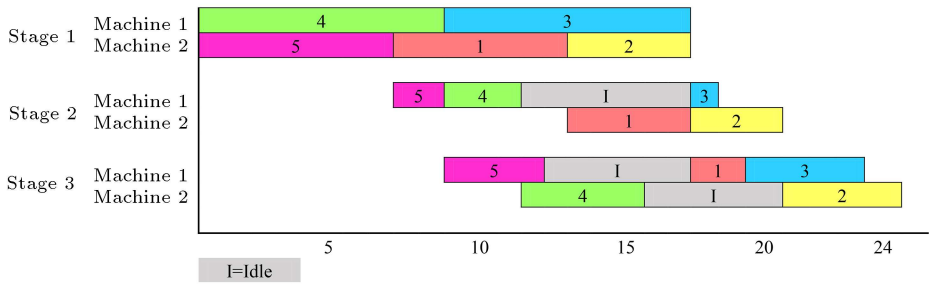

Figure 5. The Gantt chart in iteration 1.

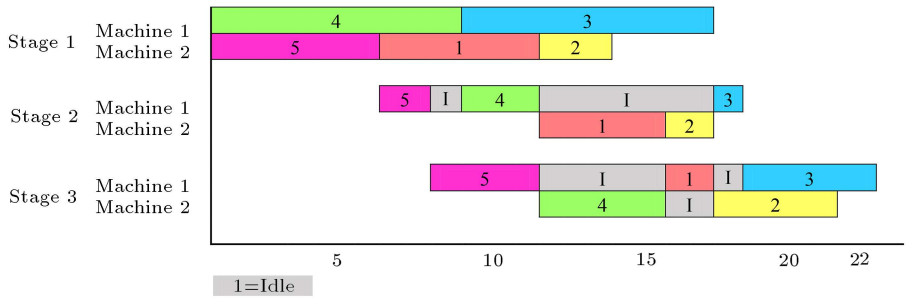

Figure 6. The Gantt chart in iteration 2.

Figure 6 shows that final $C_{\max}$ with two, two, and one renewable resource for the first, second, and last stages is equal to 22.

4.6. Improvement of the proposed algorithms

In order to improve the performance of the proposed algorithms, a local search scheme is incorporated into the metaheuristic algorithms. The local search finds entire neighborhoods of each particle by substituting every two jobs in the sequence vector. After that, the objective function of each particle is calculated. It is substituted by the best neighbor and the position vector is changed by the best neighbor. The proposed local search scheme is done on the *gbest* in each iteration.

5. Computational results

In this section, some numerical experiments are designed to investigate validation of the mathematical model. Furthermore, the performance of the proposed algorithms is investigated by comparing them with the optimal solutions and with each other. In this section, two different experiments are conducted for this purpose. At first, the performance of the proposed metaheuristic approaches is evaluated by comparison with optimal solutions through the small-size test problems. Afterward, the performances of the proposed metaheuristic approaches are compared with each other based on medium to large-size test problems. Optimal solutions of the test problems obtained by Lingo 9.0 software and the proposed metaheuristic algorithms, were implemented in MATLAB and tested on a computer with 2.4 GHz CPU and 3 GB of RAM.

5.1. Comparison of the proposed metaheuristic algorithms with the optimal solutions

This section is dedicated to evaluating the performance of the proposed metaheuristic algorithms by comparing them with optimal solutions, which are obtained by the proposed MILP model. Regarding the NP-hardness of the FFS scheduling problem with renewable resources, test problems are limited to a small size. For this purpose, 15 test problems were designed of small size to compare the mathematical model with the metaheuristic algorithms. In order to generate small-size test problems, five characteristics are used to typify the test problems. They are; the number of jobs, normal processing time, number of stages, number of machines at each stage, and number of renewable resources. Test problem characteristics are summarized in Table 3.

Based on the test problems characteristics, 15 test problems with different size are generated and each of them is solved by Lingo 9.0 software with a time limit of 3600 seconds. Their performance is evaluated in terms of CPU time and Optimal GAP, which is determined as follows:

$$Optimal\ Gap = \frac{C_{\max} - Optimal\ solution}{Optimal\ solution} \times 100, \quad (29)$$

in which, $C_{\max}$ and Optimal solution show the maximum completion time, generated by the metaheuristic algorithms and MILP model, respectively. It is necessary to mention that the local search scheme has a significant influence on the performance of the original PSO and improves its performance.

By considering Table 4, the PSO and SA-SPO algorithms have solved 9 and 11 out of 15 problems, optimally. Furthermore, the average optimal gap for both metaheuristic approaches is 3.0% and 1.2%, respectively. Moreover, the average CPU time to achieve the optimal solution in different test problems by the MILP model, PSO, and SA-PSO algorithms is equal to 379.1, 2.77, and 7.93, respectively. Thus, it can be concluded that both proposed metaheuristic algorithms are able to generate optimal/near-optimal solutions in a reasonable time. By considering the optimal gap column in Table 4, it is observed that the

Table 3. Test problems characteristics of the small size test problems.

Characteristic	Level
Number of jobs	$U[3,5]$
Normal processing time	$U[5,10]$
Number of stages	$U[2,3]$
Number of machines in each stage	$U[2,3]$
Number of renewable resources	$U[2,3]$

Table 4. Comparison of the metaheuristic algorithms with optimal solutions.

Test problem	Optimal solution	CPU time	PSO			SA-PSO		
			$C_{\max}$	CPU time	Optimal gap (%)	$C_{\max}$	CPU time	Optimal gap (%)
1	12.0	12	12.0	2.10	0.0	12.0	2.19	0.0
2	11.4	10	11.4	2.24	0.0	11.4	2.34	0.0
3	12.0	5	12.0	2.36	0.0	12.0	2.66	0.0
4	11.1	2	11.1	2.49	0.0	11.1	2.88	0.0
5	16.3	5	17.5	3.03	7.4	17.1	8.30	4.9
6	15.3	9	15.5	3.19	1.3	15.5	8.48	1.3
7	12.8	20	12.8	2.50	0.0	12.8	6.64	0.0
8	12.3	89	12.3	2.58	0.0	12.3	5.78	0.0
9	12.8	52	12.8	2.62	0.0	12.8	6.92	0.0
10	12.3	181	12.3	2.76	0.0	12.3	8.06	0.0
11	18.4	813	19.3	3.46	4.9	18.4	8.73	0.0
12	16.5	125	16.5	3.55	0.0	16.5	8.90	0.0
13	16.2	1024	17.9	2.86	10.5	16.4	12.97	1.2
14	15.6	2464	16.9	2.97	8.3	15.6	16.03	0.0
15	16.2	875	17.5	2.93	8.0	16.7	18.19	1.3

Table 5. Information related to the medium to large-size test problems.

Characteristic	Level
Number of jobs	$10 - 20 - 30 - 50 - 70 - 100$
Normal processing time	$U[5, 20]$
Number of stages	$3 - 5 - 7 - 10$
Number of machines in each stage	$3 - 5 - 7$
Number of renewable resources	$U[10, 50]$

solutions which are presented by the SA-PSO algorithm have better quality.

5.2. Comparison of the metaheuristic algorithms for medium to large-size test problems

This section is devoted to comparing the performance of the proposed metaheuristic algorithms based on some test problems. Regarding the NP-hardness of the FFS scheduling problem with renewable resources, the proposed MILP model cannot achieve optimal solutions in a reasonable time for medium to large-size test problems. Therefore, only metaheuristic algorithms are compared with each other. In order to generate the test problems, different levels of the test problem characteristics are summarized in Table 5.

For evaluation purposes, 72 test problems are randomly generated based on Table 5. Two criteria, CPU time (sec) and Relative Percentage Deviation (RPD), are used to compare the proposed algorithms. The RPD is determined as follows:

$$RPD = \frac{C_{\max} - best(C_{\max})}{best(C_{\max})} \times 100, \tag{30}$$

where $C_{\max}$ is the objective value that is obtained by a given algorithm and $best(C_{\max})$ is the best solution that is obtained from both algorithms. The obtained results are presented in Table 6.

The average CPU time and RPD in each group of test problems that are obtained by the PSO and SA-PSO algorithms in Table 6 are presented in Table 7.

As can be seen in Table 7, the hybrid SA-PSO provides better results than the PSO algorithm based on the average RPD. Furthermore, the average CPU time of both algorithms in each group of test problems is approximately similar.

For more scrutiny and as a formal comparison, the performance of the proposed algorithms is compared,

Table 6. Computational results of Particle Swarm Optimization (PSO) and SA-PSO algorithms on medium and large size test problems.

Test problem	Number of jobs	Number of stages	Number of machines in each stage	PSO			SA-PSO		
				$C_{\max}$	CPU time	RPD	$C_{\max}$	CPU time	RPD
1	10	3	3	64	112	0.00%	64	118	0.00%
2	10	3	5	73	115	0.00%	74	113	1.37%
3	10	3	7	78	124	4.00%	75	120	0.00%
4	10	3	10	85	178	0.00%	85	189	0.00%
5	10	5	3	102	182	0.00%	104	190	1.96%
6	10	5	5	103	185	0.00%	103	182	0.00%
7	10	5	7	100	188	1.01%	99	193	0.00%
8	10	5	10	86	190	0.00%	86	189	0.00%
9	10	7	3	123	165	0.82%	122	166	0.00%
10	10	7	5	122	159	0.00%	124	167	1.64%
11	10	7	7	110	163	0.92%	109	160	0.00%
12	10	7	10	103	166	0.00%	103	158	0.00%
13	20	3	3	121	420	0.83%	120	412	0.00%
14	20	3	5	110	427	0.00%	112	417	1.82%
15	20	3	7	102	417	0.00%	103	424	0.98%
16	20	3	10	99	433	1.02%	98	427	0.00%
17	20	5	3	138	419	0.73%	137	423	0.00%
18	20	5	5	122	469	2.52%	119	477	0.00%
19	20	5	7	116	478	1.75%	114	472	0.00%
20	20	5	10	104	487	0.97%	103	478	0.00%
21	20	7	3	137	466	0.74%	136	478	0.00%
22	20	7	5	134	480	1.52%	132	476	0.00%
23	20	7	7	116	550	0.87%	115	558	0.00%
24	20	7	10	114	556	1.79%	112	551	0.00%
25	30	3	3	159	567	1.92%	156	559	0.00%
26	30	3	5	144	569	0.70%	143	564	0.00%
27	30	3	7	134	573	2.29%	131	577	0.00%
28	30	3	10	166	570	0.00%	167	574	0.60%
29	30	5	3	163	571	1.24%	161	578	0.00%
30	30	5	5	148	578	0.00%	150	580	1.35%
31	30	5	7	136	582	3.03%	132	587	0.00%
32	30	5	10	133	575	3.10%	129	573	0.00%
33	30	7	3	185	677	1.65%	182	689	0.00%
34	30	7	5	166	679	1.84%	163	681	0.00%
35	30	7	7	148	670	0.00%	148	682	0.00%
36	30	7	10	145	682	2.84%	141	671	0.00%
37	50	3	3	204	666	0.00%	204	679	0.00%
38	50	3	5	197	833	1.03%	195	840	0.00%
39	50	3	7	185	827	3.35%	179	817	0.00%
40	50	3	10	166	816	1.84%	163	826	0.00%
41	50	5	3	225	820	6.13%	212	833	0.00%

Table 6. Computational results of Particle Swarm Optimization (PSO) and SA-PSO algorithms on medium and large size test problems (continued).

Test problem	Number of jobs	Number of stages	Number of machines in each stage	PSO			SA-PSO		
				$C_{\max}$	CPU time	RPD	$C_{\max}$	CPU time	RPD
42	50	5	5	217	828	1.88%	213	829	0.00%
43	50	5	7	210	1276	0.96%	208	1288	0.00%
44	50	5	10	201	1288	2.03%	197	1279	0.00%
45	50	7	3	240	1293	1.69%	236	1279	0.00%
46	50	7	5	231	1289	1.76%	227	1297	0.00%
47	50	7	7	222	1279	3.26%	215	1288	0.00%
48	50	7	10	215	1567	2.38%	210	1571	0.00%
49	70	3	3	299	1563	2.05%	293	1567	0.00%
50	70	3	5	290	1566	2.47%	283	1574	0.00%
51	70	3	7	281	1579	2.93%	273	1561	0.00%
52	70	3	10	260	1917	1.96%	255	1922	0.00%
53	70	5	3	305	1912	2.35%	298	1925	0.00%
54	70	5	5	294	1921	2.08%	288	1908	0.00%
55	70	5	7	289	1915	1.05%	286	1925	0.00%
56	70	5	10	275	1923	2.23%	269	1917	0.00%
57	70	7	3	315	1920	1.61%	310	1900	0.00%
58	70	7	5	309	1934	2.66%	301	1923	0.00%
59	70	7	7	311	3168	2.30%	304	3175	0.00%
60	70	7	10	300	3178	1.69%	295	3169	0.00%
61	100	3	3	367	3173	1.94%	360	3166	0.00%
62	100	3	5	360	3184	2.27%	352	3165	0.00%
63	100	3	7	341	3176	2.10%	334	3183	0.00%
64	100	3	10	336	3178	2.44%	328	3189	0.00%
65	100	5	3	392	3186	2.08%	384	3191	0.00%
66	100	5	5	374	3189	2.19%	366	3169	0.00%
67	100	5	7	353	3178	2.62%	344	3168	0.00%
68	100	5	10	340	3183	3.34%	329	3175	0.00%
69	100	7	3	423	3174	3.17%	410	3169	0.00%
70	100	7	5	386	3191	2.12%	378	3178	0.00%
71	100	7	7	381	3175	2.14%	373	3165	0.00%
72	100	7	10	339	3190	2.42%	331	3155	0.00%

statistically. Figure 7 shows the means plot and Least Significant Difference (LSD) intervals (at the 95% confidence level) for the algorithms.

The results demonstrate that the SA-PSO algorithm statistically outperforms the PSO algorithm with a 95% confidence level.

In order to evaluate the effects of different controllable parameters of the test problems, an average RPD plot for the controllable parameters is depicted, here. As a result, the number of jobs, the number of stages, and the number of machines at each stage are considered controllable parameters.

5.2.1. The number of jobs

The interaction between the type of algorithm and the number of jobs is depicted in Figure 8 based on the average RPD.

It can be seen that, in small size problems, both

Table 7. The average CPU time and Relative Percentage Deviation (RPD) in each group of test problems in medium and large size problems.

Problem size (job×stage)	PSO		SA-PSO	
	Average CPU time	Average RPD	Average CPU time	Average RPD
10×3	132	1.00%	135	0.34%
10×5	186	0.25%	189	0.49%
10×7	163	0.43%	163	0.41%
20×3	424	0.46%	420	0.70%
20×5	463	1.49%	463	0.00%
20×7	513	1.23%	516	0.00%
30×3	570	1.23%	569	0.15%
30×5	577	1.84%	580	0.34%
30×7	677	1.58%	681	0.00%
50×3	786	1.55%	791	0.00%
50×5	1053	2.75%	1057	0.00%
50×7	1357	2.27%	1359	0.00%
70×3	1656	2.35%	1656	0.00%
70×5	1918	1.93%	1919	0.00%
70×7	2550	2.07%	2542	0.00%
100×3	3178	2.19%	3176	0.00%
100×5	3184	2.56%	3176	0.00%
100×7	3183	2.46%	3167	0.00%
Total average	**1254**	**1.65%**	**1253**	**0.14%**

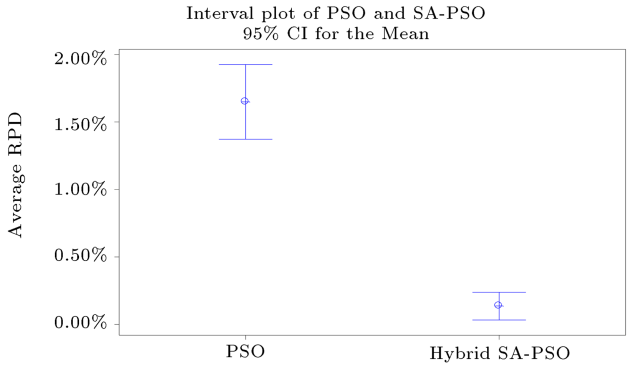

Figure 7. The means plot and Least Significant Difference (LSD) intervals.

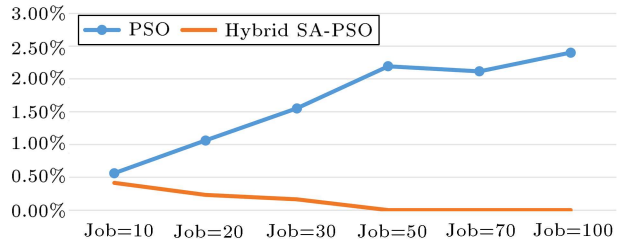

Figure 8. The interaction between the type of algorithms and the number of jobs.

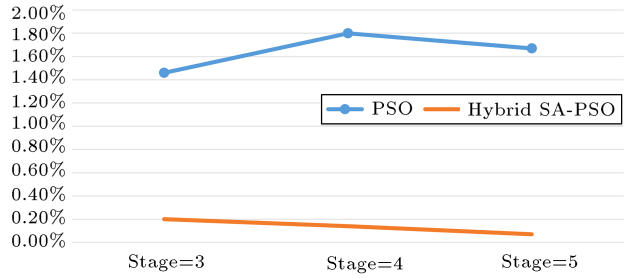

Figure 9. The interaction between the type of algorithms and the number of stages.

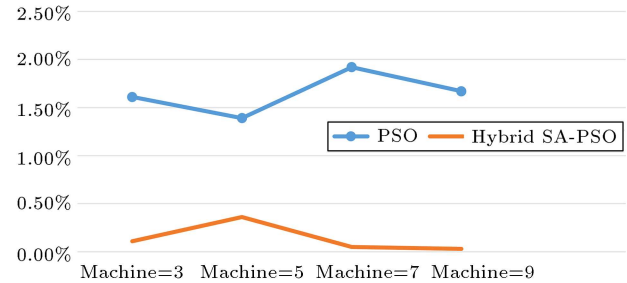

Figure 10. The interaction between the type of algorithms and the number of machines in each stage.

algorithms have similar performance, but for the larger size test problems, there is a significant difference between the proposed algorithms.

5.2.2. The number of stages

The average RPD plot to consider the effect of the number of stages on the quality of the proposed algorithms is depicted in Figure 9.

Figure 10 shows that the SA-PSO algorithm works better than the PSO algorithm in all the cases.

5.2.3. The number of machines at each stage

Another average RPD plot is used to see the effect of

the number of machines at each stage on the performance of the proposed algorithms (see Figure 10).

Regarding Figure 10, it can also be concluded that the SA-PSO algorithm shows the best performance in all cases.

6. Conclusions and future research

In this study, the effect of renewable resources on Flexible Flow Shop (FFS) scheduling problems with unrelated parallel machines is studied. Assignment of renewable resources to machines can lead to decreasing the makespan. For this purpose, a mixed integer linear programming model is proposed to minimize the makespan in an FFS environment with renewable resources. The proposed model was computationally NP-hard, therefore, a Particle Swarm Optimization (PSO) algorithm, as well as a hybrid SA-PSO algorithm, are proposed to solve the model. The obtained results from the randomly generated test problems show that the SA-PSO algorithm outperforms the PSO in both small and large size problems.

Suggestions for future studies include consideration of other objective functions such as cost-related and due date-related objective functions. Other ideas for future research are the consideration of release time and machine availability constraints, and batch processing. Also, other production environments, such as job shops and flexible job shops can be considered in future studies. Hybridization of other well-known metaheuristic approaches, such as Genetic Algorithms (GA), Variable Neighborhood Searches (VNS), and the Tabu Search (TS) can also be considered for future research.

Acknowledgment

The second author would like to thank Babol Noshirvani University of Technology for the financial support of this research under grant number BNUT/388026/98.

References

1. Su, L. and Lien, C. "Scheduling parallel machines with resource-dependent processing times", Int. *J. Prod. Res.* **117**, pp. 256–266 (2009).
2. Gholami, H.R., Mehdizadeh E., and Naderi, B. "Mathematical models and an elephant herding optimization for multiprocessor-task flexible flow shop scheduling problems in the manufacturing resource planning (MRP II) system", *Sci. Iran*, **27**(3), pp. 1562–1571 (2020).
3. Ebrahimi, M., Fatemi Ghomi, S.M.T., and Karimi, B. "Hybrid flow shop scheduling with sequence dependent family setup time and uncertain due dates", *Appl. Math. Model*, **38**, pp. 2490–2540 (2014).
4. Asadi-Gangraj, E. "Lagrangian relaxation approach to minimize makespan for hybrid flow shop scheduling problem with unrelated parallel machines", *Sci. Iran.*, **5**, pp. 3765–3775 (2018).
5. Zabihzadeh, S.S. and Rezaeian, J. "Two meta-heuristic algorithms for flexible flow shop scheduling problem with robotic transportation and release time", *Appl. Soft. Comput.*, **30**, pp. 319-330 (2016).
6. Karimi, N., Zandieh, M., and Karamooz, H.R. "Bi-objective group scheduling in hybrid flexible flowshop: a multi-phase approach", *Expert. Syst. Appl.*, **37**, pp. 4024–4032 (2010).
7. Shahvari, O. and Logendran, R. "A comparison of two stage-based hybrid algorithms for a batch scheduling problem in hybrid flow shop with learning effect", Int. *J. Prod. Econ.*, **195**, pp. 227–248 (2018).
8. Almeder, C. and Hartl, R.F. "A metaheuristic optimization approach for a real-world stochastic flexible flow shop problem with limited buffer", Int. *J. Prod. Econ.*, **145**, pp. 88–95 (2013).
9. Besbes, W., Teghem, J., and Loukil, T. "Scheduling hybrid flow shop problem with non-fixed availability constraints", *Eur. J. Ind. Eng.*, **4**, pp. 413–433 (2010).
10. Sangsawang, C., Sethanan, K., Fujimoto, T., and Gen, M. "Metaheuristics optimization approaches for two-stage reentrant flexible flow shop with blocking constraint", *Expert. Syst. Appl.*, **42**, pp. 2395–2410 (2015).
11. Jolai, F., Asefi, H., Rabiee, M., and Ramezani, P. "Bi-objective simulated annealing approaches for no-wait two-stage flexible flow shop scheduling problem", *Sci. Iran.*, **20**, pp. 861–872 (2013).
12. Akrami, B., Karimi, B., and Moattar Hosseini, S.M. "Two metaheuristic methods for the common cycle economic lot sizing and scheduling in flexible flow shops with limited intermediate buffers: The finite horizon case", *Appl. Math. Comp.*, **183**, pp. 634–645 (2006).
13. Marichelvam, M.K., Prabaharan, T., and Yang, X.S. "Improved cuckoo search algorithm for hybrid flow shop scheduling problems to minimize makespan", *Appl. Soft. Comput.*, **19**, pp. 93–101 (2014).
14. Choong, F., Phon-Amnuaisuk, S., and Alias, M.Y. "Metaheuristic methods in hybrid flow shop scheduling problem", *Expert. Syst. Appl.*, **38**, pp. 10787–10793 (2011).
15. Chung, T. and Liao, C. "An immunoglobulin-based artificial immune system for solving the hybrid flow shop problem", *Appl. Soft. Comput.*, **13**, pp. 3729–3736 (2013).
16. Dios, M., Fernandez-Viagas, V., and Framinan, J.M. "Efficient heuristics for the hybrid flow shop scheduling problem with missing operations", *Comput. Ind. Eng.*, **115**, pp. 88–99 (2018).
17. Behnamian, J. and Fatemi Ghomi, S.M.T. "Hybrid flowshop scheduling with machine and resource-dependent processing times", *Appl. Math. Model.*, **35**, pp. 1107–1123 (2011).

18. Jungwattanakit, J., Reodecha, M., Chaovalitwongse, P., and Werner, F.A. "Comparison of scheduling algorithms for flexible flow shop problems with unrelated parallel machines, setup times, and dual criteria", *Comput. Oper. Res.*, **36**, pp. 358–378 (2009).

19. Edis, E.B. and Oguz, C. "Parallel machine scheduling with flexible resources", *Comput. Ind. Eng.*, **63**, pp. 433–447 (2009).

20. Yin, N., Kang, L., Sun, T., Yue, C., and Wang, X. "Unrelated parallel machines scheduling with deteriorating jobs and resource dependent processing times", *Appl. Math. Model.*, **38**, pp. 4747–4755 (2014).

21. Figielska, E. "A new heuristic for scheduling the two-stage flowshop with additional resources", *Comput. Ind. Eng.*, **54**, pp. 750–763 (2008).

22. Figielska, E. "Heuristic algorithms for preemptive scheduling in a two-stage hybrid flowshop with additional renewable resources at each stage", *Comput. Ind. Eng.*, **59**, pp. 509–519 (2010).

23. Figielska, E. "Linear programming and metaheuristic approach for scheduling in the hybrid flowshop with resource constraints", *Control. Cybern.*, **4**, pp. 1209–1230 (2011).

24. Li, K., Shi, Y., Yang, S., and Cheng, B. "Parallel machine scheduling problem to minimize the makespan with resource dependent processing times", *Appl. Soft. Comput.*, **11**, pp. 5551–5557 (2011).

25. Liu, Y. and Feng, Z. "Two-machine no-wait flowshop scheduling with learning effect and convex resource-dependent processing times", *Comput. Ind. Eng.*, **75**, pp. 170–175 (2014).

26. Kellerer, H. "An approximation algorithm for identical parallel machine scheduling with resource dependent processing times", *Oper. Res. Lett.*, **36**, pp. 157–159 (2008).

27. Jun, P., Xinbao, L., Baoyu, L., Pardalos, P.M., and Min, K. "Single-machine scheduling with learning effect and resource-dependent processing times in the serial-batching production", *Appl. Math. Model.*, **58**, pp. 245–253 (2018).

28. Wang, X. and Wang, J. "Single-machine scheduling with convex resource dependent processing times and deteriorating jobs", *Appl. Math. Model.*, **37**, pp. 2388–2393 (2013).

29. Wei, C. and Ji, P. "Single-machine scheduling with time-and-resource-dependent processing times", *Appl. Math. Model.*, **36**, pp. 792–798 (2012).

30. Wang, D., Wang, M., and Wang, J. "Single-machine scheduling with learning effect and resource-dependent processing times", *Comput. Ind. Eng.*, **59**, pp. 458–462 (2010).

31. Wang, X. and Cheng, T.C.E. "Single machine scheduling with resource dependent release times and processing times", *Eur. J. Oper. Res.*, **162**, pp. 727–739 (2005).

32. Nguyen, S., Thiruvady, D., Ernst, A.T., and Alahakoon, D. "A hybrid differential evolution algorithm with column generation for resource constrained job scheduling", *Comput. Oper. Res.*, **109**, pp. 273–287 (2019). Doi: https://doi.org/10.1016/j.cor.2019.05.009

33. Gupta, J.N.D., Kruger, K., Lauff, V., Werner, F., and Sotskov, Y.N. "Heuristics for hybrid flow shops with controllable processing times and assignable due dates", *Comput. Oper. Res.*, **29**, pp. 1417–1439 (2002).

34. Kennedy, J. and Eberhart, R.C. "Particle swarm optimization", *Proceedings of the IEEE International Conference on Neural Networks*, **4**, pp. 1942–1948 (1995).

35. Poonthalir, G., Nadarajan, R., and Geetha, S. "Vehicle routing problem with limited refueling halts using particle swarm optimization with greedy mutation operator", *RAIRO-Oper. Res.*, **49**, pp. 689–716 (2015).

36. Nayeri, S., Asadi-Gangraj, E., and Emami, S. "Metaheuristic algorithms to allocate and schedule of the rescue units in the natural disaster with fatigue effect", *Neural Comput. Appl.*, **31**, pp. 7517–7537 (2019). Doi: https://doi.org/10.1007/s00521-018-3599-6

37. Bean, J.C. "Genetics and random keys for sequencing and optimization", *ORSA J. Comput.*, **6**, pp. 154–160 (1994).

38. Poli, R., Kennedy, J., and Blackwell, T. "Particle swarm optimization: an overview", *Swarm. Intell.*, **1**, pp. 33–57 (2007).

39. Tadayon, B. and Salmasi, N. "A two-criteria objective function flexible flowshop scheduling problem with machine eligibility constraint", Int. *J. Adv. Manuf. Tech.*, **64**, pp. 1001–1015 (2013).

40. Kirkpatric, S., Gelatt, C.D., and Vecci, M.P. "Optimization by simulated annealing", *Sci.*, **220**, pp. 671–680 (1983).

41. Asadi-Gangraj, E. and Nahavandi, N. "A metaheuristic approach for batch sizing and scheduling problem in flexible flow shop with unrelated parallel machines", *Int. J. Comp. App.*, **97**, pp. 31–36 (2014).

Biographies

Nariman Abbaszadeh received his BS degree in Industrial Engineering from Mazandaran University of Science and Technology, Iran, and his MS degree in Industrial Engineering from Babol Noshirvani University of Technology, Babol, Iran. His research interests include operations research, sequencing and scheduling, and production planning.

Ebrahim Asadi-Gangraj received his BS degree in Industrial Engineering from Isfahan University of Technology, Iran, in 2005, and MS and PhD degrees in Industrial Engineering from Tarbiat Modares University, in 2008 and 2014, respectively. He is currently Assistant Professor of Industrial Engineering at Babol

Noshirvani University of Technology, Babol, Iran. His research interests include applied operations research, sequencing and scheduling, production planning, and supply chain management.

Saeed Emami received his BS, MS and PhD degrees in Industrial Engineering from Isfahan University of Technology, Iran, and is currently Assistant Professor in the Department of Industrial Engineering at Babol Noshirani University of Technology, Babol, Iran. His research interests include operations research, robust optimization, production planning and scheduling.