Journal of Optimization in Industrial Engineering
Vol.12, Issue 2, Summer & Autumn 2019, 131-147
DOI: 10.22094/JOIE.2018.242.1532

Research Article

Three Hybrid Metaheuristic Algorithms for Stochastic Flexible Flow Shop Scheduling Problem with Preventive Maintenance and Budget Constraint

Sadigh Raissi [a,*], Ramtin Rooeinfar [b] ,Vahid Reza Ghezavati [b]

[a] *Islamic Azad University, South Tehran Branch, Tehran, Iran*
[b] *School of Industrial Engineering, South Tehran Branch, Islamic Azad University, Tehran, Iran*
Received 20 November 2017; Revised 02 September 2018; Accepted 08 September 2018

Abstract

Stochastic flexible flow shop scheduling problem (SFFSSP) is one the main focus of researchers due to the complexity arises from inherent uncertainties and also the difficulty of solving such NP-hard problems. Conventionally, in such problems each machine's job process time may encounter uncertainty due to their relevant random behaviour. In order to examine such problems more realistically, fixed interval preventive maintenance (PM) and budget constraint are considered.PM activity is a crucial task to reduce the production efficiency. In the current research we focused on a scheduling problem which a job is processed at the upstream stage and all the downstream machines get busy or alternatively PM cost is significant, consequently the job waits inside the buffers and increases the associated holding cost. This paper proposes a new more realistic mathematical model which considers both the PM and holding cost of jobs inside the buffers in the stochastic flexible flow shop scheduling problem. The holding cost is controlled in the model via the budget constraint. In order to solve the proposedmodel, three hybrid metaheuristic algorithms are introduced. They include a couple of well-known metaheuristic algorithms which have efficient quality solutions in the literature. The two algorithms of them constructed byincorporationof the particle swarm optimization algorithm (PSO) and parallel simulated annealing (PSA) methods under different random generation policies. The third one enriched based on genetic algorithm (GA) with PSA. To evaluate the performance of the proposed algorithms, different numerical examples are presented. Computational experiments revealed that the proposed algorithms embedboth desirable accuracy and CPU time. Among them, the PSO-PSAΠ outperforms than other algorithms in terms of makespan and CPU time especially for large size problems.

Keywords: Stochastic flexible flow shop; Budget constraint, Preventive maintenance; Genetic algorithm; Simulated annealing; Particle swarm optimization.

1. Introduction

The flexible flow shop scheduling problem (FFSSP) consists of a flow manufacturing line with one or more parallel machine on some processing stages (or workstation) in series. Multiple products (or jobs) are produced in each stage. The objective function of FFSSP is on minimizing the total completions of all jobs (makespan or C_{max})(Hoogeveen et al., 1996; Gupta, 1998). This kind of issue often takes into account an NP-hard problem. This means that at a reasonable computational time, an optimal answer cannot be obtained. (Brucker and Kramer, 1995)have proved that a two-stage FFSSP remains NP-hard even there is only one machine on the first stage and two machines on the second stage.Most researchers have focused on the system which their relevant process time deploy a given deterministic value. However, in real circumstances, processing time of jobs at each stage is the main part of makespan. Most research focused on the system with deterministicprocessing time. However, in a real system, theprocessing time is arandom variable due to random behaviour of tool wearing, operator skill, material variability and so on (Koulamas and Kyparisis, 2000). According to Choi and Wang (2012) makespan estimationmight become invalid under differentcircumstances. Another important factor that effects on makespan is interruption caused by machine breakdowns (Fahmy and Sharif, 2009). One of the most important ways to deal with these types of random interruptions is on following preventive and scheduled tasks. Hence, unavailability of machines, due to preventive maintenance, could be incorporated as a set of constraints in the mathematical models. Taking into account such disruptions due to preventive maintenance and stochastic processing time makes the problem more real, but more complicated and has been kept on the research gap at yet.The significant contributions of this paper are to propose a more realistic scheduling model for such production and delivering an efficient solution methods.The proposed model integrates FFSSP with preventive maintenance (PM) and budget constraint under stochastic processing time. Hence,consideringqueues between consecutive stages is a respectable alternative. When a job is processed at upstream stage and all the downstream machines get to busy state or comes under

*Corresponding author Email address: raissi@azad.ac.ir

any PM activity, therefore the job should wait in the bufferswhichcause holding cost.Because the total budget in handhas a given level this leads to interrupt the operations and cause increasing of the makespan. Consequently, in the proposed model the jobs are scheduled in such a way that there is no interruption in their operations. By considering all of these affecting factors, the optimized makespan will be acquired by the model. Obviously, such FFSSP also ruins an NP-hard model. Therefore, three hybrid populations based metaheuristic algorithms are introduced as solution methods. The flexible flow shop scheduling problem has been studied extensively in the literature. Koulamas and Kyparisis(2000) developed a heuristic method for a two and a three-stage FFSSP with random processing time based on the makespan objective function. They proved the effect of the proposed heuristic by finding the solutions which were better than the available lower bounds. A Tabu search (TS) algorithm together with a procedure for a constructing a complete schedule to solve the FFSSP with limited buffers has been given by Wardono and Fathi(2004) that minimizes job completion time. Their algorithm acts based on the stage-oriented decomposition approach. Akrami et al. (2006) presented a mixed-integer linear mathematical programming for FFSSP. They assumed that there are limited buffers among stages. Two meta-heuristics, based on genetic algorithm (GA) and TS algorithm, were presented to optimality solve the model. A flow shop scheduling was investigated with limited intermediate buffers was studied by Wang and Tang (2009). They focused on the makespan minimizing as a performance criterion and they applied a TS algorithm to optimize the problem. In order to improve the diversity of the TS, a scatter search mechanism was applied. The computational results showed the high efficiency of their proposed hybrid metaheuristic. Al-Hinai and ElMekkawy(2011) proposed a flexible flow shop problem with random machine failures and a two stage hybrid GA was applied. The first stage was on minimizing the primary objective; the makespan, and the second stage was optimized the bi-objective function and integrates machines assignments and operations sequencing with the expected machine breakdown. Tran and Ng (2011)presented a water-flow algorithm to solve the FFSSP with intermediate buffers. They pooled the amount of precipitation and falling force to form flexible erosion capability. This work helped the erosion process of the algorithm to focus on exploiting promising regions strongly. They also utilized an improved procedure for constructing a complete schedule from a permutation that represents the sequence of jobs at the first stage of the scheduling problem. Kianfar et al. (2012) investigated a FFSSP with non-deterministic arrival of jobs and sequence dependent setup times. They used average tardiness of jobs as the objective function. To optimize the problem, they presented a novel dispatching rule and hybrid GA. A computer simulation model was also developed to evaluate the presented dispatching rule. The results showed that their proposed dispatching rule can lead to much better results in comparison with the traditional dispatching rules. Singh and Mahapatra (2012) proposed a novel PSO to solve FFSSP. An efficient mutation operator was embedded in PSO to prevent solutions from falling into the local optimums. The performance of the PSO was evaluated against GA by a set of test problems taken from the literature. According to the obtained results, the percentage deviation of the proposed PSO of the lower bound is equal to 2.961 and the same measure for GA is equal to3.559.In order to deal with uncertain job processing times in FFSSP's, Choi and Wang (2012) presented a decomposition-based approach. The method combines two reactive approaches with a reactive-proactive approach. Lin and Ying (2013) proposed a hybrid algorithm based on the features of artificial immune systems and the annealing process of simulated annealing algorithms (SA) to optimize the FFSSP with limited buffer storage between stages. Almeder and Hartl(2013) studied the FFSSP with limited buffer storage in the metal–working industry. The partitioned the problem as a two stage problem. The first stage contains a single machine and its buffer. The semi-finished parts are stored in this buffer until a machine of the second stage is available. The second stage contains two parallel non-identical machines. They applied a hybrid approach based on discrete-event simulation and variable neighbourhood search to optimize the problem. The hybrid approach was led to an improvement between 3% and 10% compared with the current production plan of the company. By considering unrelated parallel machines, sequence-dependent setup times, probable reworks and different ready times, Rabiee et al. (2014) investigated the no-wait two-stage FFSSP. They proposed a novel hybrid algorithm based on imperialist competitive algorithm (ICA), SA, variable neighbourhood search (VNS) and GA. The performance of the proposed hybrid algorithm was evaluated against ICA, SA, VNS, GA and ant colony optimization (ACO) and high efficiency of their proposed hybrid algorithm was revealed. Arnaout (2014) tackled a rescheduling problem for the flow shop problem associated with stochastic processing and setup time. They proposed a new repair rule which and compared it with the existing algorithms. The results obtained from the computational experiments showed that the proposed repair rule can perform better those available algorithms in literature. Rahmani and Heydari (2014) studied FFSSP under uncertain processing times and unexpected arrivals of new jobs. They proposed a new approach to find robust schedules in this situation. Their approach was a proactive–reactive method which uses a two-step procedure. In order to consider stochastic processing time, Wang and Choi (2014) introduced a novel decomposition-based the Holonic approach (DBHA) to solve a FFSSP with stochastic processing time. Their proposed method was based on genetic algorithm control (GAC) and the shorting processing time contract net procontrol (SPT-CNP). Also, K-means clustering is utilized to divide machines into different clusters according to their stochastic environments. The gained results showed that DBHA had better quality solutions against GAC and SPT-CNPA simulation based optimization approach for stochastic hybrid FFSSP in a real-world semiconductor back-end assembly facility was presented by Lin and Chen (2015). In their approach, the optimization strategy, based on genetic algorithm, was used to find the optimal assignment of the production line and machine type at each stage. The simulation model was used to evaluate the performance of solutions. They applied a real case study to prove the necessity of using simulation optimization approaches for practical applications. A novel algorithm was developed by Li and Pan (2015) to solve the hybrid flow shop with limited buffers. They combine two metaheuristic algorithms of artificial bee colony and TS and used makespan as the performance criterion. Their proposed algorithm was evaluated against several algorithms reported in the literature and the experimental results showed the high effectively and efficient performance of the proposed algorithm. Sangsawang et al. (2015) proposed two metaheuristic algorithms to optimize a two-stage re-entrant FFSSP. The first algorithm was a hybridization of GA and adaptive auto-tuning based on a fuzzy logic controller. The second one was a hybridization of particle swarm optimization (PSO) and Cauchy distribution. Experimental results revealed that both hybrid algorithms could present better solutions and are more powerful than the classical metaheuristics. The statistical analyses indicated that the hybrid PSO and hybrid GA can improve the best solutions in the literature by averages of 15.60% and 15.51%, respectively. Zabihzadeh and Rezaeian(2015) developed a mixed integer linear programming model for FFSSP. They assumed that there are some robots between stages for unloading, transferring and loading parts. Their proposed model has ability of determining the number of robots and jobs sequence. They applied two meta-heuristic algorithms; GA and ACO, to solve their model. Computational results showed that the GA is more efficient than ACO to optimize the model. Tang et al. (2016) presented a new model for dynamic FFSSP's. By taking emergency maintenance, the proposed model minimized both objectives of energy consumption and makespan. Since they developed model was NP-hard, a multi-objective PSO algorithm to optimize the model. For finite capacity material requirement planning system in a flexible flow shop, Sukkerd and Wuttipornpun(2016) presented a hybrid GA and a Tabu search algorithm (HGATS). The results showed that the HGATS can outperform on comparing with the existing algorithms. Correspondingly, computational time of HGATS is acceptable when applied to real industrial systems. Rahmani and Ramezanian(2016) addressed a stochastic FFSSP in which new jobs arrive into the process as disruptions. They developed a mathematical model to minimize total weighted tardiness. A variable neighbourhood search algorithm was used to solve the model. The efficiency of the proposed algorithm evaluated through a set of test problems. González-Neira et al. (2016) presented a multi-criteria FFSSP in which criterion is quantitative and the other is qualitative. They assumed that job processing time deploys by a stochastic manner.The integral analysis method (IAM) was implemented to solve the relevant problem. In the IAM method, the problem first was introduced, then, cardinal analysis, ordinal analysis and integration analysis are done. Results showed that IAM method is able to select the alternatives with high efficiency in terms of both types of criteria.

The most important criticism of scheduling problems is the gap between academic and practical problems. Even though the developed models have tried to be more realistic, but they fail to find the exact makespan in real life problems. This comes from the fact that the model cannot consider all the factors affected on makespan. As a result, there is a gap between obtained makespan by mathematical models and the makespan occurs in reality. In order to bridge this gap, this research presents an integrated mathematical model which is capable to consider all of the influential factors on makespan. The proposed model can simultaneously consider preventive maintenance, stochastic processing time and budget constraint. By integrating all of these subjects the model will reflect the real performance of a FFSSP.

The rest of the paper is organized as follows: In section 2 the problem definition, mathematical model formulation and assumption of the model is presented. In section 3, the proposed hybrid metaheuristic algorithms in which three hybrid metaheuristic algorithms, including combining parallel SA with two types of PSO algorithms (PSO-PSAI and PSO-PSAΠ), and a GA with parallel SA (GA-PSA) are specially explained. In section 4, computational results are presentedwhich actually compare the results of proposed metaheuristic algorithms. First, the small scale problem size is solved with CPLEX software. Then, the metaheuristic algorithms are used to find the near optimal solutions for the large scale of the problems. Analysis of the results and comparisons demonstrate the performance of the proposed solving methodologies on different problem sizes. Finally, conclusions and future researches are presented in the last section.

2. Mathematical Model Formulation

In this section the problem under consideration is described. Consider a problem of J jobs and S working stages as shown in Figure 1. Each stage s has a number of N_s identical parallel machines that operate in parallel. All jobs visit all stages from the first to the last stage. They are processed by one machine at each stage. Each job processed all the stages and every machine process maximum one job at a time. Thereis a buffer between two consecutive stages. The machines are under PM tasks. . Consequently, each machine could be available or unavailable at each time. All jobs are independent of each other and they are available at time zero. The processing time of jobs is considered to be stochastic. In order to model such randomly distributed processing time, the stochastic programing technique is applied in which each possible processing time is called as a scenario. The probability of occurrences of each scenario is shown by $Pr(\pi)$. Therefore, the averaged processing time of each job at each stage is calculated according to its processing times in different scenarios and the probability of occurrences of each scenario. The objective is to set a sequence of jobs, allocating jobs to machines at each stage and determining the time between two consecutive maintenance activities in such a way that total completion time being minimized. Figure 2 shows an example of the Gantt chart of proposed stochastic flexible flow shop scheduling problem (SFFSSP) with 3 stages and 5 jobs. There is2, 3 and 2 machines in stages 1, 2 and 3, respectively. In machine 1 in stage 1, maintenance occurs after processing job 3. The other maintenance occurs in stages 2 and 3. The job 2 is processed in stage 1 from time 2 to 5 and is started in stage 2 from time 7. Therefore, job 2 is sent to the buffer of stage 1 and cause holding cost for each period of time, until machine 1 in stage 2 gets empty. Sometimes more than one job is placed in the buffers. For example, in the buffer of stage 2, from time 11 and 12, jobs 1 and 3 are placed in and increasing the holding cost for staying of each period of time in the buffer.

Other constraints as well assumptions are listed as follows:

- There is a set of jobs denoted J (j=1, 2,…, J) jobs which are available at time zero and no job may be cancelled before completion.
- The set of L consecutive stages denoted by (s=1, 2,…, L),
- Each stage is equipped with non-identical parallel machines denoted by (m=1, 2,…, N_s).
- The set of R, denoted by (r=1,2,…, R) indicated the number of intermediate buffers.
- All jobs have to process serially through all machines at each stage for only once time.
- The interruption processing of each job on all machines at each stage is not acceptable. On the other hand, once a job is started, it must be processed to completion without any interruption either on or between machines.
- All machines are continuously available at time zero, in another word; non-machines are failing at the starting time.
- Each machine could process only one job at the same time, and each job has to visit each machine exactly once.
- The preventive maintenance activities are performed on each machine at the fixed intervals (P_{ms}).
- Once the preventive maintenance activity is carried out, there is no probability of a subsequent machine breakdown.
- All jobsat each intermediate buffer havea same holding cost, but different at each stage.

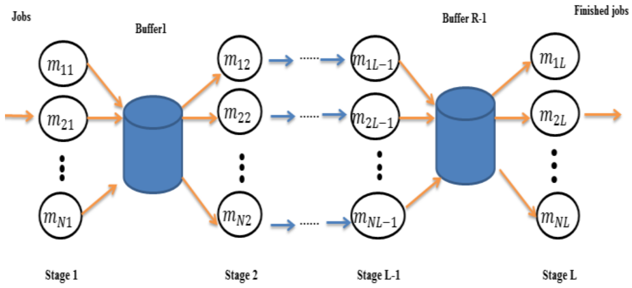

Fig.1.The framework of the flexible flow shop problem of this study

2.1. Notations

To present the model using mathematical terms, consider the following notations.

Indices

s Indexofstages$\{s$= 1,2,…, $L\}$
m Index of machines at s $\{m$= 1, 2, …,$N_s\}$
j Index of jobs$\{d, j$= 1, 2,…, $J\}$
r Index of intermediate buffers$\{r$= 1,2,…, $R\}$
h Index of job sequence$\{h, u$=1 ,2,…,$K_m\}$
n Index of maintenance activity which is done on machine j $\{n$= 1,2,…,$V_{ms}\}$
πIndex of probabilistic scenarios$\{\pi$ = 1, 2,…,$\Pi\}$

Parameters

$p_j^s(\pi)$The processing time of job j at stage s in scenario π
$Pr(\pi)$The probability of occurrences scenario π
h_{rs}^jHolding cost of job $j$inintermediate buffer of$r$ at stage s
μ_{ms}The repair rate of machine m at stage s
λ_{ms}The failure rate of machine m at stage s
D_{ms}The duration time of maintenance activity of machine$m$atstage$s$
$\bar{n}_{ms}$The mean number of jobs that are performed on machine m at stage s
βThe minimum of availability of the system
M An arbitrary large position number
B Total budget

Dependent decision variables

S_j^sStarting time for the processing of job j at stage s
C_j^sCompletion time of job j at stage s
P_{ms}The time between two consecutive maintenance activities on machine m at stage s
m_{ms}The number of maintenance activities on machine m at stage s
d_{rs}^jWaiting time of job j atintermediate buffer$r$ at stage s
$Z_j^r(t)\begin{cases}1 & \text{if job } j \text{ is in intermediate buffer } r \text{ at time } t\\ 0 & \text{otherwise}\end{cases}$
T_{ms}The completion time of the last PM action on machine m at stage s
$A_{ms}(t)$The availability of machine m at stage s at time$t$
$A_s(t)$The unavailability of stage s at time t
$A_{sys}(t)$The unavailability of system at time t
W 0 or 1

Independent decision variables

$X_{jh}^{ms}\begin{cases}1 & \text{if job } j \text{ is processed in sequence } h \text{ by machine } m \text{ at stage } s\\ 0 & \text{otherwise}\end{cases}$

$Y_n^{ms}\begin{cases}1 & \text{if } n \text{ th maintenance activity on machine } m \text{ at stage } s \text{ is executed}\\ 0 & \text{otherwise}\end{cases}$

2.2 Mathematical model

In our proposed model,availability of machine *m*in stage *s* at time *t* under PM activities could be calculated according to Villemeur(1991) by Eq. 1.

$$A_{ms}(t)=\frac{\mu_{ms}}{\mu_{ms}+\lambda_{ms}}+\frac{\lambda_{ms}}{\mu_{ms}+\lambda_{ms}}exp[-(\mu_{ms}+\lambda_{ms})(t-T_{ms})] \quad (1)\text{re}$$

we considered system configuration as a parallel-series system in which machines ateach stage are parallel and stages are a series. A stage is said to be unavailable if all of its machines are unavailable. Therefore, the unavailability of stage *s* is calculated by Eq. 2 and unavailability of the total system by Eq. 3.

$$A_s(t)=\prod_{m=1}^{N_s}\left(1-A_{ms}(t)\right) \quad (2)$$

$$A_{sys}(t)=1-\prod_{s=1}^{L}\left(1-A_s(t)\right) \quad (3)$$

Therefore, the proposed model as an extension to basic model was taken fromGonzález-Neira et al. (2016) will be as follows.

$$Min\ \{max\left(C_j^s\right)\} \quad (4)$$

St:

$$\sum_{m=1}^{N_s}\sum_{h=1}^{K_m}X_{jh}^{ms}=1 \qquad \forall j\in J; s\in L \quad (5)$$

$$\sum_{m=1}^{N_s}\sum_{j=1}^{J}X_{jh}^{ms}=1 \qquad \forall s\in L; h\in K_m \quad (6)$$

$$C_j^s\geq S_j^s+\sum_{\pi}^{\Pi}Pr(\pi)p_j^s(\pi) \quad (7)$$

$$C_j^{s-1}-M\left(1-Z_j^r(t)\right)\leq t\leq S_j^s+M\left(1-Z_j^r(t)\right)$$

$$\forall j\in J; s\in L; \forall r\in R|s=r \quad (8)$$

$$S_j^s+\left(1-X_{jh}^{ms}\right)M\geq C_d^s-(1-X_{du}^{ms})M$$
$$\forall j,d\in J\ |j\neq d;\ h,u\in K_m\ |u<h, m\in N_s; s\in L \quad (9)$$

$$\sum_{j=1}^{J}\sum_{r=1}^{R}d_{rs}^j h_{rs}^j\leq B \quad (10)$$

$$d_{rs}^j\geq S_j^{s+1}-C_j^s \quad (11)$$

$$(nP_{ms}Y_n^{ms}-C_j^s)X_{jh}^{ms}\geq -M\ (1-W)$$
$$\forall j\in J; \forall m\in N_s; \forall s\in L; h\in K_m; 0\leq n\leq V_{ms} \quad (12)$$

$$(C_j^s-p_j^s(\pi)-nP_{ms}Y_n^{ms}-D_{ms})\ X_{jh}^{ms}\geq -M(W)$$
$$0\leq n\leq V_{ms} \quad (13)$$

$$P_{ms}=\frac{\sum_{j=1}^{J}\sum_{h=1}^{K_m}X_{jh}^{ms}\sum_{\pi}^{\Pi}\Pr(\pi)\,p_j^s(\pi)}{m_{ms}}$$

$$\forall m\in N_s; \forall s\in L \quad (14)$$

$$T_{ms}\leq nP_{ms} \quad (15)$$

$$1-A_{sys}(t)\geq\beta \quad (16)$$

$$C_j^s, d_{rs}^j\geq 0 \qquad \forall j\in J; \forall s\in L \quad (17)$$

$$X_{jh}^{ms}\in\{0,1\}\forall j\in J; \forall m\in N_s; \forall s\in L; h\in K_m \quad (18)$$

$$Y_n^{ms}\in\{0,1\}\forall m\in N_s; \forall s\in L; n\in V_{ms} \quad (19)$$

Eq. (4) indicates the objective function. The constraint (5) ensures that the assignment of each job to one and only one machine at each stage. Constraints set (6) determines that the position of a machine sequence is takes by only one job at each stage. Constraints set (7) states that it is not allowed starting processing jobs at the next stage unless they have completed processing at the previous stage. Constraint sets (8) and (9) indicate that no interference should be taken among jobs on a common machine at any stage if the machine is available. On the other words, the difference between the processing times of any two jobs assigned to the same machine should not have any overlap.The constraint set (10) guarantees that the holding costs should be less than the available budget. Constraints set (11) determines the waiting time of jobs in each buffer.Constraints set (12) and (13) ensure that there are no overlap among operations and maintenance task. Constraint sets (14)-(16) are related to control the system availability.Finally, constraint set (17)-(19) controls the decision variable types.

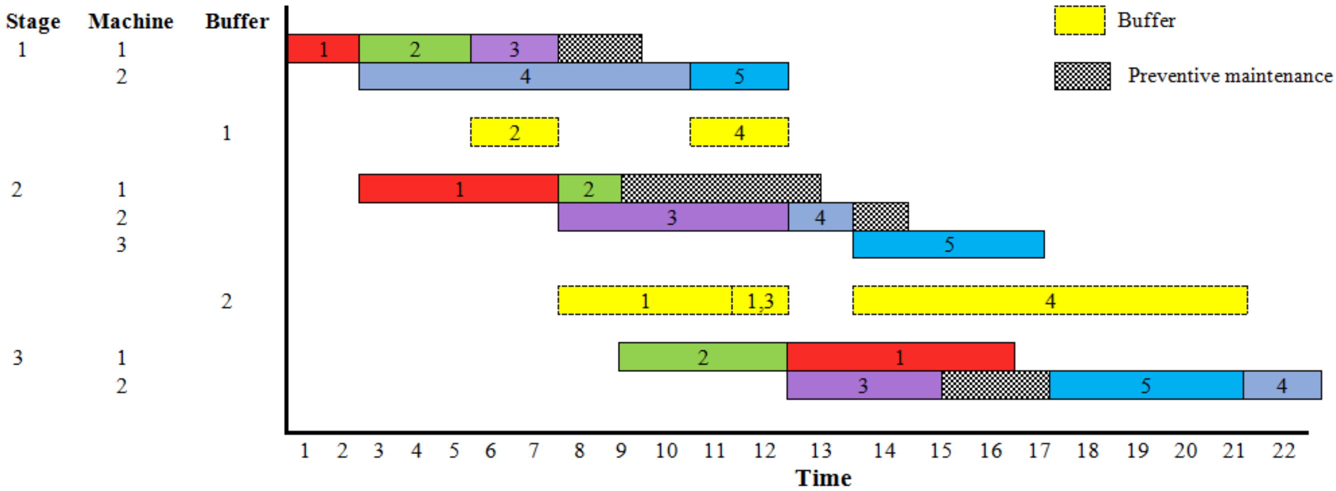

Fig. 2. An example of proposed SFFSSP Gantt chart of this study

3. Solution Methodologies

As mentioned earlier, the proposed modelof SFFSSP is an NP-hard problem and solving this problem with exact methods in reasonable CPU time is impossible, so we should use heuristic or metaheuristic algorithms to solve it(Pinedo& Chao, 1999). Pinedo (2002) proved that some exact methods have been developed for solvingSFFSSP's, but they are not suitable for more than 20 jobs and 10 machines. Also, heuristic algorithms are often may be trapped onsome local solutions, but metaheuristic algorithms can be described as a master strategy that guides and modifies subordinate heuristics to explore the solutions beyond the local optimally (Osam et al., 1996).We describe threehybridmetaheuristic algorithms, including combining parallel SA with two types of PSO algorithms (named PSO-PSAI and PSO-PSAП), and a GA with parallel SA (named GA-PSA) which have good solution quality in the literature(Singh and Mahapatra, 2012; Kianfar et al., 2012; Rabiee et al., 2014; Tang et al., 2016; Sukkerd and Wuttipornpun, 2016). Theproposed metaheuristic algorithms are explained in the nextsubsections on details.

3.1. Hybrid PSO-PSA algorithms

PSO algorithm is an evolutional solution method performed on a population of candidate solutions called particles. These particles move around in the search-space according to simple routine. The particles movements are guided by the best found positions in the search-space, which are continually updated as better positions are found by the particles. At each iteration, the in position of a particle (X vector) is updated by calculating the velocity (Vel vector) using the differences between the current position of the particle and the two following vectors (Kennedy and Eberhart, 1995):

- The best position experienced by the particle in all previous iterations. This is called the particle best (Pbest).
- The best position experienced by all particles in population in all previous iterations. This is called the global best (*gbest*).

Generally, a PSO is a continuous algorithm inherently, while SA is a discrete one. Experiments show that combining PSO with a discrete algorithm such as SA creates better performances. Poli, Kennedy and Blackwell (2007) presented a review on the variation and the hybridization of the PSO. We proposed two types of hybrid PSO-PSA algorithms named PSO-PSAI and PSO-PSAП to have both advantages of these methods. The basic idea of the hybrid PSO-PSA algorithm is running the PSO algorithm and improving the best results by applying parallel SA (PSA). In order to have variety in the proposed algorithm, we consider every bad, normal and good solutions could be selected with same chances.Combining PSO and PSA decreases the probability to be trapped in the local optimal solutions. Also, by introducing a suitable neighbourhood formation structure, the search process is enhanced and finds the near global optimum solution. The essential components of our proposed PSO-PSAI and PSO-PSAП are similar, except in generating initial solutions as described below. The pseudo-code for our PSO-PSA algorithm is shown Figure 6.

3.1.1. Initial solution of PSO-PSAI

The random generation policy is used to generate the initial population of PSO-PSAI.

3.1.2. Initial solution of PSO-PSAП

In PSO-PSAП a new procedure is applied to generate initial solutions. First, we consider SFFSSP with relaxed conditions in which the binary constraints of model are relaxed. In other word the $X_{jh}^{ms}$is considered to be a real number between 0 and 1. Next, the relaxed model was solved by CPLEX. The optimum values of decision variables are numbers between 0 and 1 which are looked as a probability distribution. These values are used as primal inputs of initial solutions for PSO-PSAΠ. For example, if is $X_{23}^{12} = 0.8$ then we set $X_{23}^{12} = 1$ with the 0.8 and $X_{23}^{12} = 0$ with the 0.2. The initial solutions obtained by this methodare distributed in the high quality of solution space. In order to perform a comprehensive search the low quality of solution space must be investigated. Therefore, some initial solutions are generated, unlike the obtained probability distributions. For example, if $X_{23}^{12} = 0.8$, then we set $X_{23}^{12} = 1$ with the 0.2 and $X_{23}^{12} = 0$ with the 0.8, the two groups if initial solutions are merged and formed initial solutions of PSO-PSAΠ.

The input parameters of both proposed PSO-PSAI and PSO-PSAΠ are: the population size (N_{pop}), the number of successive iterations in which the best solution does not change (M), the cognition learning factor (C_1), The social factor (C_2) the inertial weight (w), the population size (N_{pop}), the number of internal loop (*In loop*), and the temperature decreasing rate (α). The other components of proposed PSO-PSAI and PSO-PSAΠ are the same and described as follows:

3.1.3. The solution structure of PSO-PSA

The solution is represented by two sub-matrixes, each of them are associatedto a special area of decision variable. The first sub-matrix presents the sequence of jobs contains $S*J$ matrix where S is the number of stations and J is the number of jobs. An enhanced version of random key representation, proposed by Norman and Bean (1999) is used to show the sequence of jobs which is capable to preserve the solution feasibility. In this way, each job at each station is assigned a real number between$(1, N_s)$. The integer part is the number determines the machine number to which the job is assigned and the fractional part is used to sort the jobs assigned to that machine. For example, consider a problem with 4 jobs, 5 stations and 4 machines at each station. An example of a solution for this problem is presented in Figure 3. According to Figure 4, in station 2, the jobs 1 and 4 are both processed on machine 3, and the job 2 is processed on machine 1, and the job 3 is processed on machine 2. Also the order of jobs to be scheduled on machine 3 is job 1 followed by job 4.The second sub-matrix is related to the number of maintenance activity on machines. It is a $S*M$ matrix where S is the number of stations and M is the number of machines at each station. Each cell of this matrix is filled with a random number between 1 and the maximum number of allowable maintenance activity. Figure 4 is an example of a problem with 5 stations and 4 machines at each station. As shown in Figure 4, three, one, three and four maintenance activities are performed on machines 1, 2, 3 and 4 in station 1, respectively. Therefore, the decision variables Y_1^{11} , Y_2^{11} and Y_3^{11}will be equal to one.

	Job 1	Job 2	Job 3	Job 4
Station 1	4.41	2.29	3.12	1.65
Station 2	3.37	1.84	2.62	3.73
Station 3	1.97	3.42	3.85	4.24
Station 4	2.54	1.16	3.96	2.52
Station 5	1.04	3.73	2.14	4.91

Fig.3. An example of representation of solution (Sequence of jobs)

	Machine 1	Machine 2	Machine 3	Machine 4
Station 1	3	1	3	4
Station 2	2	2	4	3
Station 3	1	4	2	3
Station 4	3	3	4	2
Station 5	2	1	3	3

Fig. 4. An example of representation of solution (Maintenance activity)

3.1.4. Particle movements

For particle movements, the following formulas are used to update the velocity and position vectors of a particle:

$$Vel_i(k+1) = w * Vel_i(k) + C1 * r1 * \left(Pbest_i - X_i(k)\right) + C2 * r2 * (gbest - X_i(k)) \quad (19)$$

$$X_i(k+1) = X_i(k) + Vel_i(k+1) \quad (20)$$

In equation (19), $Vel_i(k)$ is the velocity of particle i in the k^{th} iteration and $X_i(k)$ states the position of particle i in iteration k. Also, $Pbest_i$ is the vector for the best known position of particle i and *gbest* is the best position vector of all particles in the whole population. w is called the inertia weight that determines the impact of the current velocity of a particle on its velocity at the next iteration. The parameters C_1 and C_2 are acceleration coefficients which have constant values to determine the impact of *Pbest* and *gbest*values in defining the velocity, respectively. r_1 and r_2 are two random numbers uniformly distributed in [0,1].

3.1.5. Local search improvements

One of the challenges of the algorithms is trapped in local optimal solutions. In the other words, it is possible that the near optimum solution which has found yet, is selected and the algorithm accepts this solution as a final optimum solution and stops. The hybrid algorithms are used to combine the base algorithm with different strategies to improve the algorithm performances. We utilized medium radius local search for the proposed hybrid PSO-PSA algorithms. In this method, the local searches are not used on each swarm, and we used the double change technique. If the improvements on the convergence of fitness function are achieved, we replace it to the previous swarm, but, if no improvements have been seen, we accept this by Bultzen probability.

3.1.6. Initial temperature

A suitable initial temperature is the one that results in an average increase of acceptance probability near to one. The value of initial temperature will clearly depend on the scaling of fitness and, hence, it should be problem-specific. Therefore, we first generate a large set of random solutions, then a standard division of them is calculated and is used to determine the initial temperature in the way that the acceptance probability of primary generations reach to 0.999. Consequently, the initial T_k is set to 1000based on some preliminary examinations.

3.1.7. Cooling Schedule

The performance of this algorithm also depends on the cooling schedule, which is essentially the temperature updating function. In the proportional decrement scheme, temperatures at the k and k+1 steps of the outer loop, $T_k$and T_{k+1}, are related by:

$$T_{k+1} = \alpha T_k \qquad (21)$$

Where α is cooling rate and is obtained by some experiment.

3.1.8. Stopping criteria

To limit the number of iterations of PSO-PSA algorithms, some convergence experiment was performed and the best criterion was applied as follows:
PSO-PSA will be stopped when the best solution does not change after a pre-determined number of successive iterations (*M*). Also, the PSA is allowed to search in a temperature level, for In-loop iterations. The optimum value of *M* and In-loop is determined by Taguchi experiments. The pseudo-code of the proposed PSO-PSA is described in Figure 5.

Procedure of hybrid PSO-PSA algorithms

P: initial particle with random positions and velocities
Until*termination condition is met* **Do**
    For each particle *i***do**
    Update the velocity of particle *i*
    Update the position of particle *i*
    Evaluate particle *i*
    Update *Pbest* and *gbest*
    Endfor
Start PSA with some of the best and the worst chromosomes

Set repetition counter $k = 0$
Until*termination condition is met* **Do**
Set repetition counter $M = 0$
Until*M= in_loop***Do**
Generate a neighbour solution: ω_2
Calculate $\Delta_{\omega 1,\, \omega 2}=f(\omega) - f(\omega_0)$
If $\Delta_{\omega 1,\, \omega 2} \leq 0$, then $\omega_1 = \omega_2$
If $\Delta_{\omega 1,\, \omega 2} > 0$, then $\omega_1 = \omega_2$ with probability exp $(-\Delta_{\omega 1,\, \omega 2} / t_k)$
$m = m + 1$
End
$T_{k+1} = \alpha\, T_k$
$k = k + 1$
End
Transfer the improved solution to the particles
End

Fig. 5. Pseudo-code of hybrid PSO-PSA

3.2. Hybrid GA-PSA algorithm (GA-PSA)

Genetic algorithm (GA) has no ability to search effectively to find the best global optimum solution. Also, this algorithm isn't a capable to complete local searches on solutions. Therefore, we can combine the power of GA in global search with simulated annealing (SA) local searches to address the global optimum solution. This hybrid algorithm which combines GA and SA has both advantages of these algorithms and help to improve solution performances.In this hybrid algorithm, first the GA generates initial solutions with crossover and mutation operators. Next, some of these solutions have been selected for the parallel SA (PSA) as initial solutions. Then, the parallel local search process on the selected solution starts. To have variety in the proposed algorithm, we consider every bad, normal and good solutions could be selected with same chances. The PSA procedure in the same applied in PSO-PSA. The other components of PSO-SA are as follows:

3.2.1. Initialization

The input of GA-PSA is the population size (N_{pop}), the number of successive iterations in which the best solution does not change (*M*), the crossover probability (*Pc*), and the similarity coefficient (*SC*), and the mutation probability (*Pm*) the number of internal loops (*In loop*), and the temperature decreasing rate (α) are first initialized. Then, to generate an initial population, a random generation policy is utilized in this step. Since the solutions obtained by a metaheuristic algorithm are sensitive to their parameter values, a statistical procedure based on the Taguchi parameter tuning method is used to tune the parameters.

3.2.2. Selection operator

One of the most key elements of a GA is the selection operator which is used to select chromosomes (parents) which lead to generate new chromosomes (offspring). The proposed selection operator is roulette wheel selection method in which parent chromosomes are probabilistically selected based on their fitness function value. The better chromosomes are selected with the highest probability. Using the roulette wheel selection each chromosome in the population occupies a slot with a slot size proportional to the chromosome fitness. When the wheel is randomly spun, the chromosome corresponding to the slot where the wheel stopped is selected as the first parent. This process is repeated to find the second parent. Clearly, since better chromosomes have larger slots, they have better chances to be chosen in the selection process.

3.2.3. The chromosome structure

The chromosome structure of the GA-PSA is just like that used in "solution structure" in PSO-PSA.

3.2.4. Chromosome evaluation

In order to evaluate chromosomes, each chromosome is simulated for 30 times. Then, the obtained results are averaged and considered as the chromosome fitness function. In the other word, by changing the stochastic input parameters, the fitness function of the chromosome will change. Therefore, the fitness function of each chromosome is not under deterministic value and show the stochastic nature of the model.

3.2.5. The crossover operator

As defined in the previous section, the chromosomes have two parts, so a crossover operator is applied in these two parts. For each part of the chromosome a single-point crossover is applied. For the first part of the chromosome, a cross point between (1, J) is generated (J is the number of jobs). Then, the crossover operator is applied according to Figure 8. The same procedure is applied to generate the second of the offspring chromosomes.

3.2.6. The mutation operator

The mutation operator is used in only some iterations. The similarity coefficient (SC) determines if mutation is applied or not. We can calculate the SC as follows:

$$SC_{ab} = \frac{\sum_{j=1}^{J}\sum_{h=1}^{K_m}\sum_{m=1}^{N_s}\sum_{s=1}^{L}\partial(X_{jh}^{ms}(a), X_{jh}^{ms}(b))}{S \times J} \tag{22}$$

Where X_{jh}^{ms}(a) and $X_{jh}^{ms}(b)$ are decision variables in chromosomes a and b. For comparing the similarity between two chromosomes, we consider the similarity of each gene that can be expressed as follows:

$$\partial(X_{jh}^{ms}(a), X_{jh}^{ms}(b)) = \begin{cases} 1 & if\ X_{jh}^{ms}(a) = X_{jh}^{ms}(b) \\ 0 & otherwise \end{cases} \tag{23}$$

The average similarity coefficient of the population is calculated as follows:

$$\overline{SC} = \frac{\sum_{a=1}^{N-1}\sum_{b=a+1}^{N} SC_{ab}}{\binom{N}{2}} \tag{24}$$

Where$N$ is the number of chromosomes in the population. Considering a pre-defined threshold similarity coefficient, the specified mutation operator will be automatically applied. Two swapping types are used in proposed GA-PSA. Swapping type 1 is used to define the neighbourhood $N(s)$ in local search (PSA) and swapping type 2 is used as mutation operator of GA.

- Swapping type 1

The mutation operator is applied on both two parts of chromosomes. To this aim, a column of each chromosome sub matrix is randomly selected and is inversely arranged. Figure 6 shows an example of the mutation operator.

Parent			
4.41	2.29	3.12	1.65
3.37	1.84	2.62	3.73
1.97	3.42	3.85	4.24
2.54	1.16	3.96	2.52
1.04	3.73	2.14	4.91

Offspring			
1.04	2.29	3.12	1.65
2.54	1.84	2.62	3.73
1.97	3.42	3.85	4.24
3.37	1.16	3.96	2.52
4.41	3.73	2.14	4.91

Fig.6. An example of mutation operator of swapping type 1 (Sequence of jobs)

- Swapping type 2

The swapping type 2 works which select two rowsof each chromosome sub matrix is randomly selected and swapped (see Figure7).

Parent			
4.41	2.29	3.12	1.65
3.37	1.84	2.62	3.73
1.97	3.42	3.85	4.24
2.54	1.16	3.96	2.52
1.04	3.73	2.14	4.91

Offspring			
1.65	3.12	2.29	4.41
2.54	1.84	2.62	3.73
1.97	3.42	3.85	4.24
3.37	1.16	3.96	2.52
4.41	3.73	2.14	4.91

Fig. 7. An example of mutation operator of swapping type 2 (Sequence of jobs)

Cross point

Parent 1			
4.41	2.29	3.12	1.65
3.37	1.84	2.62	3.73
1.97	3.42	3.85	4.24
2.54	1.16	3.96	2.52
1.04	3.73	2.14	4.91

Offspring 1			
4.41	2.29	1.11	2.25
3.37	1.84	3.07	4.66
1.97	3.42	4.52	3.38
2.54	1.16	2.86	2.93
1.04	3.73	1.55	3.72

Parent 2			
3.93	4.42	1.11	2.25
3.72	2.85	3.07	4.66
2.96	1.84	4.52	3.38
2.77	3.77	2.86	2.93
4.13	2.45	1.55	3.72

Offspring 2			
3.93	4.42	3.12	1.65
3.72	2.85	2.62	3.73
2.96	1.84	3.85	4.24
2.77	3.77	3.96	2.52
4.13	2.45	2.14	4.91

Fig. 8. An example of crossover operator (Sequence of jobs)

3.2.7. Stopping criteria

The stopping criteria in the GA-PSA is similar to the ones described for PSO-PSA. Figure 9 shows the pseudo-code of the proposed GA-PSA algorithm.

Procedure of hybrid GA-PSA algorithm

P: generate the initial population (N_{pop})
Untiltermination condition is met**Do**
Evaluate the chromosomes
Apply the crossover operator
Apply the mutation operator
Start PSA with some of the best and the worst chromosomes
For all the selected chromosomes **Do**
Set repetition counter $k = 0$
Until*termination condition is met* **Do**
Set repetition counter $M= 0$
Until$M= in\_loop$**Do**
Generate a neighbour solution: ω_2
Calculate $\Delta_{\omega 1,\omega 2}=f(\omega) - f(\omega_0)$
If $\Delta_{\omega 1,\omega 2} \le 0$, then $\omega_1 = \omega_2$
If $\Delta_{\omega 1,\omega 2} > 0$, then $\omega_1 = \omega_2$ with probability exp $(-\Delta_{\omega 1,\omega 2}/ t_k)$
$m = m + 1$
End
$T_{k+1}= \alpha\, T_k$
$k = k + 1$
End
End
Transfer the improved solutions to the genetic population and combine all the generated populations
End

Fig. 9. Pseudo-code of GA-PSA

4. Computational Evaluation

All experimental results have been carried out on an ASUS laptop with a 2.4 GHz, core i5 processor using a 4GB of RAM. All metaheuristic algorithms have been implemented in MATLAB Software (Version 7.10.0.499, R2010a) and the linear programming models have been solved using CPLEX 12. Also, the Minitab 17 software has been used to apply theTaghuchi design of experiment method for parameters tuning.Two sets of test problems are applied to solve the model. These test problems are generated based on the data given in Table 1. Here, we considered 5 scenarios, each of which has equal probability to be occurring.Three machines workat each station, considering the number of jobs and the numbers of stations, the two test sets aregeneratedin small and large problem sizes. First, 18small sized test problems are generated and the solutions, obtained by the algorithms, are evaluated against global optimum amounts.Furthermore, 60 large sizedtest problems are applied to compare the performance of the algorithms. We considered two kinds of total budget called hereinafter by type 1 and 2. Both budget types are associated to the number of jobs. The budget types 1 and 2 are respectively calculating by multiplying of number of jobs in 1200 and 1000. Therefore, the budget type 2 is more strict than type 1.

Table 1
Factor and their levels

Factors	Levels
Number of jobs (*j*)	[4, 5, 6, 20, 50, 60, 100]
Number of stations (*s*)	[2, 3, 4, 6, 9]
Holding cost ($/sec)	Uniform [50, 100]
Process times (sec)	Uniform [1, 99]
The minimum of availability	0.95
The repair rate of machine (sec)	Uniform [10, 50]
The failure rate of machine (sec)	Uniform [150, 450]
Number of scenarios	5

4.1. Parameter tuning

As mentioned before, the initial parameters of hybrid PSO-PSA algorithms (including both PSO-PSAI and PSO-PSAΠ) are the population size (N_{pop}), the number of successive iterations in which the best solution does not change (M), the cognition learning factor (C_1), The social factor (C_2) the inertial weight (w), the population size (N_{pop}), the number of internal loop (*In loop*), and the temperature decreasing rate (α). Also, the population size (N_{pop}), the number of successive iterations in which the best solution does not change (M), the crossover probability (Pc), and the similarity coefficient (SC), and the mutation probability (Pm) the number of internal loops (*In loop*), and the temperature decreasing rate (α) were used in GA-PSA.To investigate the influence of those parameters on the performance of the algorithms, we implement the Taguchi's method in the design of experiments (DOE) (Montgomery, 2005).In the Taguchi's method, the results are converted into an estimator called single to noise ratio (S/N). The S/N ratio shows both the mean and the variation in the response variables. To minimize the objective function the S/N ratio is calculated as the following formula:

$$^{S}/_{N} = -10\log\left(^{1}/_{n}\sum_{i=1}^{n}\frac{1}{y_i^2}\right) \tag{23}$$

Which, n and $y_i$indicate number of replications and process response value at i'th replication. We chose the orthogonal array L_{27} for both PSO-PSA and GA-PSA. In Taguchi's design of experiments, the relative percentage deviations (RPD's) transform into S/N ratio, we figure out the response value of each parameter in the proposed algorithms.After testing the different parameter levels and analysing the obtained results, the better initial parameter levels were gained. The initial and optimumparameter levels of each proposed hybrid algorithm are shown in Tables 2.According to the parameter values in Table 2, we illustrate the trend of each factor level of PSO-PSA and GA-PSA are in Figures 10 and 11, respectively. Also, the results of each test problem solved by the proposed algorithms are shown in Figure 12. The statistical results of the objective function (makespan)and CPU time for the small and large sized problems are presented in Tables 5 and 6.Additionally, we utilized the relative percentage deviation (RPD) to test the performances of applied metaheuristics as below:

$$RPD = \frac{Alg_{sol} - L_{sol}}{L_{sol}} \qquad i = 1, \dots, n \tag{32}$$

Where, $Alg_{sol}$is the objective function value for a given algorithm, $L_{sol}$is the best value of the objective function between algorithms and n is the number of small size or large size problems.

Table 2
Initial and optimum parameter levels of hybrid algorithms

Algorithms	Parameters	Factor levels			Optimum amount
		1	2	3	
PSO-PSA	$N_{pop\ PSO\text{-}PSA}$	100	200	300	200
	M	10	20	30	20
	$C1$	1	1.5	2	1.5
	$C2$	0.7	1	1.5	1
	Ω	0.3	0.4	1	0.4
	In loop	5	10	15	10
	α	0.87	0.91	0.95	0.91
GA-PSA	$N_{pop\ GA\text{-}PSA}$	100	200	300	200
	M	10	20	30	20
	Pc	0.85	0.9	0.95	0.9
	SC	0.6	0.7	0.8	0.7
	Pm	0.005	0.01	0.015	0.01
	In loop	5	10	15	10
	α	0.87	0.91	0.95	0.91

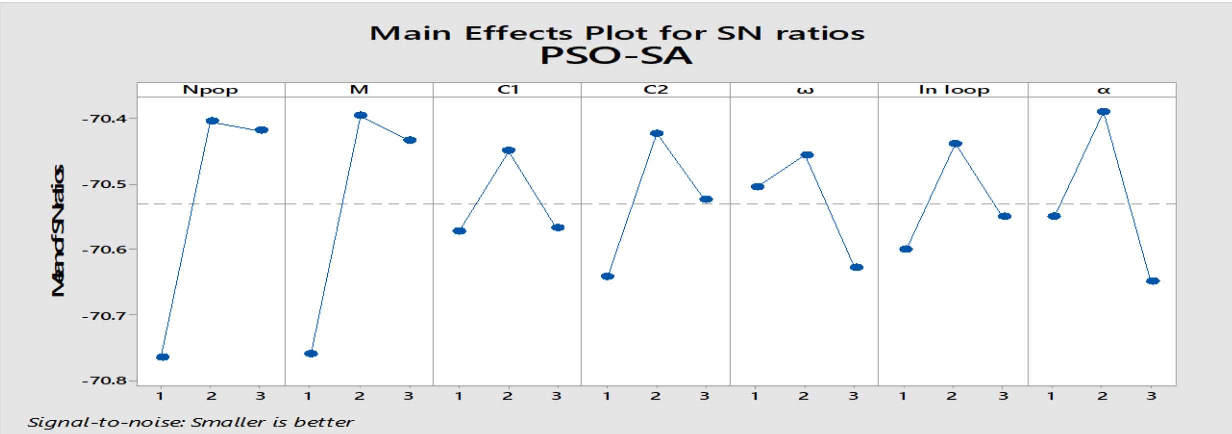

Fig. 10. Factor level trend of PSO-PSAalgorithm

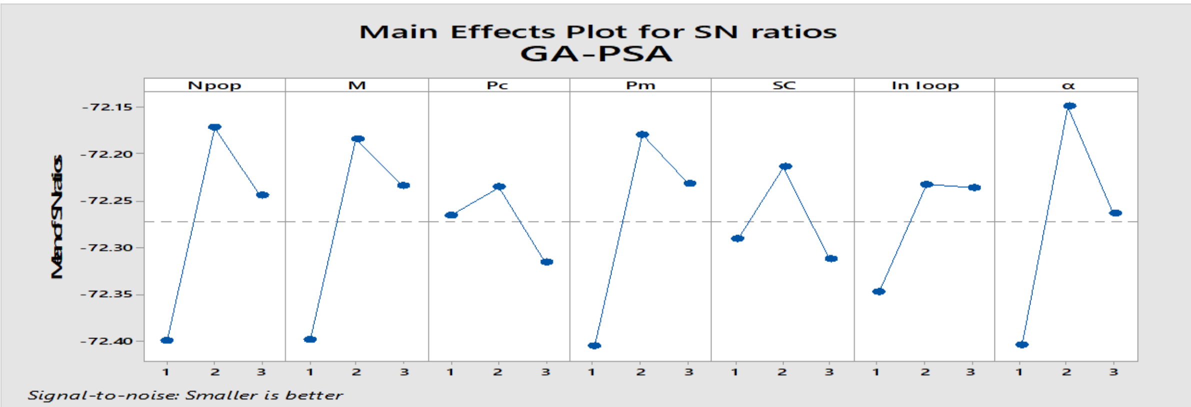

Fig. 11. Factor level trend of GA-PSAalgorithm

Table 5
Solutions found by the algorithms of the small-sized test problems

Problem #	Budget type	NOJ (*j*)	NOS (*s*)	CPLEX Makespan	CPLEX CPU time	CPLEX Gap (%)	PSO-PSAΠ Makespan	PSO-PSAΠ CPU time	PSO-PSAI Makespan	PSO-PSAI CPU time	GA-PSA Makespan	GA-PSA CPU time
1	1	4	2	205	22	0	205	22	205	26	205	29
	2			210	22	0	210	23	229	37	233	39
2	1			209	35	0	209	37	216	52	219	55
	2			217	36	0	218	35	225	47	228	49
3	1		3	214	95	0	214	97	215	109	227	115
	2			240	91	0	241	94	248	117	253	122
4	1			218	155	0	218	156	227	160	249	163
	2			223	150	0	224	152	237	182	250	186
5	1		4	232	185	0	235	191	243	305	247	321
	2			236	184	0	238	187	246	287	249	328
6	1			249	235	0	249	238	257	398	260	415
	2			250	234	0	252	236	264	382	262	402
7	1	5	2	237	645	0	238	651	248	1039	251	1100
	2			241	647	0	245	650	252	1077	255	1138
8	1			253	822	0	256	831	265	1005	270	1176
	2			262	830	0	268	833	273	1015	276	1165
9	1		3	270	882	0	274	890	280	996	284	1034
	2			280	890	0	282	893	292	1016	298	1086
10	1			289	918	0	292	923	302	987	305	1150
	2			304	922	0	308	931	317	1083	321	1154
11	1		4	335	1012	0	337	1024	346	1056	350	1149
	2			348	1048	0	350	1055	355	1088	369	1047
12	1			326	943	0	329	951	336	1008	339	1062
	2			351	950	0	353	958	363	1032	368	1094
13	1	6	2	349	1083	0	352	1099	361	1125	367	1139
	2			362	1116	0	366	1128	374	1176	379	1182
14	1			351	1043	0	355	1074	363	1133	369	1273
	2			366	1055	0	371	1094	381	1156	385	1258
15	1		3	350	1103	24.54	354	1127	367	1176	369	1201
	2			372	1144	28.55	376	1185	392	1192	395	1282
16	1			347	1078	29.17	351	1112	358	1185	363	1246
	2			360	1123	31.47	363	1175	371	1211	374	1267
17	1		4	361	1197	68.35	364	1225	373	1312	377	1424
	2			367	1255	66.92	371	1292	379	1332	383	1487
18	1			358	1205	74.18	363	1243	370	1294	376	1431
	2			394	1270	78.53	398	1293	384	1345	399	1481
Average	1			268.28	703.22	10.90	288.61	716.17	296.22	798.11	301.50	860.17
	2			299.06	720.39	51.37	301.89	734.11	310.11	820.33	315.39	875.94

Gap (%) = CPLEX optimality gap

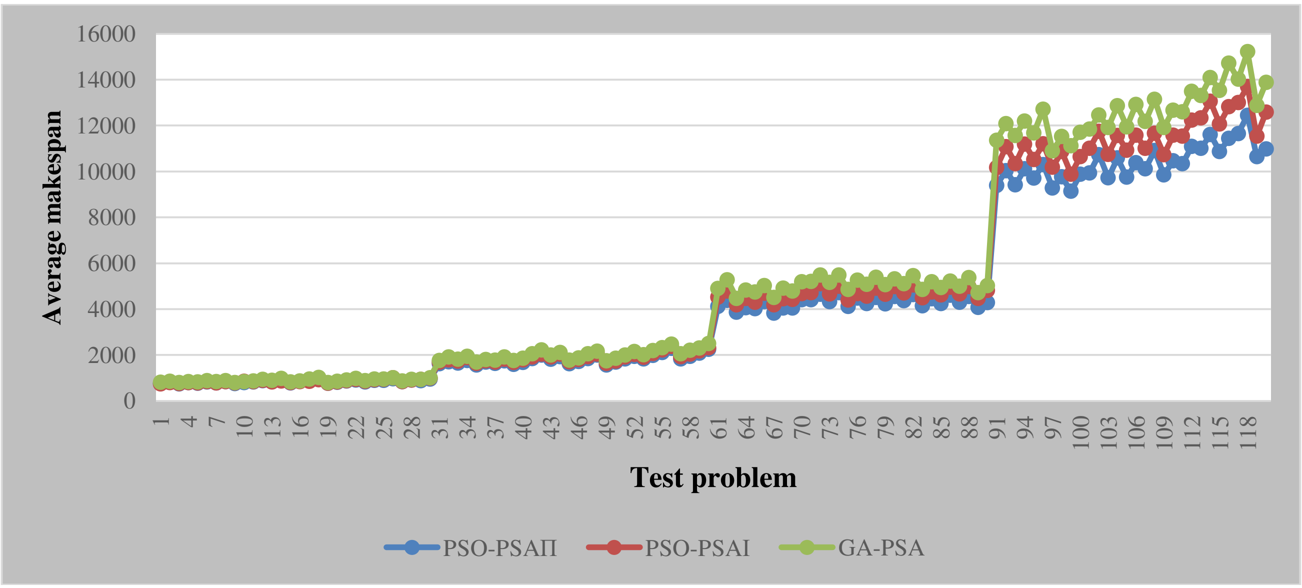

Fig. 12. The performance of proposed hybrid algorithmsof the large-sized test problems in terms of average makespan

Table 6
Solutions found by the algorithms of the large-sized test problems

Problem #	Budget type	NOJ (j)	NOS (s)	PSO-PSAΠ		PSO-PSAI		GA-PSA	
				Makespan	CPU time	Makespan	CPU time	Makespan	CPU time
1	1	20	3	958	2231	937	2265	1009	2319
	2			1005	2277	996	2309	1055	2393
2	1			941	2226	959	2261	992	2318
	2			994	2202	1002	2278	1038	2373
3	1			964	2230	988	2278	1031	2302
	2			1032	2282	1042	2266	1086	2285
4	1			984	2236	981	2279	1035	2333
	2			1034	2224	1034	2273	1088	2383
5	1			952	2227	985	2291	993	2329
	2			993	2265	1071	2281	1039	2382
6	1		6	1127	2385	1138	2450	1161	2483
	2			1179	2463	1197	2408	1231	2479
7	1			1152	2397	1131	2438	1201	2508
	2			1223	2467	1174	2464	1276	2579
8	1			1080	2395	1097	2428	1119	2469
	2			1147	241	1142	2426	1165	2486
9	1			1188	2373	1149	2443	1245	2471
	2			1252	2434	1223	2406	1297	2532
10	1			1164	2380	1170	2431	1104	2468
	2			1218	2379	1228	2426	1159	2464
11	1		9	1174	2417	1178	2458	1216	2529
	2			1210	2490	1234	2524	1278	2531
12	1			1120	2416	1155	2471	1178	2521
	2			1190	2490	1225	2450	1249	2480
13	1			1197	2433	1233	2443	1257	2471
	2			1260	2476	1304	2441	1307	2457
14	1			1143	2421	1145	2465	1169	2500
	2			1215	2437	1213	2495	1233	2549
15	1			1187	2417	1233	2476	1236	2511
	2			1250	2412	1293	2485	1309	2521
16	1	50	3	2617	2903	2684	3021	2767	3197
	2			2706	2990	2793	3030	2912	3137
17	1			2652	2911	2730	3028	2813	3134
	2			2758	2945	2881	3058	2936	3216
18	1			2562	2910	2639	3074	2697	3251
	2			2693	2956	2753	3124	2799	3339
19	1			2635	2908	2697	3010	2768	3159
	2			2740	2952	2813	3013	2916	3130
20	1			2589	2901	2680	3050	2757	3180
	2			2676	2936	2789	3042	2858	3270
21	1		6	2839	3010	2920	3141	3055	3300
	2			2992	3097	3035	3085	3218	3378
22	1			2815	3009	2918	3115	2994	3240
	2			2922	3085	3036	3075	3106	3233
23	1			2620	3010	2698	3116	2770	3255
	2			2725	3103	2801	3146	2873	3319
24	1			2843	3031	2933	3199	3050	3352
	2			2996	3019	3035	3286	3169	3444
25	1			2568	3022	2637	3184	2750	3300
	2			2687	3073	2727	3260	2860	3269
26	1		9	2827	3181	2911	3324	3004	3491
	2			2943	3178	3024	3427	3153	3571
27	1			2825	3186	2898	3318	3009	3516
	2			2989	3225	3064	3399	3187	3460
28	1			3109	3164	3214	3288	3325	3399
	2			3296	3137	3373	3273	3478	3465
29	1			2827	3181	2931	3280	3051	3390
	2			2948	3203	3043	3369	3210	3338
30	1			3087	3193	3178	3344	3299	3517
	2			3245	3292	3322	3380	3489	3476
31	1	60	3	4117	3578	4525	3834	4900	4070
	2			4388	3690	4793	3926	5726	4084
32	1			3861	3596	4194	3837	4468	4035
	2			4059	3616	4466	3763	4823	4125
33	1			4026	3594	4304	3847	4752	4097
	2			4327	3470	4684	3786	5026	4179
34	1			3828	3586	4192	3876	4507	4134
	2			4043	3670	4452	3968	4910	4243
35	1			4048	3589	4424	3839	4787	4112

	2			4412	3694	4675	3840	5185	4207
36	1		6	4406	3709	4712	3999	5205	4221
	2			4650	3692	5107	3922	5487	4180
37	1			4331	3729	4658	4025	5161	4246
	2			4711	3764	4993	3975	5487	4230
38	1			4123	3701	4403	4030	4860	4351
	2			4479	3788	4675	4128	5256	4483
39	1			4250	3700	4570	3912	5072	4240
	2			4513	3809	4916	3988	5392	4342
40	1			4236	3722	4641	3916	5060	4157
	2			4552	3745	4982	3943	5322	4256
41	1		9	4668	3954	5008	4221	5415	4494
	2			4927	4002	5350	4189	5760	4638
42	1			4443	4024	4793	4248	5155	4499
	2			4749	3961	5051	4177	5592	4553
43	1			4249	3947	4626	4156	4943	4431
	2			4607	4001	4958	4232	5218	4511
44	1			4705	3958	5156	4227	5302	4449
	2			4857	3937	5283	4232	5673	4477
45	1			4569	3949	4761	4226	5018	4554
	2			4985	4037	4912	4168	5325	4537
46	1	100	3	9395	4845	10169	5459	11358	6083
	2			10037	4805	11081	5445	12081	6024
47	1			9416	4857	10344	5313	11573	6022
	2			10112	5003	11193	5335	12201	5913
48	1			9712	4799	10521	5446	11681	5957
	2			10299	4866	11197	5561	12717	5860
49	1			9279	4811	10184	5301	10896	5982
	2			9762	4962	10869	5307	11522	5906
50	1			9137	4824	9879	5423	11125	6040
	2			9880	4756	10651	5410	11700	6227
51	1		6	9938	4968	11013	5629	11839	6274
	2			10733	5022	11748	5732	12457	6258
52	1			9725	4972	10744	5587	11916	6240
	2			10589	4932	11570	5544	12867	6367
53	1			9747	4920	10919	5415	11947	6167
	2			10378	5059	11581	5340	12926	6248
54	1			10115	4937	11005	5566	12176	6305
	2			10917	4948	11661	5489	13142	6327
55	1			9844	5017	10732	5479	11919	5978
	2			10468	4972	11580	5498	12673	6091
56	1		9	10733	5428	11540	6568	12597	7390
	2			12093	5366	12239	6677	13491	7630
57	1			11007	5799	12329	6558	13316	7855
	2			11613	5817	13058	6643	14096	7953
58	1			10873	5849	12069	6557	13537	7462
	2			11434	5997	12826	6438	14717	7577
59	1			11654	5892	13002	6694	14018	7583
	2			12466	6020	13705	6782	15217	7467
60	1			10650	5824	11537	6502	12880	7893
	2			10987	5742	12587	6697	13879	7834
Average	1			4547.18	3579.70	4900.02	3850.98	5307.42	4158.90
	2			4842.08	3617.55	5211.50	3867.98	5656.53	4193.93

NOJ (j) =number of jobs
NOS (s) =number of stations

After solving all test problems, the results showed that in the all test problems, PSO-PSAΠhas better performances in terms of makespan and CPU time. The comparisons of makespan show that the PSO-PSAΠ provides better solution quality with difference average RPD (ARPD) of 5.24, and 12.02 in comparison to PSO-PSAI, and GA-PSA, respectively. Also, we can see that in the comparison of CPU times, PSO-PSAΠ give better results in terms of difference ARPD of 6.04, and 13.42 against PSO-PSAI, and GA-PSA. Figures 13 and 14 show the mean plot of the CPU time and makespan of the proposed metaheuristic algorithms, respectively.According to computational results, it can be inferred that in small-sized test problems, all the algorithms approximately have the same computational effort. Therefore, it can be proved that all the metaheuristic algorithms are able to reach optimal/near optimal solutions.But, as the sizes of the problems are increased, the difference between algorithms is more revealed so that PSO-PSAΠ overcomes all other algorithms. However, the PSO-PSAI, and GA-PSA are highly affected by the problem size so that by increasing it, the CPU time is exponentially increased. Figure 15 depicts the 95% confidence intervals of makespan and Figure 16 showsthe 95% confidence intervals for RPD among the proposed algorithms.

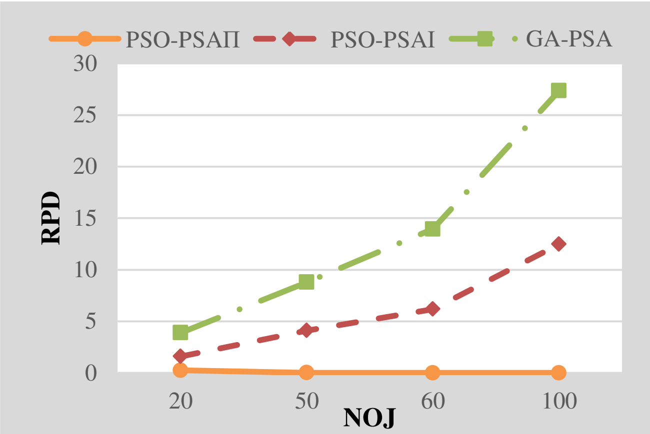

Fig. 13. Means plot of CPU time of hybrid algorithms

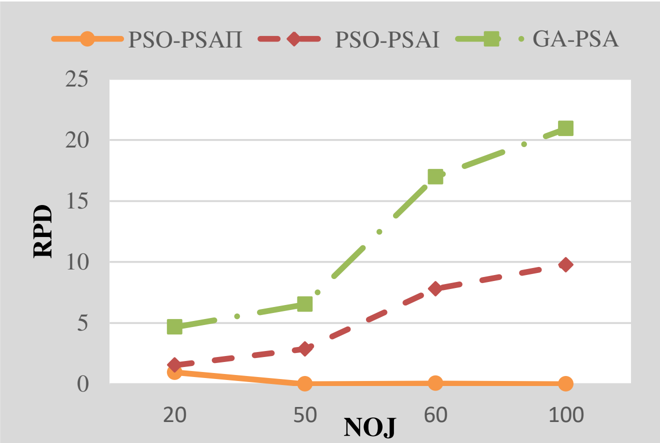

Fig. 14. Means plot of makespan of hybrid algorithms

5. Conclusion

In this study, we developed the stochastic flexible flow shop scheduling problem (SFFSSP) considering fixed interval preventive maintenance (PM) and budget constraint. This new type of problem which is the main contribution of the research is presented an integrated mathematical model which is capable to consider all of the influential factors on makespan. In theproposed SFFSSP, there is a buffer between each stage, if all machines are busy or under PM action, the job can wait in the buffer. The budget constraint controls the holding costs of total jobs in all buffers should not be greater than available budget. The proposed model considers not only the preventive maintenance, but also stochastic processing time and budget constraint. By integrating all of these subjects the model will reflect the real performance of a SFFSSP. Since the problem was strongly NP-hard, three hybrid algorithms, including PSA with two types of PSO algorithms (PSO-PSAI and PSO-SAП), and a GA (GA-PSA) were proposed to solve the model, which have suitable quality solutions in the literature.To compare the computational results, we tested two types of test problems containing 18 small sized test problems and 60 large sized test problems. As the results showed, the PSO-PSAП algorithm provided better quality solutions in both makespan and CPU time among the test problems. Also, the higher performance of proposed PSO-PSAП algorithm with respect to other algorithms is more revealed in the large sized test problems.

The presented model is still open to considering other options, such as sequence dependent setup time, machine random breakdown, and the problem of job availability at time zero. Also, it might be exciting in working on bi-objective SFFSSP's, which the other objective function could be minimizes the maximum tardiness.Another research direction could be incorporating different transportation types to transport jobs between each stage. Additionaleffort can try to solve model by developing a new solution methodology such as a new hybrid algorithm or a new population-based algorithm can be investigated.We assumed that each job has a same holding cost at each intermediate buffer, but it is different at each stage. Another aspect deserving future efforts is to consider that the holding costs of each job are different at any intermediate buffer.

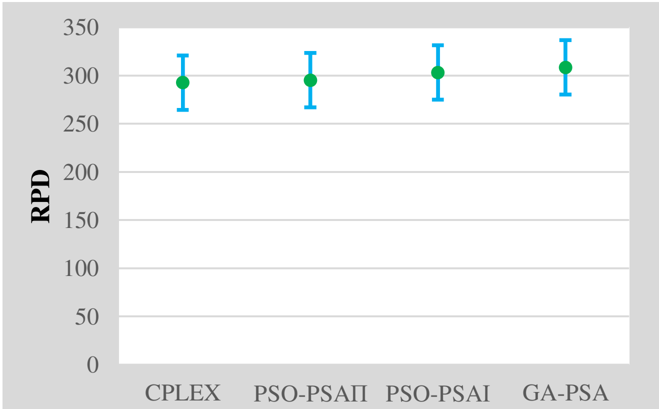

Fig 15. The 95% confidence intervals of makespanof the small-sized test problems

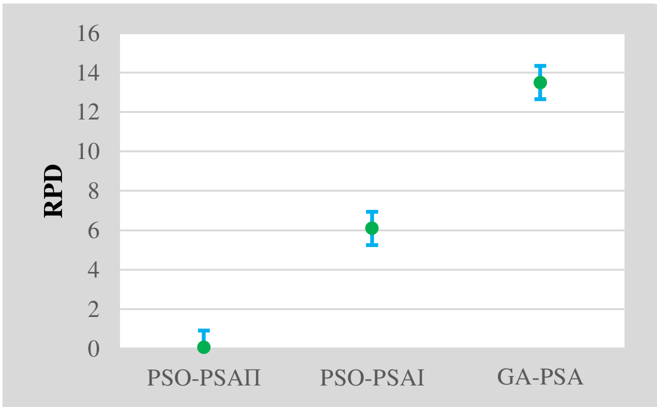

Fig 16. The 95% confidence intervals of makespanof the large-sized test problems

References

Akrami, B., Karimi, B., Hosseini, S.M.M., (2006). Two metaheuristic methods for the common cycle economic lot sizing and scheduling in flexible flow shops with limited intermediate buffers: the finite horizon case.*Applied Mathematics and Computation*, 183, 634- 645.

Al-Hinai, N., ElMekkawy, T.Y., (2011). Robust and stable flexible job shop scheduling with random machine breakdowns using a hybrid genetic algorithm.*International Journal of Production Economics*, 132, 279-291.

Almeder, C., Hartl, R.F., (2013). A metaheuristic optimization approach for a real-world stochastic flexible flow shop problem with limited buffer.*International Journal of Production Economics*, 145, 88-95.

Arnaout, J.P., (2014). Rescheduling of parallel machines with stochastic processing and setup times.*Journal of Manufacturing Systems*, 33 (3), 376-384.

Brucker, P., Kramer, B., (1995). Shop scheduling problems with multiprocessor tasks on dedicated processors. *Annals of Operations Research*, 50, 13–27.

Choi, S.H., Wang, K., (2012). Flexible flow shop scheduling with stochastic processing times: A decomposition-based approach.*Computers & Industrial Engineering*, 63, 362-373.

Fahmy, S.A., Sherif, A., Balakrishnan, S., ElMekkawy, T.Y., (2009). A generic deadlock-free reactive scheduling approach.*International Journal of Production Research*, 47 (20), 5657-5676.

González-Neira, E.M., García-Cáceres, R.G., Cabellero-Villalobos, J.P., Molina-Sánchez, L.P., Montoya-Torres, J.R., (2016). Stochastic flexible flow shop scheduling problem under quantitative and qualitative decision criteria.*Computer and Industrial Engineering*, 101, 128-144.

Gupta J.N.D., (1988). Two stage hybrid flow shop scheduling problem. *Journal of Operational Research Society*, 39(4), 359-64.

Hoogeveen, J.A., Lenstra, J.K., Veltman, B., (1996). Minimizing the makespan in a multiprocessor flow shop is strongly NP-hard. *European Journal of Operational Research*, 89, 172-175.

Kennedy, J.,Eberhart,R.C.,(1995). Particle swarm optimization.*Proceeding of IEEE International Conference on Neural Network*, Piscataway: IEEE 4, 1942-1948.

Kianfar, K., FatemiGhomi, S.M.T., OroojlooyJadid, A., (2012). Study of stochastic sequence-dependent flexible flow shop via developing a dispatching rule and a hybrid GA.*Engineering Applications of Artificial Intelligence*, 25, 494-506.

Koulamas, C., Kyparisis, G.J., (2000). Scheduling on uniform parallel machines to minimize maximum lateness.*Operations Research Letters*, 24(6),175-179.

Li, J.Q., Pan, Q.K., (2015). Solving the large-scale hybrid flow shop scheduling problem with limited buffers by a hybrid artificial bee colony algorithm. *Information Sciences*, 316:487–502.

Lin, SW., Ying, K.C., (2013). Minimizing makespan in a blocking flowshop using a revised artificial immune system algorithm.*Omega*, 2 (4), 383-389.

Lin, J.T., Chen, C.M., (2015). Simulation optimization approach for hybrid flow shop scheduling problem in semiconductor back-end manufacturing.*Simulation Modeling Practice and Theory*, 51, 100-114.

Montgomery, D.C. (2005). Design and analysis of experiments. *Arizona, John Wiley and Sons*.

Norman, B.A., Bean, J.C., (1999). A genetic algorithm methodology for complex scheduling problems. Naval Research. Logistics, 46: 199-211.

Osman, I.H., Kelly, J.P., (1996). Metaheuristics: an overview. *Metaheuristics: theory and applications.* Boston: Kluwer Academic Publisher, 1-21.

Pinedo, M., Chao, X., (1999). Operations scheduling with applications inmanufacturing and services. *New York: McGraw-Hill*.

Pinedo, M., (2002). Scheduling theory, algorithms, and systems. *Englewood Cliffs, New Jersey: Prentice-Hall. Chapter 2*.

Poli, R., Kennedy, G., Blackwell, T., (2007). Particle swarm optimization: an overview. *Swarm Intelligence*, 1(3), 33-57.

Rabiee, M., Sadeghi Rad, R., Mazinani, M., Shafaei, R., (2014). An intelligent hybrid metaheuristic for solving a case of no-wait two-stage flexible flow shop scheduling problem with unrelated parallel machine.*International Journal of Advanced Manufacturing Technology*, 71: 1229-1245.

Rahmani, D., Heydari, M., (2014). Robust and stable flow shop scheduling with unexpected arrivals of new jobs and uncertain processing times. *Original Research Article*, 33 (1), 84-92.

Rahmani, D., Ramezanian, R., (2016). A stable reactive approach in dynamic flexible flow shop scheduling with unexpected disruptions: A case study. *Computers & Industrial Engineering*, 98, 360-372.

Sangsawang, C., Sethanan, K., Fujimoto, T., Gen, M., (2015). Metaheuristics optimization approaches for two-stage reentrant flexible flow shop with blocking constraint.*Expert Systems with Applications*, 42, 2395–2410.

Singh, M.R., Mahapatra, S.S., (2012). A swarm optimization approach for flexible flow shop scheduling with multiprocessor tasks.*International Journal of Advanced Manufacturing Technology*, 62: 267-277.

Sukkerd, W., Wuttipornpun, T., (2016). Hybrid genetic algorithm and tabu search for finite capacity material requirement planning system in flexible flow shop with assembly operations. *Computers & Industrial Engineering*, 97, 157- 169.

Tang, D., Dai, M., Salido, M.A, Giret, A., (2016). Energy-efficient dynamic scheduling for a flexible flow shop using an improved particle swarm optimization. *Computers in Industry*, 81, 82–95.

Tran, T.H., Ng, K.M. (2011). A water flow algorithm for flexible flow shop scheduling with intermediate buffers.*Journal of Scheduling*, 14, 483-500.

Villemeur, A., (1991). Reliability, availability, maintainability and safety assessment. *USA: Wiley*.

[32] Wang, X., Tang, L., (2009). A Tabu search heuristic for the hybrid flowshop scheduling with finite intermediate buffers. *Computers & Operations Research*, 36 (3), 907–918.

Wang, K., Choi, S.H., (2014). A holonic approach to flexible flow shop scheduling under stochastic processing times.*Computers & Operations Research*, 43, 157-168.

Wardono, B., Fathi, Y., (2004). A Tabu search algorithm for multi-stage parallel machine problem with limited buffer capacities. *European Journal of Operational Research*, 155 (2), 380-401.

Zabihzadeh, S.R., Rezaeian, J., (2015). Two metaheuristic algorithms for flexible flow shop scheduling problem with robotic transportation and release time. *Applied Soft Computing*, 40, 319-330.

This article can be cited: Raissi, S., Rooeinfar, R. & Ghezavati, V. R. (2019)
Three Hybrid Metaheuristic Algorithms for Stochastic Flexible Flow Shop Scheduling Problem with Preventive Maintenance and Budget Constraint *Journal of Optimization in Industrial Engineering*. 12 (2), 131-147.

http://www.qjie.ir/article_543744.html
DOI: 10.22094/JOIE.2018.242.1532