Journal of Industrial and Systems Engineering
Vol. 10, Special Issue on Scheduling, pp. 78-91
Autumn (November) 2017

A particle swarm optimization for minimizing total earliness/tardiness costs of two-stage assembly flowshop scheduling problem in a batched delivery system

Amir Abbas Shojaie[1*], Sina Sajedi[1]

[1]*Department of Industrial Engineering, Islamic Azad university, South Tehran Branch, Tehran, Iran*
amir@ashojaie.com, sina_sad11@yahoo.com

Abstract
This paper considers a two-stage assembly flowshop scheduling problem in which a number of single-item is ordered with a due date. Each of them is manufactured by assembling several different pieces that must be processed on first stage in a flowshop scheduling system. When all parts of each product are completed in the first stage, they are assembled into a final product on an assembly machine in the second stage. Due to delivery cost, in order to reduce this cost, completed products can be held until completion of some other products to be delivered in a same batch. The proposed problem addresses scheduling a set of operation with specific due date that must be processed on all of the machines in the two stages and dispatched to customers in a batch delivery system. The aim is to minimize total weighted earliness/tardiness and delivery costs. As the problem is demonstrated to be NP-hard, a genetic algorithm (GA) and a particle swarm optimization (PSO) are presented to solve the problem in real scales. In order to evaluate the proposed algorithms, a number of problems are solved by them and the results are compared. The computational results illustrate that the proposed PSO has a qualifier performance than the GA.

Keywords: Two-stage assembly flow shop, earliness, tardiness, batch delivery.

1- Introduction

The two- stage assembly flow shop problem (TAFSP) has two stages with *m* parallel and independently machines that is located at the stageone and only one assembly machine at the stagetwo. There are *n* jobs and each job has *m+1* operation that *m*operations are performed by *m* machine in parallel at the first stage and the last operation is done by the assembly machine in assembly stage. This system can be used for producing a variety of products by assembling and combining different set of parts. Assembly flow shop problem (AFSP) has several applications in real life problem that are mentioned in the literature review, Potts et al. (1995) investigated an application in personal computer manufacturing where central processing units, hard disks, monitors, keyboards,…,etc are manufactured at the first stage and these parts assembled at the last stage. Lee et al. (1993) described its application in a fire engine assembly plant, in which the bodyand chassis of fire engine are manufactured in parallel and in two different departments.

*Corresponding author

ISSN: 1735-8272, Copyright c 2017 JISE. All rights reserved

After completion of body and chassis, and delivery of the engine (purchased from outside), they fed to the assembly line to assemble the final fire engine. Allahverdi and Al-Anzi (2007), considered another application in queries scheduling on distributed data base systems.
The remainder of this study is organized as follows. Section 2 presented a literature review of two-stage assembly flow shop scheduling. Section 3 describes the problem. The proposed meta-heuristics are described in section 4. Section 5 presents the experimental design which analyzes parameters of problem and compares the results with other meta-heuristics in literature review. Finally, Section 6 presents the conclusions and directions for future work.

2- Literature review

AFSP was introduced by Lee et al. (1993) and Potts et al. (1995). Lee et al. (1993) investigated AFSP with considering two machines at the first stage. Also Potts et al. (1995) studied it with arbitrary machines at the stage1. They illustrate that AFSP for two machines at the first stage with goal of minimizing the makespan is Np-hard. Hariri and Potts (1997), proposed a lower bound and the defined dominance relations. Another branch and bound algorithm was proposed by Houari and Daous (1999). Tozkapan et al. (2003) studied TAFSP with objective of minimizing the total weighted flow time and proposed a lower bound dominance relation and used a heuristic algorithm to derive an initial upper bound. Sun et al. (2003) considered AFSP with three-machinesand objective of minimizing the makespan then they applied a heuristic algorithm for solving problem. Al-Anzi and Allahverdi (2007) investigated AFSP with total completion time; they obtained optimal solutions for two special cases and applied simulated annealing (SA), Tabu-search (TS) and hybrid Tabu search (Ntabu). Allahverdi and Al-Anzi (2007) proposed three meta-heuristics algorithm that are named SA, self-adaptive differential evolutionary algorithm (SDE) and ant colony optimization (ACO) for obtaining solutions to minimize the weighted sum of makespan and mean completion time.They demonstrated SA achieved good solution in a reasonable time compare another proposed meta-heuristic algorithms. Sung and Kim (2008) studied a two-stage multiple- machine assembly scheduling problem for minimizing the sum of completion times, the first stage has two independent machines and each of which produces its own type of components and assembly stage wasconsisted of two identical and independent parallel machines. They proposed a simple heuristic for solving the problem. Sung and Juhn (2009) investigated TASFP that each job assembled with two types of components and objective was the minimizing of makespan. For the assembly, one type of the components was outsourced subject to job dependent lead time but other type was fabricated in-house at the first stage. They proposed three heuristics and a branch and bound to solved random generated test problems. Allahverdi and Al-Anzi (2007) investigated TAFSP with set up times by minimizing the total completion time and they reached a dominance relation. They proposed three heuristics:hybrid tabu-search (Ntabu) and SDE and new self-adaptive differential evolutionary algorithm (NSDE). Shokrollahpour et al. (2011) studied TAFSP and presented an imperialist competitive algorithm (ICA) for verifying of the proposed ICA thenthey compared it with Allahverdi and Al-Anzi (2007). The computational resultsdemonstrated in which ICA reach to better solutions but it consumed more time. Torabzadeh and Zandieh (2010) investigated TAFSP by minimizing the weighted sum of completion time and makespan. They entered concepts of cloud theory into SA and called CSA then they compared it with SA based on quality of solutions. The computational result showed that CSA was better than SA in quality and time. Mozdgir et al. (2013) studied TASFP with multiple non-identical assembly machines in stage onefor minimizing the weighted sum of makespan and mean completion time. Seidgar et al. (2014) proposed a mixed integer linear mathematic model for TASFP and solved small sized problems with exact approach namely Branch and Bound (B&B) then for solving the proposed problem in large sized problem they hybridized ICA with artificial neural network (ANN) and compared with CSA that introduced by Torabzadeh and Zandieh (2010). The computational result demonstrated ICA is better than CSA. Xiong and Xing (2014) considered the distributed two-stage assembly flow-shop scheduling problem to minimize the weighted sum of makespan and mean completion time. The results statistically showed both GA-RVNS and VNS have a high performance in compare with the GA without RVNS-based local search step (GA-NOV). Seidgar et al. (2015) studied TAFSP with machine breakdown and preventive maintenance. They applied ICA and ANN for handling computational complexity and entered simulation approach to handle structure complexity. In other hand Ju-Yong Lee, et al. (2016) defined a simple assembly type flow shop case, and problems with various characteristics and develop an exact algorithm with a two-stage assembly-type flow shop scheduling with the objective of minimizing the total tardiness and first stage consists of two independent machines and the second stage consists of a single machine. Kazemi, et al. (2017) develop a model to minimize the sum of tardiness plus delivery costs. They defined the assembly flow shop scheduling problem with the objective function has not been addressed so far. The mathematical model happens to be a mixed integer nonlinear programming model which cannot guarantee to reach the solution at reasonable time therefore they developed the imperialist competitive algorithm (ICA) and a hybrid algorithm (HICA) by incorporating the dominance relations.

According to the mentioned literature review, two stage assembly flow shop is one of the most important shop floor, which is used in practical production systems. On the other hand, just in time scheduling plays an important role in customer satisfaction, which is critical in today's competitive markets. In addition, due to increasing cost and environmental effects of transportation activities, manufacturers are forced to reduce such activities. Therefore, in this paper, we concentrate on a just in time two stage assembly flow shop considering batch delivery system, which is more efficient in transportation point of view.

3- Problem definition

In this problem, *n* single-item products are manufactured that each is made by assembling several different parts. The parts are produced in a flow-shop at the first stage consisting *m-1* parallel machines and then an assembly machine in the second stage assembled them into a product. In order word, a set of *n* multiple-operation jobs that each job requires *m* operations that *m-1* operations are done at the first stage on *m-1* parallel machines then last operation is performed at assembly stage. Operations of each job in assembly stage starts only after completion of operations of the job on the machines in the first stage. Each job is completed after finishing all of its operations in both of stages. All of the jobs are available at the beginning of the process and preemption is not allowed. The machines cannot be idle when there is any available job. In addition, sequences of the jobs are the same on the all of the machines. Jobs are delivered by through batches that have delivery costs. Delivery cost of each batch is independent from the number jobs which it includes. Therefore, in order to reduce costs, jobs could be held to be delivered together in a same batch. As the following mathematical model illustrates, the problem looking for an appropriate sequence and assignment to batches that minimizes the costs such as earliness, tardiness, and delivery costs. Following the indices, parameters, and variables are shown for the proposed mathematical model.

Indices

i	The number of jobs(i=1,…,n)
j	Index of sequence for jobs (j=1,…,n)
k	Index of batch for jobs(k=1,…,n)
l	The number of machines(l=1,…,m)

Parameters

D_i	Due date of job i
γ	Delivery cost of a batch
α_i	Earliness cost for job i
β_i	Tardiness cost for job i
p_{il}	The Processing time of job*i* on machine *l*

M A big number

Decision variables

E_i	Earliness of job i
T_i	Tardiness of job i
C_{il}	Completion time of job I on machine l

$X_{ij}\begin{cases}1 & \text{if } i \text{ th of job is assigned insequence } j\\ 0 & \text{otherwise}\end{cases}$

$Y_{ik}\begin{cases}1 & \text{if } i \text{ th of job is delivered bybatch } k\\ 0 & \text{otherwise}\end{cases}$

$Z_k\begin{cases}1 & \text{if } k \text{ th of batch is created}\\ 0 & \text{otherwise}\end{cases}$

3-1- Mathematical model

According to the mentioned parameters and variables, the mathematical model is as follows:

$$MinZ=\sum_{i=1}^{n}\alpha_i E_i+\sum_{i=1}^{n}\beta_i T_i+\gamma\sum_{k=1}^{n}Z_k \quad (1)$$

$$\sum_{j=1}^{n}X_{ij}=1 \quad i=1,\dots,n \quad (2)$$

$$\sum_{i=1}^{n}X_{ij}=1 \quad j=1,\dots,n \quad (3)$$

$$\begin{cases}C_{il}\ge P_{il}-(1-X_{i1})M\\ C_{il}\le P_{il}+(1-X_{i1})M\end{cases} \quad i=1,\dots,n,l=1,\dots,m-1 \quad (4)$$

$$\begin{cases}C_{il}\ge C_{i'l}+P_{il}-(2-X_{ij}-X_{i'j-1})M\\ C_{il}\le C_{i'l}+P_{il}+(2-X_{ij}-X_{i'j-1})M\end{cases} \quad i,i'=1,\dots,n,i'\ne i,\ j=2,\dots,n,l=1,\dots,m-1 \quad (5)$$

$$C'_i=\max_{l=1}^{m-1}\{C_{il}\} \quad i=1,\dots,n \quad (6)$$

$$\begin{cases}C_{im}\ge C'_i+P_{im}-(1-X_{i1})M\\ C_{im}\le C'_i+P_i+(1-X_{i1})M\end{cases} \quad i=1,\dots,n \quad (7)$$

$$\begin{cases}C_{im}\ge \max\{C'_i,C_{i'm}\}+P_{im}-(2-X_{ij}-X_{i'j-1})M\\ C_{im}\le \max\{C'_i,C_{i'm}\}+P_{im}+(2-X_{ij}-X_{i'j-1})M\end{cases} \quad i,i'=1,\dots,n,i'\ne i,\ j=2,\dots,n \quad (8)$$

$$\sum_{i=1}^{n}Y_{ik}=1 \quad i=1,\dots,n \quad (9)$$

$$\begin{cases}D_i\ge C_{i'l}-(1-Y_{ik})M\\ D_i\le C_{i'l}+(1-Y_{ik})M\end{cases} \quad i,i'=1,\dots,n,k=1,\dots,n \quad (10)$$

$$D_i\ge D_{i'}-(2-Y_{ik}-Y_{i'k})M \quad i,i'=1,\dots,n,k=1,\dots,n \quad (11)$$

$$D_i\le D_{i'}+(2-Y_{ik}-Y_{i'k})M \quad i,i'=1,\dots,n,k=1,\dots,n \quad (12)$$

$$D_i=d_i+T_i-E_i \quad i=1,\dots,n \quad (13)$$

$$Y_{ik} \le Z_k M \quad k = 1,...,\mathrm{n} \tag{14}$$

$$X_{ij}, Y_{ik}, Z_k \in \{0,1\} \quad i, j, k = 1,...,n \tag{15}$$

The objective function (1) minimizes the sum of earliness, tardiness, and delivery costs. Constraints (2) and (3) require that each job must be assigned to one priority and each priority must be filled by one job. Constraints (4) aim to calculate completion time of job in the first priority. Constraints (5) calculate completion time of jobs in the other priorities. Constraint (6) calculates the latest completion of each job time on the machines of first stage. Constraints (7) and (8) calculate the completion time of jobs for the first and other priority respectively in the assembly stage. Constraint (9) states that each job is assigned to a batch. Constraints (10) ensure that delivery time of each job is equal to or more than its completion time. Constraints (11) and (12) require that jobs that are delivered in the same batch have the same delivery time. Constraint (13) calculates earliness and tardiness of each job. Constraint (14) determines the batches which are receive any job to be delivered. Constraint (15) defines binary variables of the problem.

4- Solving methodologies

According to Lee et al. (1993) and Potts et al. (1995) illustrated that AFSP for two machines at the first stage with goal of minimizing the makespan is Np-hard. Therefore, our proposed problem with more complexity by minimizing earliness, tardiness in a batch delivery system definitely is also Np-hard. Meta-heuristic algorithm which is based on natural evolutionary is a common approach to optimize such problems. In this study, a genetic algorithm and a particle swarm optimization are proposed as solution procedures that are introduced in the following.

4-1- Genetic algorithm (GA)

The genetic algorithm (GA) is an optimization technique according to the evolutionary process of biological organisms in nature. The theoretical basis of GA was originally introduced by Holland (1989) and popularized by Goldberg (1995). The main reasons for prosperity of GAs are their resistance to becoming trapped in local optimum, their good performance in large-scale problems and the usability of these algorithms for a wide range of optimization problems.

In GA, a population of individuals (chromosomes), which codify potential solutions to a particular optimization problem, evolves toward better solutions through successive generations. The evolution usually starts with a population of randomly generated individuals. In each generation, the fitness of every individual in the population is evaluated considering a given objective function. The best-fit individuals are chosen from the current population for reproduction and merged or modified through crossover and mutation operators to form a new population. The new population is then used in the next replication of the algorithm. The fitness of new individuals is evaluated and least-fit population is replaced with new individuals. In general, the algorithm terminates when either a maximum number of generations has been produced or a satisfactory fitness level has been met. In order to apply a genetic algorithm to the proposed problem, some modifications are necessary. This adaptation is presented in the following.

4-1-1- Solution representation

Solution representation or chromosome is the main part of each meta-heuristic algorithm, which has direct effect on performance of the algorithms, must be designed according to problem. Chromosome in the proposed GA is presented by a matric with two rows that the first one determines priority of jobs to be scheduled and the second row is related to assignment of jobs to the delivery batches. Table1 illustrates a chromosome of the algorithm for a sample with five jobs. According to priorities of the jobs, they are processed with the order of 5, 2, 1, 4, and 3. They are delivered by three batches that job 1 and 5 are assigned to batch 1, job 2 and 3 are delivered by batch 3, and job 4 is delivered single.

Table 1. Solution representation

Job	1	2	3	4	5
Priority	0.45	0.64	0.12	0.37	0.98
Batching	1	2	2	3	1

4-1-2- Fitness function

In order to evaluate a chromosome, first, it should be interpreted to a solution to calculate the objective value. Fitness function in the proposed GA is as the same as the objective value. In other words, after decoding the chromosome to a solution, the objective value can be easily earned.

4-1-3- Genetic operators

Crossover and mutation play a significant role in performance of a genetic algorithm. Crossover operator combines two selected chromosomes through ***Roulette-wheel*** procedure and transforms some of their characteristics to generated offspring. There are two separated approaches to apply crossover for each row of the chromosome. For the first row, there is continues procedure in which a random number with uniform distribution function is generated for each job. Suppose for an element the valuesfor the parent 1 and 2 are equal to *a*, *b* respectively, and α is the generated random number. Therefore, if the values of related element in the offspring are *c*, *d*, they are obtained by $c = a + \alpha(b - a)$ and $d = b + \alpha(a - b)$ respectively. In addition, in order to apply crossover operator for the second row, there is ***guide string*** with length of the number of jobs with binary values. The offspring are produced according to their parents with the difference that the values are exchange for those elements that their related elements in the ***guide string*** are equal to one. Figures 1, illustrates the crossover operators in the proposed algorithm.

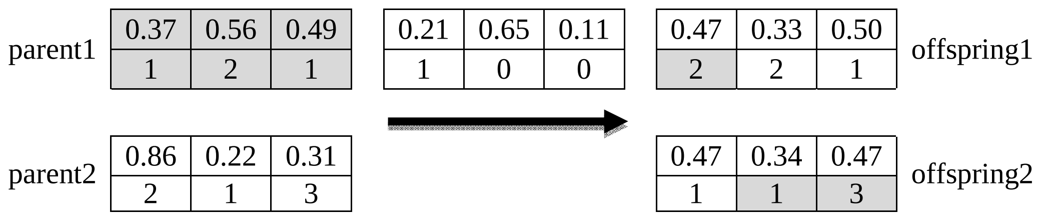

Figure 1. performance of crossover operator

Mutation is the other genetic operator that is extremely significant to eschew from local optimal solutions and preserves diversity in each population. . In this GA, mutation is implemented by *reverse* operator in which two jobs are chosen randomly and their values and the value of jobs between them are reversed figure 2, clearly shows performance of the mutation operator.

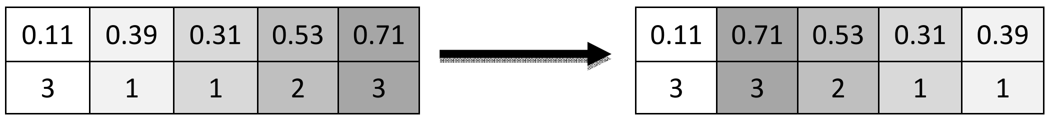

Figure 2. performance of mutation operation

4-2- Particle swarm optimization (PSO)

The particle swarm optimization (PSO) is an evolutionary algorithm developed by Kennedy and Eberhart (2006). The development of the PSO algorithm inspired by social behavior of animals such as bird flocking, fish schooling, and swarm theory. Similar to the GA, PSO is a population-based optimization approach, has fitness values to evaluate the population, updates the population and searches for the optimum with random techniques. However, unlike GA, PSO has no evolution operators such as crossover and mutation (Umarani and Selvi, 2010). During recent years, PSO algorithm has been successfully used to deal with many optimization problems due to the ease of its implementation and its fast convergence in comparison with many global optimization algorithms (Sha and Hsu, 2008).

The PSO algorithm is initialized with a population of random solutions (particles) and each potential solution is initialized with a randomized position and velocity. These particles fly in a virtual search space by following the current optimum particles. The particle motion is mainly influenced by three factors: the velocity of the particle in the latest iteration, the P-best position which is the best solution found by each particle itself so far and the G-best position which is the best solution found by the whole swarm so far. At each iteration, the position and velocity of each particle *i* toward its P-best and G-best positions are updated using equations (16) and (17), respectively. Then, the position of particle *i* in the solution space is mapped, its fitness value according to the optimization fitness function is assessed and the P-best and G-best positions are changed if necessary. This process would repeat until the termination condition is reached.

$$X_i^{t+1} = X_i^t + V_i^{t+1} \tag{16}$$

$$V_i^{t+1} = \omega\, V_i^t + \varphi_1 r_1 (Pbest_i^t - X_i^t) + \varphi_2 r_2 (Gbest^t - X_i^t) \tag{17}$$

Where V_i^t is called the velocity of particle *I* which represents the distance to be traveled by this particle from its current position. X_i^t is the current position of particle *i*, ω is the inertia weight which controls the momentum of the particle, φ_1 and φ_2 are the balance factors between the influence of individual's knowledge and social knowledge in moving the particle towards the target, r_1 and r_2 are uniformly distributed random numbers which are used to maintain diversity of the population.

A general particle swarm optimization can be adapted to for a certain problem with some modifications. Solution representation is the most important part that needs to be modified according to the problem. In addition, as PSO is has continuous approach in its evolutionary procedure, the proposed solution can be applicable for proposed PSO with converting the second row of the solution representation from discrete values to real numbers. These values are between zero and maximum number of batch that can be interpreted to batch numbers by ceiling them to the first bigger integer number.

5- Computational Results

The required data for investigating the problem's performance are seven factors, namely: range of processing time, number of job, number of machine at stage1 and weights. These factors have several levels that are illustrated in table2.

Table 2. Proposed problems' factor levels

Factor	Levels			
Processing time	Unif(15-25)			
Number of jobs	20	40	60	80
Number of machines	2	4	6	8
Alfa weight	Unif(5-10)			
Beta weight	Unif(5-10)			
Gama weight	Unif(140-200)			
Due date	Unif($\min_{i\in I}\{\sum_{m\in M} P_{im}\}$, $\frac{1}{N}\sum_{i\in I}\sum_{m\in M} P_{im}$)			

We compared the performance of the proposed algorithms that are namely (GA) and PSO. We apply the average relative percent deviation ($\overline{RPD}$) for comparing them. The relative percent deviation is computed as below:

$$RPD = \frac{(Objective\ of\ the\ metaheuristic - Objective\ of\ the\ best\ metaheuristic)}{(Objective\ of\ the\ best\ metaheuristic)}$$

There are 16 combinations for the different values of *n* and *m*. Each test problem loopsfive times and therefore, a total of 90 instances are produced for testing the algorithms. Computational results are illustrated in table 3.

Table 3. Comparison of results of the proposed GA and PSO

			Proposed GA			**Proposed PSO**		
Problem name	Job	Machine	Best solution	$\overline{RPD}$	Computational time [a]	Best solution	$\overline{RPD}$	Computational time [a]
TAFP01	20	2	565236	0.017959	10.65077	557417	0.009483026	23.95484
TAFP 02	20	4	683080	0.007867	12.04975	682834	0.005941415	26.47095
TAFP 03	20	6	491079	0.017623	10.56732	482828	0.013544161	25.3451
TAFP 04	20	8	602403	0.01017	13.49935	614292	0.024398119	47.07747
TAFP 05	40	2	4144116	0.001368	47.46451	4144937	0.000398517	65.42602
TAFP 06	40	4	4844491	0.003703	49.67983	4839320	0.000211083	62.9125
TAFP 07	40	6	4872979	0.000985	54.60746	4881897	0.002414539	74.34702
TAFP 08	40	8	5269098	0.003475	38.89637	5256586	0.000457997	98.314
TAFP 09	60	2	15664691	0.010021	129.5669	15560430	7.85325E-05	103.8281
TAFP 10	60	4	16398753	0.000241	96.62414	16397409	0.000129929	90.1243
TAFP 11	60	6	14831902	0.000545	140.4678	14836628	0.000587989	101.6985
TAFP 12	60	8	16847667	0.000862	89.99994	16847954	0.000318204	116.6463
TAFP 13	80	2	33679134	0.000902	139.865	33658177	5.70292E-05	135.6702
TAFP 14	80	4	39423146	0.000139	224.3427	39424372	0.000229649	160.465
TAFP 15	80	6	37935104	0.000354	167.9421	37935746	0.000189112	140.5656
TAFP 16	80	8	33691694	0.005099	159.4598	33657063	0.000196096	56.48657

The computational experiments are demonstrated in figures3-9.Figure 3 shows that GA and PSOare same in problem 20 job and in problems 40, 60 and 80jobs,PSO is better than GA. Figure 4 demonstrates PSO has better performance in the problems with 2,4and 6 jobs.

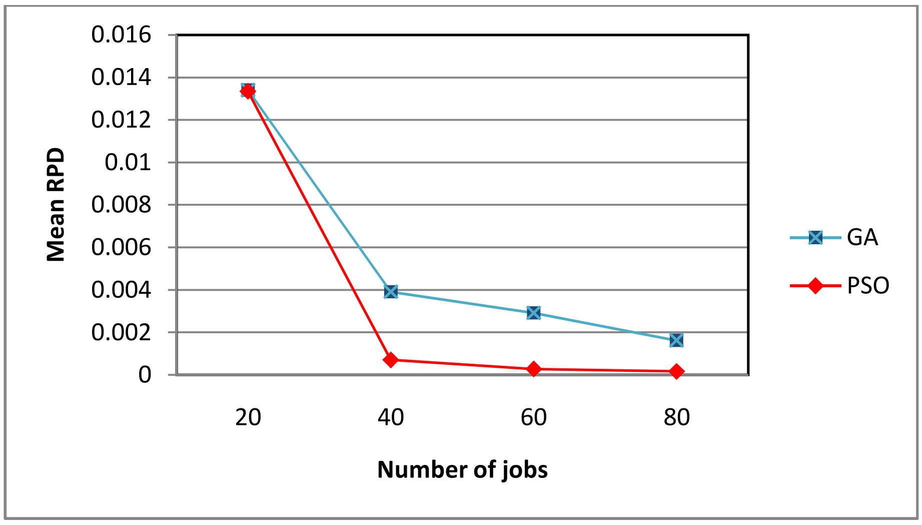

Figure3. Average relative percent deviation ($\overline{RPD}$) for different value of job

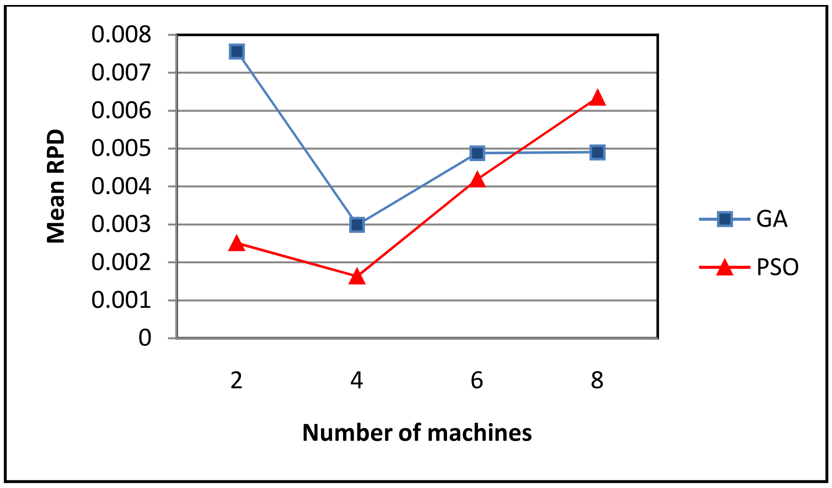

Figure4. Average relative percent deviation ($\overline{RPD}$) for different value of machine

Least significance difference (LSD) method are drawn and illustrated in figure 5.It reveals that there is not any significant difference between GA and PSO and shows that PSO has better performance than GA.

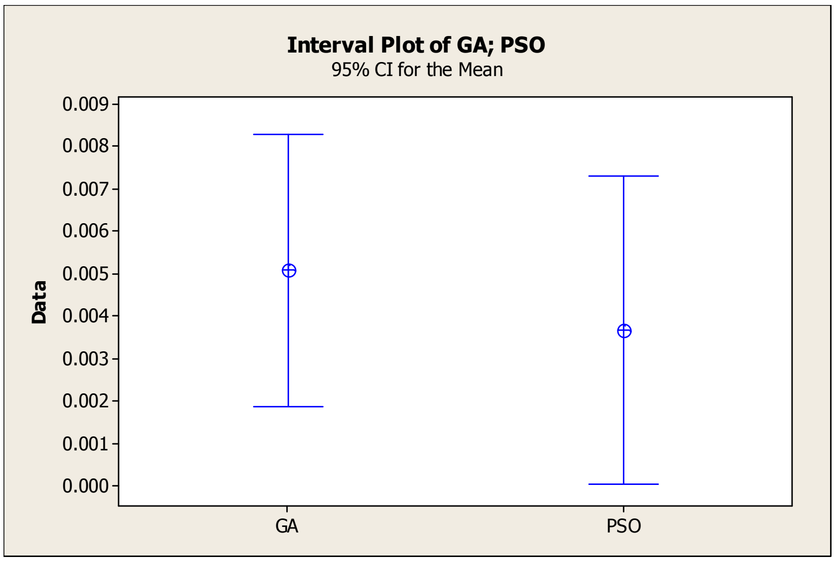

Figure5. Mean of LSD interval for the proposed algorithms

For conceptualizing the process of the proposed GA and PSO, the mean and best objective value of solutions are drawn according to the number of iterations in figures 6 and 7, *TAFP08*. As it is presented in figures 6 and 7, the mean and best objectives of solutions converge during the process of GA and PSO.

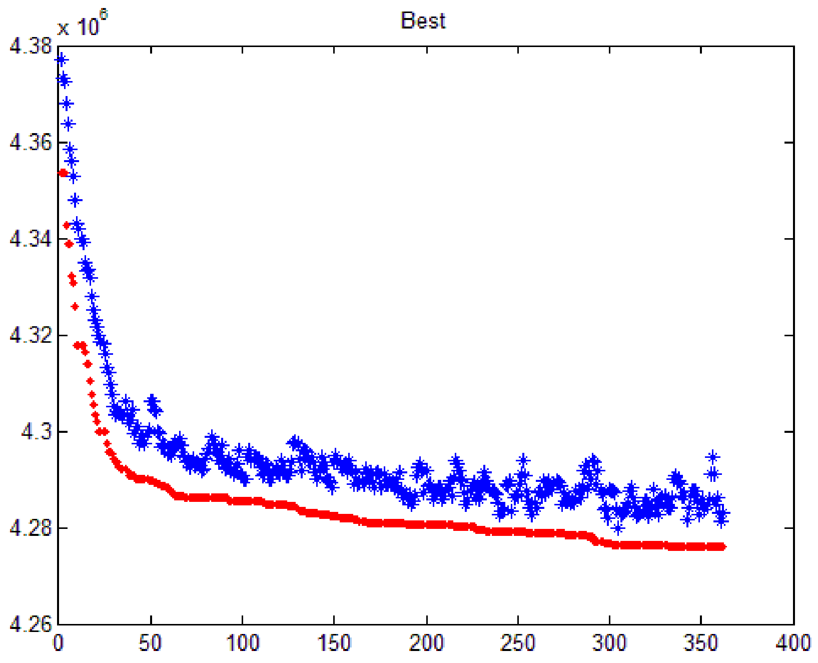

Figure 6. The best and mean objective value curves in GA (job=40, machine=8)

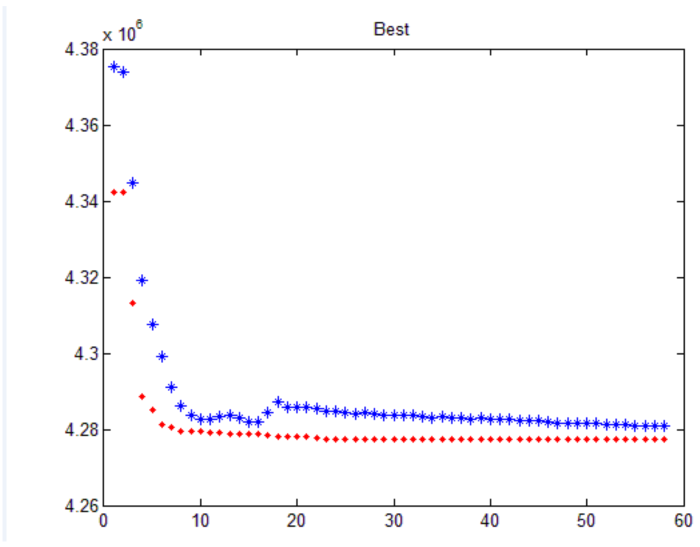

Figure 7. The best and mean objective value curves in PSO (job=40, machine=8)

Figures 8 and 9 illustrated computational time for each GA and PSO based on number of job and number of machine respectively.

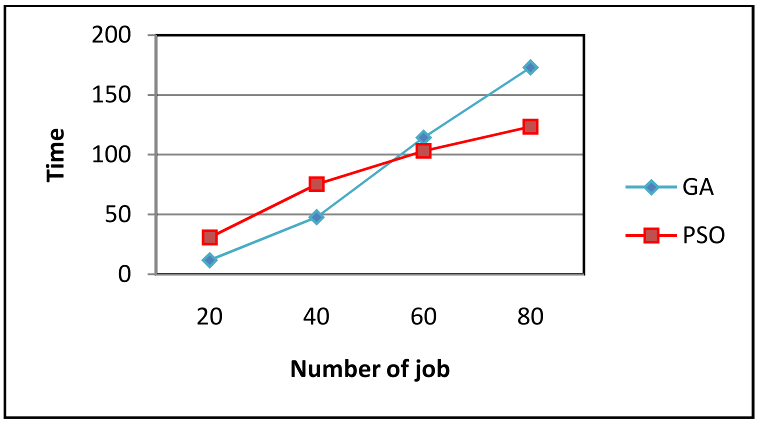

Figure8. Average of computational time for different value of number of jobs

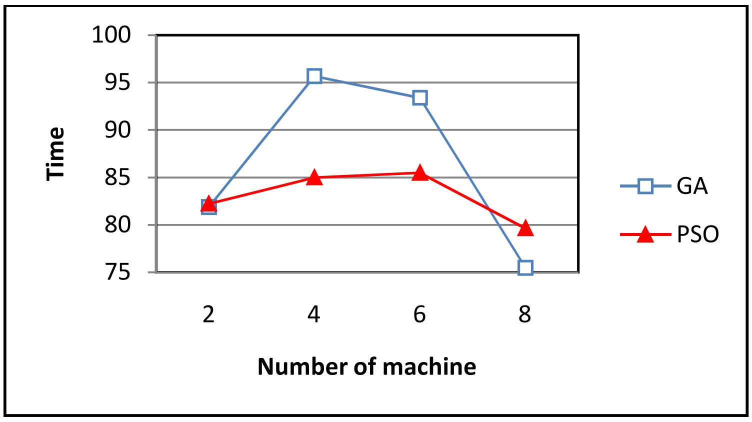

Figure9. Average of computational time for different value of number of machine

6- Conclusion and Future works

In this problem, a two-stage assembly flowshop scheduling problem has been presented that minimizes earliness/tardiness in a batch delivery system. There have been *n* multiple-operation jobs that each job requires *m* operations that *m-1* of them must do at the first stage on *m-1* parallel machines then last operation is applied at assembly stage. Assembling process for a job only starts when all of its operations have been finished at the first stage. A mathematical model is also presented to demonstrate the problem obviously. As it has been shown that the problem is Np-hard, a genetic algorithm and a particle swarm optimization are proposed for the problem that can easily solve the problem in large scale in a short time. To evaluate the performance of the algorithm they are applied for several example and their achievements are compared in view of RPD and CPU time. The results have been presented through diagrams and tables which illustrate that the proposed PSO has better performance in majority of examples.

The presented problem in this paper has been considered the earliness/tardiness costs in a two-stage assembly flowshop with a batch delivery system. As guidelines for future research, the sequence depended setup time can be considered between jobs. In other words, preparing a machine for a job takes time that is related to the obvious job that has been processed on the machine. On the other hand, in the real world, jobs might not be available at the beginning; consequently, jobs can be contemplated with release time.Theseconsiderations are prevalent in many industries with two-stage flowshop scheduling system; and thus, are valuable for further research.

Reference

Al-Anzi, FS and Allahverdi, A, (2007). A self-adaptive differential evolution heuristic for two- stage assembly scheduling problem to minimize maximum lateness with setup times. *European Journal of Operational Research*, 182:80–94.

Allahverdi, A. and Al-Anzi, FS. (2007). the two-stage assembly flowshop scheduling problem with bicriteria of makespan and mean completion time. *International Journal of Advanced Manufacturing Technology*, 37 (1): 166–177.

Goldberg, D. (1989). Genetic algorithms in search: Optimization and machine learning. Reading, MA: Addison-Wesley.

Haq, A.N. and Kannan, G. (2006). Effect of forecasting on the multi-echelon distribution inventory supply chain cost using neural network, genetic algorithm and particle swarm optimization. International Journal of Services Operations and Informatics, 1(1-2), 1–22.

Hariri, A. M. A., and C. N. Potts. (1997). A Branch and Bound Algorithm for the Two-Stage Assembly Scheduling Problem. *European Journal of Operational Research*, 103: 547–556.

Haouari, M., and Daouas, T, (1999). Optimal scheduling of the 3-machine assembly-type flow shop. *RAIRO Operation Research*, 33(4):439–445.

Holland, J.M. (1975). Adaption in natural and artificial systems. Ann Arbor, MI: The University of Michigan.

Ju-Yong Lee and June-Young Bang. (2016). A Two-Stage Assembly-Type Flowshop Scheduling Problem for Minimizing Total Tardiness. Mathematical Problems in Engineering, Article ID 6409321, 10 pages.

Kazemi, H., Mazdeh, M. M., & Rostami, M. (2017). The two stage assembly flow-shop scheduling problem with batching and delivery. *Engineering Applications of Artificial Intelligence*, *63*, 98-107.

Kennedy, J. and Eberhart, R. (1995). Particle swarm optimization. Proceedings of the 1995 IEEE International Conference on Neural Network, IV, 4, 1942–1948.

Lee, C-Y., Cheng, T.C.E., and Lin, B.M.T., (1993). Minimizing the makespan in the 3-machineassembly-type flowshop scheduling problem. *Management Science*, 39 (5): 616–625.

Mozdgir, A., FatemiGhomi, S.M.T., Joli, F., and Navaei, J., (2013). Two-stage assembly flow-shop scheduling problem with non-identical assembly machines considering setup times. *International Journal of Production Research,* 51:3625-3642.

Potts, C.N., Sevast'janov, S.V.,VanWassenhove, L.N., and Zwaneveld, C.M., (1995). The two-stage assembly scheduling problem: complexity and approximation. *Operations Research,* 43 (2): 346–355.

Seidgar, H., Kiani, M., Abedi, M., and Fazlollahtabar, H. (2014). An efficient imperialist competitive algorithm for scheduling in the two-stage assembly flow shop problem. *International Journal of Production Research,* 52(4):1240-1256.

Seidgar, H., Zandieh, M., Fazlollahtabar, H. and Mahdavi, I., (2015). Simulated imperialist competitive algorithm in two-stage assembly flow shop with machine breakdowns and preventive maintenance. Proceedings of the Institution of Mechanical Engineers, Part B: Journal of Engineering Manufacture, doi: 10.1177/0954405414563554

Sha, D.Y. and Hsu, C.Y. (2008). A new particle swarm optimization for the open shop scheduling problem. Computers & Operations Research, 35(10), 3243-3261.

Shokrollahpour, E., Zandieh, M., and Dorri, Behrooz, (2011). A novel imperialist competitive algorithm for bi-criteria scheduling of the assembly flow shop problem. *International Journal of Production Research*, 49(11):3087-3103.

Sun, X., Morizawa, K.andNagasawa, H, (2003). Powerful heuristics to minimize makespan in fixed, 3-machine, assembly-type flowshop scheduling. *European Journal of Operational Research,* 146(3):498–516.

Sung, C.S., and Juhn, J., (2009). Makespan minimization for a 2-stage assembly scheduling problem subject to component available time constraint. *International Journal of Production Economics,* 119:392-401.

Sung, C.S., and Kim, H. A., (2008). A two-stage multiple-machine assembly scheduling problem for minimizing sum of completion times. *International Journal of Production Economics,* 113: 1038-1048.

Torabzadeh, E., and Zandieh,M.,(2010). Cloud theory-based simulated annealing approach for scheduling in the two-stage assembly flow shop. *Advances in Engineering Software*.41:1238-1243.

Tozkapan, A., Kirca, O., and Chung, C.-S., (2003). A branch and bound algorithm to minimise the total weighted flow time for the two-stage assembly scheduling problem. *Computers and Operations Research*, 30 (2): 309–320.

Umarani, R. and Selvi, V. (2010). Particle Swarm Optimization-Evolution, Overview and Applications. International Journal of Engineering Science and Technology, 2(7), 2802-2806.

Xiong, F., and Xing, K., (2014). Meta-heuristics for the distributed two-stage assembly scheduling problem with bi-criteria of makespan and mean completion time, *International Journal of Production Research*, 52(9):2743-2766.