International Journal of Mechanical, Industrial and Aerospace Sciences
ISSN: 2517-9950
Vol:6, No:2, 2012

A Particle Swarm Optimization Approach for the Earliness-Tardiness No-Wait Flowshop Scheduling Problem

Sedighe Arabameri, Nasser Salmasi

***Abstract*—**In this researcha particle swarm optimization (PSO) algorithm is proposedfor no-wait flowshopsequence dependent setuptime scheduling problem with weighted earliness-tardiness penalties as the criterion ($F_m|nwt, S_{ijk}|\sum w_j'E_j + w_j^{"}T_j$).The smallestposition value (SPV) rule is applied to convert the continuous value of position vector of particles in PSO to job permutations.A timing algorithm is generated to find the optimal schedule and calculate the objective function value of a given sequence in PSO algorithm. Twodifferent neighborhood structures are applied to improve the solution quality of PSO algorithm.The first one is based on variable neighborhood search (VNS) and the second one is a simple one with invariable structure. In order to compare the performance of two neighborhood structures, random test problems are generated and solved by both neighborhood approaches.Computational results show that the VNS algorithmhas better performance than the other one especially for the large sized problems.

***Keywords*—**minimization of summation of weighed earliness and tardiness, no-waitflowshop scheduling, particle swarm optimization, sequence dependent setup times

I. Introduction

IN this research, a no-wait flowshop scheduling problem(NWFSP) has been investigated. Hall and Sriskandarajah[1] mentioned severalapplications of no-waitscheduling problems in different industries such as steel, plastic modeling, silverproduction, chemical, and pharmaceuticalindustry. In a NWFSP, it is assumed that *n* jobs areprocessed on a flowshop with*m* machineswithout interruption on a machine or between machines.In other words, when the process of a job starts on the first machine,its process should not be interrupted until its process on the last machine is completed without waiting in the line of any machine.

SedigheArabameri is with the Department of Industrial Engineering,Sharif University of Technology, Tehran, Iran. e-mail: s_arabameri@ie.sharif.edu).

Nasser Salmasi is an associate professor in the Department of Industrial Engineering at Sharif University of Technology, P.O. Box 11155-8639, Tehran, Iran. e-mail:nsalmasi@sharif.edu).

It is assumed that the setup time of each job on each machine depends on the previous processed job on the machine. The goal is to find the best sequence of processing jobs on machines in order to minimize the summation of the weightedearliness and tardiness. The research problem is notedas$F_m|nwt, S_{ijk}|\sum w_j'E_j + w_j^{"}T_j$ based on Pinedo[2]. NWFSPwith makespan criterion is proved to be NP-hard by Rock[3].Thus, our proposed research problem is NP-hard too since it deals with a more complex objective function as well as considering sequence dependent setup times for jobs on each machine.

Therefore, heuristic and metaheuristic algorithm sare needed to solve industry sized problems. Hall and Sriskandarajah[1] provide a review of all research performed in no-waitscheduling problems before 1996. They study the computational complexity as well as available heuristic algorithms for no-wait and blocking scheduling problems. Gangadharan and Rajendran [4] and Rajendran [5] developtwo heuristic algorithms to solve NWFSP with makespan criterion and showthat their heuristicsoutperform than existing heuristic algorithms in the literature.Dileepan [6] consider two-machine NWFSP with maximum lateness as criterion and present several theoretical results for the proposed research problem. Wang and Cheng [7] study the two-machine NWFSPwith batch setups and develop a heuristic algorithm to minimize maximum lateness as criterion. Allahverdi and Aldowaisan [8]consider NWFSP with weighted sum of makespan and maximum lateness criterion. They propose a hybrid simulated annealing algorithm and also a hybrid genetic algorithmfor the proposed research problem. They also develop a lower bound for the case of the two-machine problem and use that in a branch and bound algorithm. Wang et al. [9] apply an accelerated tabu search algorithm with three different neighborhood strategies to solve NWFSP with maximum lateness criterion. Pan et al. [10] present a novel discrete differential evolution(DDE) algorithm for solving NWFSP with makespan and maximum tardiness criteria. They develop a local search algorithm to incorporate into the DDE algorithm to balance global and local exploitation.In recent years a significant interest has been arisen in applyingparticle swarm optimization (PSO) algorithm in scheduling problems. Liu et

al. [11] and Pan et al. [12] examine the performance of PSO in NWFSP with makespan as objective function.Pan et al. [13] propose a discrete PSO algorithm to solve the NWFSP with both makespan and total flow time criteriasimultaneously. They hybridize discrete PSO with variable neighborhood descent (VND) algorithm to improve the solution quality. They also propose several speed-up methods for neighborhood structures. Pan et al. [14] present a novel multi-objective PSO algorithm for solving NWFSP with makespan and maximum tardiness criteria at the same time. Tasgetiren et al. [15] develop a PSO algorithm for the single machine total weighted tardiness scheduling problem. They use the smallest position value (SPV) rule, a non-decreasing order mechanism, to convert a position vector of a particle to a job permutation. With the same approach, Tasgetiren et al. [16] solve the permutation flowshop problem with makespan and maximum lateness minimization criteria. They hybridize a local search algorithm based on variable neighborhood search (VNS) with PSO algorithm and show that VNS improves the performance of the PSO algorithm for the proposed research problem. To the best of our knowledge there is no research in NWFSP with minimization of total weighted earliness and tardiness as objective function. This is our motivation to apply PSO for no-wait flowshop scheduling problems with minimization of total weighted earliness and tardiness as objective function.

In this research we apply SPV method to convert continuous PSO to discrete PSO. We also develop a VNS algorithm based on Tasgetiren et al. [16] to improve the results of PSO algorithm for the proposed research problem.The notationsused in this research are as the followings:

n: the number of jobs should be processed

m: the number of machines in the flowshop cell

P_{ij}: the process time of job$j$ on machine i

S_{ijk}: the setuptimeof job k onmachine$i$ifjob j is the immediatelyprecedingjob(sequencedependentsetuptime)

d_j: the due date of job j

w_j': the earliness penalty of job j for each time unit of earliness

$w_j^{"}$: the tardiness penaltyof job j for each time unit of tardiness

The goal is to determine the best sequence of processing the jobs on machinesto minimize theweighted earliness-tardiness penalties.

II. Particle Swarm Optimization

PSO is a population based optimization algorithmwhich is based on metaphor of social interaction and communication such as bird flocking and fish schooling (Pan et al. [12]). Eberhart and Keneddy[17] introduce thismetaheuristicalgorithm for the first time to optimize various continuousnonlinear functions.We apply this metaheuristic algorithm to solve the proposed research problem.

PSO is an iterative algorithm starts with a number of initial solutions, known as particles. The number of initial particles is called *p-size*. Each particle is presented by two n dimensional factors as:position and velocity. Let $X_i^t = [x_{i1}^t, x_{i2}^t, \dots x_{in}^t]$ denotes the position of the i^{th}particle in thet^{th}iterationwherex_{ij}^trepresents the j^{th}dimension of the n-dimensional position vector and$V_i^t = [v_{i1}^t, v_{i2}^t, \dots v_{in}^t]$ denotesthe velocity of the i^{th}particle at the t^{th}iterationwherev_{ij}^trepresent the j^{th} dimension of the n-dimensionalvelocityvector.In this research, the dimension of search space i.e., n, is the number of jobs. Allparticles move through the n-dimensional searching space by learning from themovement of swarm population. For this reason, particles move toward areas with better objective function values.The position with the best objective function value observed ever by each particle is presented by *p-best*. The best position observed ever by all particles is called *g-best*. For the i^{th}particle in the t^{th}iteration, these parameters are presented by $P_i^t = [p_{i1}^t, p_{i2}^t, \dots p_{in}^t]$and $G^t = [g_i^t, g_i^t, \dots g_i^t]$, respectively. Since particles move toward better positions during searching process, the velocity of each particle changes based on the values of *p-best* and *g-best* vectors in each iteration. The range of variation of the velocity vector members should be in a predefined range which is determined with a parameter calledV_{max}. In this research based on extensive experiments V_{max} is chosen equal to 4 and thus, the range of the velocity vector membersshould be in $[-4, 4]$ interval. The velocity of thei^{th} particle in the t+1th iterationis updated using the previous velocity (V_i^t) and the previous position (X_i^t) as following:

$$V_i^{t+1} = w * V_i^t + c_1 * r_1 * (P_i^t - X_i^t) + c_2 * r_2 * (G^t - X_i^t) \quad (1)$$

Where w is the inertia weight which controls the impact of the velocity in the t^{th} iteration in calculating the velocity in the (t+1)th iteration for the i^{th} particle. Moreover c_1 and c_2 are constants called acceleration coefficients. $r_1$andr_2 are random numbers generated uniformly between [0,1].The position of the i^{th} particle at the (t+1)th iteration is updated based on (2).

$$X_i^{t+1} = X_i^t + V_i^{t+1} \quad (2)$$

A. Initial population

The number of initial population is presented by *p-size*.Several efficient rules to generate the initial population in PSO algorithm are applied in this research.The first two particles are generated based on earliest due date (EDD) and Longest Tardiness/Earliness Rate (LTER) rules. According to EDD rule, the jobs are ordered in increasing order of d_j and according to LTER rule, the jobs are ordered in decreasing order of $w_j^{"}/w_j'$. 2+0.1*p-*size*particles are generated by assigning jobs with higher tardiness/earliness ratesto the first slots and the jobs with lower rates to the last slots. The rest of initial particles are generated randomly.

B. Conversion continuous positions to a sequence of jobs

The SPV rule is appliedto find the sequence of jobs ofa particle at each iteration. The position vector of each particle is an n dimensional vector.Each element of the vector is related to a job.To determine this relation based on SPV rule, all members of the position vector are sorted from the smallest to the largest value. For instance, assume that in the t^{th} iteration the position vector of the i^{th} particle is X_i^t = [0.25, 0.08, 0.92, 0.53, 0.32]. In this case the sequence of processing

jobs isJ_2-J_1-J_5-J_4-$J_3$for this particle by sorting the jobs based on their position values.

C. Timing algorithm

The objective function value of each particle at each iteration is used to update the values of *p-best* and *g-best*. Since the objective function of the research is not a regular one, finding the optimal schedule of processing the jobs even for a given sequence is not an easy task. Thus, a timing algorithmis proposed to find the optimal schedule as well as the objective function value of each sequence generated at each iteration. The idea is based on Szwarc and Mukhopadhyay [18]. They propose an exact algorithm to schedule all jobs in a given sequence for a single machine scheduling problem to minimize total weighted earliness and tardiness penalties. We generalize this algorithm to our proposed research problem. Assume that a sequence of jobs {1, 2, …,*n*} is given. As the first step,the jobs are scheduled with no delay from the beginning of the planning horizon with respect to no-wait property. This schedule is called as the *initial schedule*. It is clearthat this schedule provides the earliest time that a job can be processed in the given sequence. In the*initial schedule*, there might exist idle timesbetween processing two adjacent jobs on machines because of the no-wait property. As the second step, the jobs are grouped as clusters. The clustering of jobs is based on a rule guarantees that in the optimal solution among the process time of jobs belong to a cluster, it does not exist any idle time, except the idle times needed to satisfy the no-wait property.Then, the start time of processing of these clusters is shifted forward in several iterations in order to improve the objective function value of the sequence. At each iteration, a number of clusters are selected to shift forward. This shift makes all jobs in these clusters to be processedlater than the initial schedule. This shift causesadditional idle timeson all machines. These imposed idle timesare called *extra idle times*. Assume that C_j^e is the completion time of the j^{th} job in the *initial schedule*. The following lemma which is a generalization of a lemma proposed by Szwarc and Mukhopadhyay [18] is usedto describe the timing algorithm.

Lemma (1): If $d_{j+1} - d_j \leq C_{j+1}^e - C_j^e$, then there is no *extra idle time* between processing jobs j and $j + 1$ on all machines in the optimal schedule. In other words, job j+1 is processed at the earliest possible time after job j by considering no-wait constrainton all machines.

Proof: The right hand side of the inequality, i.e., $C_{j+1}^e - C_j^e$is theminimum possible difference between the completion time of two adjacent jobs j and j+1 at the optimal schedule. We show that in all possible cases, this minimum difference is kept at the optimal schedule for each two adjacent jobs that the inequality stated in lemma (1) holds. The validity of the lemma is discussed in all possible cases. Assume that C_j denotes the completion time of the j^{th} job in the optimal schedule.

Case 1: job j is completed early ($d_j > C_j$). Assume that job j+1 is early too i.e., $d_{j+1} > C_{j+1}$. In this case if job j+1 is shifted forward to make an *extra idle time* between processing the jobs, the objective function is reduced by shifting job j to the right and eliminate the idle time between processing two jobs. Assume that job j+1 is tardy i.e., $d_{j+1} < C_{j+1}$, then shifting either job j or job j+1 backward or forward to make an *extra idle time*, increases the objective function value. If $d_{j+1} = C_{j+1}$, then shifting job j backward or job j+1 forward to make an *extra idle time* increasesthe objective function value.

Case 2: job j is completed at its due date ($d_j = C_j$). If $d_{j+1} > C_{j+1}$, the inequality of lemma (1) does not satisfy. Thus, the only situation that lemma (1) can be used is the case in which job j+1 is late or on time i.e., $d_{j+1} \leq C_{j+1}$. If job j+1 is shifted to the right or job j is shifted to the left and make an *extra idle time* between processing the two jobs,in both cases the objective function value is increased.

Case 3: job j is completed after its due date ($d_j < C_j$). If job j+1 is early or on time ($d_{j+1} \geq C_{j+1}$), the inequality of lemma (1)is not satisfied. The inequality of lemma (1) can be valid ifjob j+1 is tardy i.e., $d_{j+1} < C_{j+1}$. If job j+1 is shifted to the right, the objective function is increased by increasing tardiness of job j+1. If job j is shifted to the left and make an *extra idle time* between processing the two jobs, job j+1 can be shifted to the left to reduce the objective function value.

A sequence of jobs such as$u$,…,v is called a *jobcluster* if for each pair of adjacent jobs j and j+1, lemma (1) holds and for job j=u-1 and job j=$v$the lemma (1) does not hold. Therefore, according to lemma (1) all jobs in a cluster should be processed without any *extra idle time*.

The relation between the earliness of two early jobs orthe tardiness of two tardy jobs in a cluster isdefined based on a lemma from Szwarc and Mukhopadhyay [18] as the following:

Lemma (2):*In a job cluster, the early jobs precede the tardy jobs. Moreover, if jobs j and* j+1 *both are early* $E_j \geq E_{j+1}$ *and if both are tardy* $T_j \leq T_{j+1}$.

Each sequence of processing jobs can be decomposed into a set of l clusters such as $r_1, r_2, \dots, r_l$. It is clear that the completion time of all jobs in a cluster increase by shifting the process of the cluster to the right. The goal is to determine the length of time that each cluster should be shifted to the right (compared to the schedule provided in the *initial schedule*)in order to find the optimal schedule for a given sequence. Consider a cluster that consists of a couple of jobs.The jobs inthe cluster may be early, on time, or tardy. According to lemma (2), the early jobs precede the tardy jobs.Assume that$j_f$is the last job that is early in the clusteri.e., the job with the smallest earliness in the cluster. Thus, all jobs before job j_f, if there exist any, are early and the earliness of those jobsare more than the earliness of job j_f; and all jobs after job j_f, if there exist any, are on time or tardy. Consider a cluster that consists of jobs k,…,h. Following notations are needed in the proposed method:

$$\Delta_j = \sum_{l=k}^{j} w_l' - \sum_{l=j+1}^{h} w_l^{"} \quad j = k, \dots, h \tag{3}$$

Where Δ_j is calculated for every job belonging to a cluster. It is clear that the value of Δ_j is fixed for each job during all iterations. According to timing algorithm, at each iteration, a set of clusters are determined to shift forward. A set of consecutive clusters such as $r_s, r_{s+1}, \dots, r_l$ are called a block if they are chosen to be shifted to the right with each other inan iteration. Let:

$$E(f) = E_{j_f} = d_{j_f} - C_{j_f} \tag{4}$$

$$\Delta(f) = \Delta_{j_f} = max_{k \le j \le j_f} \Delta_j \tag{5}$$

$E(f)$presentsthe earliness of the last early job in the f^{th}cluster. Itrepresents the maximum acceptable shifting unit of time for the f^{th}cluster belonging toa block that guarantees the improvement of objective function value.Based on this definition$min\big(E(s), \dots, E(l)\big)$ represents the appropriate shifting unit of time for all clusters of the block since it is promising for all clusters.$\Delta(f)$presents the maximum value of $\Delta_j$among the early jobs.$\Delta(f)$is used to calculate the value of decreasing objective function at an iteration. If none of the jobs in the f^{th} cluster is early then $E(f)$ and $\Delta(f)$are replaced by ∞ and $-\sum_{l=k}^{h} w_l^{"}$, respectively. If at least one early job exists at each cluster of a block, then a shift of the entire block by one time unit reduces the total cost by $\sum_{f=s}^{l} \Delta_f$.

The timing algorithm proposed for the research problem is an iterative algorithm which identifies a block of clusters at each iteration to be shifted. This block is shifted with the length of the smallest $E(f)$ of all clusters in the block. The algorithm is stopped if no such block is found.

The timing algorithm can be summarized as follows:

Step 1.

Schedule all jobs in the earliest possible time. Call this schedule as the *initial schedule*. Let C_j^e be the completion time of job *j*in the *initial schedule*. Set $C_j = C_j^e$ for all jobs. Create the clusters based on lemma (1) and compute $\Delta(f)$ for each cluster.

Step 2.

Find the smallest *s* such that $\sum_{f=1}^{s} \Delta(f) \le 0$.

Assign$C_j$for each job *j* in the first *s* clusters.

If $s = l$ then STOP, otherwise, go to Step 3.

If no such s exists, then go to Step 4.

Step 3.

Remove the first s clusters from the list.

Reindex all remaining clusters and jobs.

Go to Step 2 to consider the set of remaining clusters.

Step 4.

Find $min(E(1), \dots, E(l))$.

Add$min(E(1), \dots, E(l))$ to all C_j

Eliminate all early jobs that are no longer early.

Update $E(f)$and $\Delta(f)$.

Go to Step 2.

III. Neighborhood Search Approach

In this research two different neighborhood search approaches are applied. The first one is based on VNS algorithmproposed by Tasgetiren et al. [16] and the second one is based on a simple neighborhood structure called *inset neighborhood*. In both neighborhood search approaches, at each iteration, if the objective function value of the new position of a particle has a chance to enhance the value of *p-best*s or *g-best*, a neighborhood search is performed around the new position to find better positions. If the neighborhood search finds better positions, the better one is considered as the new position of the particles and *p-best*s or *g-best* are updated based on the new position.After updating all *p-best*s by neighborhoodsearch approach and identifying the new *g-best*, a neighborhood search is performed around the new *g-best* to find a better one.Tasgetiren et al. [16] hybridize PSO with VNS to enhance the PSO algorithm performance to solve the permutation flowshopscheduling problem to minimize both makespan and maximum lateness criteria. Our suggested neighborhood search methods in VNS algorithm are as follows:

1) Remove job in thek^{th} position and insert it to the h^{th} position (insert(*k*,*h*)).

2) Swap two jobs between thek^{th} and theh^{th} positions (swap(*k*,*h*)).

3) Interchange two adjacent jobs in thek^{th} and the$k+1^{th}$ positions.(sub_interchange(*k*,*k+1*))

In VNS algorithm all three structures are used. The VNS algorithm is the customized version of the one proposed by Tasgetiren et al. [16]. There are two major differences between our proposed VNS and the one proposed byTasgetiren et al. [16]. The first difference is applying VNS for *p-best*s rather than *g-best*at each iteration. The second one is adding sub_interchange structure to VNS algorithm. In the proposed method an insert (swap) moveis performed around the permutation as the first step which is either *p-best*s or *g-best*. Ifthe objective function value of the new generated solution is better than the original one, the insert (swap) moveis continued around the new improved solution.Otherwise,the other neighborhood generated structure, i.e., swap (insert) moveis performed on the best found solution so far. Moreover, after each improvement by insert (swap) move, sub_iterchangeis performed to find better solutions.

The second neighborhood structure is called *insert neighborhood*. In this approach,at each iteration, a job is randomly selected and is removed from its current position and then inserted to another place in the sequence randomly.The details of two proposed neighborhood search approachesare shown in Appendix A.

The number of iterations for both neighborhood structures is called *max-iter*. The value of *max-iter*is determined based on extensive experiments.

IV. Design of Experiments for Parameter Setting

Thevalues of parameters related to the PSO algorithmare determined by experimental design techniques. Test problem instances are generated randomly in different sizes from small, medium, and large size. The problems with at most 20jobs are categorized as small sized problems. Medium sizedproblems are the ones with21 to 60jobs and large sizedproblems are those with 61to 100 jobs. In this research, problems with two, three, and six machine are considered. These problems are generated based on Salmasi et al. [19] suggestions about generating test problems for flowshop scheduling problems. In the interest of time, we perform the Taguchi method (Ross [20])to identify the appropriatefactors levels.InTaguchi method a set of orthogonal array is developed. These trials are a subset of full factorial design trials which reflect full required information. Taguchi defines two major sets of factors, controllable and uncontrollable noise factors. Thefactors used to generate test problems i.e., the number of jobs and the number of machines in a problem instance are considered as a noise factor with3×3levels(three levelsfor the number of jobs and three levelsfor the number of machines).The goal is to find the best levels of controllable factors in PSO algorithm with both neighborhood search approaches.For each developed PSO algorithm, i.e., $PSO_{VNS}$and PSO_{insert}three controllable factors exist. These factors are presented in Table I and Table II, respectively. The goal is to find the best levels for these factors. As shown in Table I, all controllable factors in $PSO_{VNS}$are defined in three levels.

TABLE I
FACTORS AND THEIR LEVELS IN PSO_{VNS}

Level	factors		
	Neighborhood Structure(A)	Inertia Weight(B)	p-size(C)
1	Insert+Swap(0.1)	0.7298	20
2	Insert+Swap(0.5)	0.4-0.9	30
3	Swap+Insert(0.1)	0.4-1.2	50

The first factorwhich is called as *Neighborhood structure*(A) represents the order of the first two neighborhood move and the maximum number of iterationsfor theVNS algorithm.In the first level of factor A (Insert+Swap(0.1)) as the first step, the insert (*k*,*h*) move is performed. If this move fails to provide solutions with better objective function value, the swap (*k*, *h*) move is performed. In this level the number of iterations is set to *max-iter*=0.1**n*.The second level of factor A is similar to the first one with *max-iter*= 0.5**n*.The structure of the third level is vice versa. As the first step, theswap (*k*,*h*)move is performed. If this move fails to provide a solution with better objective function value,theinsert (*k*,*h*)move is performed. In this level the number of iterations is set to *max-iter*=0.1**n*.

The second factor is called*inertia weight*. At the first level, the value of *w* is considered as a fixed number i.e., 0.7298 in all iterations. In other words, thisconstant is the coefficient of all components in equation (1). This factor is helpful in convergence of the PSO algorithm according to the following equations:(Poli et al. [21])

$$\varphi = \frac{2}{C - 2 + \sqrt{C^2 - 4C}}, C = c_1 + c_2 > 4 \quad (6)$$

$$V_i^{t+1} = \varphi * [V_i^t + c_1 * r_1 * (P_i^t - X_i^t) + c_2 * r_2 * (G^t - X_i^t)] \quad (7)$$

We set c_1 and c_2 to 2.05 to satisfy the condition.

$$(C = c_1 + c_2 = 2.05 + 2.05 = 4.1) \rightarrow \varphi = 0.7298$$

The value of *inertia weight* for the other two levels are set as ranges presented in Table II. The formula used to generate the value of *w*at each iterationin these two levels is presented by equation (8). In this formula, w_{max} and w_{min} are the highest and the lowest values in the range, respectively. For instance, in the second level, these parameters get the values 0.9 and 0.4, respectively. *Iteration*represents the number of current iteration and *max_iteration*represents the maximum number of iterations.

$$w = w_{max} - \frac{(w_{max} - w_{min}) * iteration}{\max\ \_iteration} \quad (8)$$

It is clear that by increasing the number of iterations, the effects of *g-best* is increased compared to the effect of *p-best* in finding new position for each particle since the value of *w* is increased by performing more iterations.

The third factor(C)indicatesthe size of the initial population(*p-size*) which is defined at three levels 20,30, and 50.

The controllable factors of PSO_{insert}algorithm and relevant levels are presented in Table II. The first factor which is called the*search strength* represents the number of iterations ofPSO_{insert}algorithm. The number of iterations is set to *max-iter*=0.25**n* for the first level.The second and the third levels of factor A are similar to the first one with *max-iter*= 0.50**n* and *max-iter=n*, respectively.The last two factors of PSO_{insert}algorithm are similar tothe last two factors of PSO_{VNS}algorithm.

TABLE II
FACTORS AND THEIR LEVELS IN PSO_{INSERT}

Level	factors		
	Search strength (A)	Inertia Weight(B)	p-size(C)
1	0.25	0.7298	20
2	0.50	0.4-0.9	30
3	1.00	0.4-1.2	50

The stopping criteria in Taguchi methodisconsidered as the time spend to solve the problem. In this research, the time spend to solve each problem instance is set to 20,60, and 180 seconds for small,medium and large sizedproblems respectively. $c_1$andc_2 are set to 2 in cases which do not need to satisfy condition $c_1 + c_2 > 4$, i.e., when the value ofw is considered as 0.7298 according to equation (6).There are 3×3 classes (three classes for the number of jobs and three classes for the number of machines) for all problems.If two probleminstances are generated for each class randomly; thus, 18 instances should be generated.Since the position and the velocity vectors are generated randomly at each run and the solution of the problem may be different at each run, we run

each of 18 instances two times to gain better result.If we apply full factorial design we need to perform 27 treatments (three controllable factors with three levels for each). So 18×2×27=972 instances should be solved totally.Taguchi suggest orthogonal array L_9 for an experiment with three factors each of them in three levels. ArrayL_9 is given in Table III. Taguchi recommends analyzing variation using signal to noise ratio(S/N).Since the goal is to minimize the objective function value, the appropriate S/N ratio formula is suggested as equation (9):

$$S/N = -10 * \log_{10}\left(\frac{1}{n}\sum_{i=1}^{n} y_i^2\right) \quad (9)$$

This ratio indicates the amount of variation in the response variable sincethe signal denotes the desirable value and noise denotes the undesirable value(standard deviation).

TABLE III
ORTHOGONAL ARRAY L9 DESIGN

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

Fig.1 and Fig.2 illustrate the results of Taguchi method. As shown in Fig. 1 and Fig.2, level 1 for all factors in $PSO_{VNS}$and level 2 for all factors in PSO_{insert}identify the best level. The result of Taguchi method is summarized in Table IV and Table V.

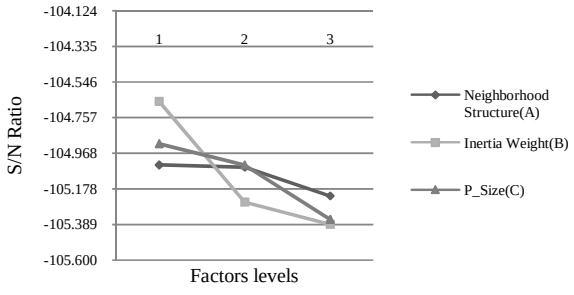

Fig. 1The S/N ratio of parameters in PSO_{VNS}

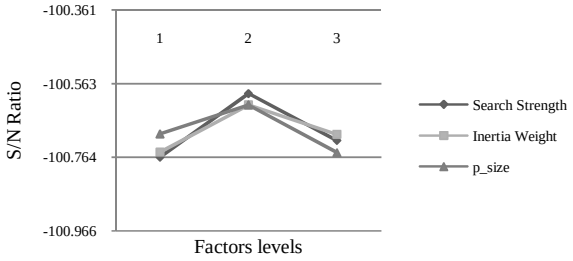

Fig. 2 The S/N ratio of parameters in PSO_{insert}

TABLE IV
THE BEST LEVEL FOR ALL FACTORS IN PSO_{VNS}

	Factors		
	Neighborhood Structure(A)	Inertia Weight(B)	p-size(C)
best level	Insert+Swap(0.1)	0.7298	20

TABLE V
THE BEST LEVEL FOR ALL FACTORS IN PSO_{INSERT}

	Factors		
	Search Strength(A)	Inertia Weight(B)	p-size(C)
best level	0.50	0.4-0.9	30

V. TEST PROBLEM SPECIFICATION

Based on Salmasi et al. [19] theratioofsetup times of jobs on *consecutive machines*is an important factor in generating test problems for flowshop scheduling problems.They consider three different levels for this factor. In a sequential machine pair if the setup time of jobs in the first machine is less than the setup time of jobs in the second one, the ratio of setup times belongs to the first level. If the setup time of jobs in the first machine is the same as the setup time of jobs in the second one, the ratio of setup times belongs to the second level and if the setup time of jobs in the first machine is larger than the setup time of jobs in the second machine,the ratio of setup timesbelongs to the third level. These levels are shown in Table VI-VIII for two, three, and six machine problems,respectively.Thus,there are three andnine different levels for two and three-machine problems, respectively. For six-machine problems since the number of levels is increased, Salmasi et al. [19] suggest applying one factor for the ratio of setup times for all consecutive machines in the interest of time. All setup times in sequential machine pairsare considered at the same level. Thus, for two-machine problems there are 3×3 different levels (three levels for the number of jobs and three levels for the number of setup time ratio). For three machine problems there are 3×9 different levels(three levels for the number of jobs and nine levels for the number of setup time ratio) and3×3 different levelsfor six-machine problems (three levels for the number of jobs and three levels for the number of setup ratio). Three problem instances are generated for each level of two, three, and six-machine problems.

TABLE VI
THE SETUP TIME OF EACH MACHINE ON TWO-MACHINE PROBLEMS

Machine	Level 1	Level 2	Level 3
M1	[1,50]	[1,50]	[17,67]
M2	[17,67]	[1,50]	[1,50]

TABLE VII
THE SETUP TIME OF EACH MACHINE ON THREE-MACHINE PROBLEMS

Machine	Level 1	Level 2	Level 3
M1	[1,50]	[1,50]	[45,95]
M2	[17,67]	[1,50]	[17,67]
M3	[45,95]	[1,50]	[1,50]

TABLE VIII
THE SETUP TIME OF EACH MACHINE ON SIX-MACHINE PROBLEMS

Machine	Level 1	Level 2	Level 3
M_1	[1,50]	[1,50]	[300,350]
M_2	[17,67]	[1,50]	[170,220]
M_3	[45,95]	[1,50]	[92,142]
M_4	[92,142]	[1,50]	[45,95]
M_5	[170,220]	[1,50]	[17,67]
M_6	[300,350]	[1,50]	[1,50]

The process time of jobs on machines are generated from uniform distribution in the interval of [1,20].The earliness penalties for earliness unit and the tardiness penalties for tardiness unit are generated from [1,30] uniformly.The due dates are generated as follows:

$[LB\left(1-T-\frac{R}{2}\right), LB(1-T+\frac{R}{2})]$; Where$LB$ is an approximation of the earliest possible completion time of the last job. T and $R$are selected from the set{0.2,0.5,0.8} randomly.Since the combinations of{0.8,0.5} and {0.8,0.8} provide negative values for due dates, these combinations are ignored.

The stopping criteria for the problems are defined as follows:

- The maximum number of iterationsis set to 1000, 2000, and 5000 for small, medium and large sized problems, respectively.
- The maximum number of iterations without improvement is set to 0.7**max-iteration* for all problems.
- The CPU time is set to 600 seconds for all problems.

VI. EXPERIMENTAL RESULTS

Both proposed PSO algorithms were coded in C++ and run on an AMD phenom (tm) 9600 Quad-Core Processor 2.31 GHz PC with 2 GB memory. The performances of the two proposed algorithms are compared as *paired t-test* for two, three, and six machine problems, separately. The results of the experiment with SPSS software are presented in Appendix B (Table XII-XIII). The *p*-value for two-machine problems is equal to 0.358 implying that there is no evidence about existing any difference between the performance of two proposed algorithms in two machine problems. But the performance of these two algorithms is significantly different in three and six-machine problems since the *p*-value for these experiments are almost equal to zero. Since the average objective function values provided by $PSO_{VNS}$are lower than the PSO_{insert} we can conclude that PSO_{VNS} has a better performance than PSO_{insert} for three and six machine problems. The result of the experiments is shown in Table IX-XI. The percentage error is calculated according to the following formula:

$$percentage\ error = \frac{(the\ PSO\ algorithm\ solution - the\ best\ solution)}{the\ best\ solution} * 100$$

TABLE IX
THE AVERAGE PERCENTAGE ERROR FOR TWO-MACHINE PROBLEMS

M1/M2 Ratio	size	Percentage error(%) PSO_{VNS}	Percentage error(%) PSO_{insert}
Level 1	Small	0.4	0.0
	Medium	0.0	0.4
	Large	0.0	0.4
Level 2	Small	0.0	1.1
	Medium	4.4	3.1
	Large	0.0	3.7
Level 3	Small	0.0	4.1
	Medium	1.4	1.6
	Large	0.0	1.0
Average		**0.69**	**1.71**

TABLE X
THE AVERAGE PERCENTAGE ERROR FOR THREE-MACHINE PROBLEMS

M1/M2 Ratio	M2/M3 Ratio	size	Percentage error(%) PSO_{VNS}	Percentage error(%) PSO_{insert}
Level 1	Level 1	Small	0.9	0.0
		Medium	1.1	0.0
		Large	0.0	0.0
	Level 2	Small	0.0	0.0
		Medium	3.6	2.0
		Large	0.0	6.4
	Level 3	Small	0.1	9.8
		Medium	1.0	7.7
		Large	0.8	3.3
Level 2	Level 1	Small	0.0	0.0
		Medium	1.5	0.0
		Large	0.0	0.0
	Level 2	Small	0.0	0.0
		Medium	1.6	2.2
		Large	2.2	4.7
	Level 3	Small	0.0	6.4
		Medium	1.5	4.2
		Large	0.2	1.7
Level 3	Level 1	Small	0.0	0.0
		Medium	0.0	0.2
		Large	0.0	1.1
	Level 2	Small	0.0	1.1
		Medium	0.0	3.1
		Large	0.0	3.5
	Level 3	Small	0.0	3.5
		Medium	1.9	1.4
		Large	0.0	0.1
Average			**0.61**	**2.31**

TABLE XI
THE AVERAGE PERCENTAGE ERROR FOR SIX-MACHINE PROBLEMS

M1/M2 Ratio	size	Percentage error(%)	
		PSO_{VNS}	PSO_{insert}
Level 1	Small	0.0	0.1
	Medium	0.0	0.9
	Large	0.0	1.3
Level 2	Small	0.0	1.2
	Medium	1.4	1.2
	Large	1.9	3.0
Level 3	Small	0.0	6.0
	Medium	0.0	5.2
	Large	0.0	3.0
Average		**0.37**	**2.43**

VII. CONCLUSIONS

In this research we approach the no-wait flowshopsequence dependent setup time scheduling problem with minimization of weighted earliness-tardiness as the objective for the first time.Since the research problem is NP-hard, a metaheuristic algorithm based on PSO algorithm is proposed to solve the research problem. Two different neighborhood approaches called PSO with variable neighborhood search (PSO_{VNS}) and PSO with invariable neighborhood search (PSOinsert) are applied to improve the performance of proposed PSO algorithm. A timing algorithm is customized to the proposed research problem to find the optimal schedule for a given order of jobs in PSO algorithm. Taguchi method is applied to determine the optimal level of parameters in PSO algorithm. Experimental results show that the performance of PSO_{VNS} is better than PSO_{insert} in the problems with three and six machine problems.

Appendix A: The pseudo code of Neighborhood search approach

The pseudo code of VNS algorithm:

$s = permutation\ which\ asked\ to\ search\ around$
$s' = s;$
$r_1 = rand(1,n); r_2 = rand(1,n); r_1 \neq r_2$
$Loop = 0;$
$do\{$
$kcount = 0;$
$max\_method = 2;$
$do\{$
$if(kcount == 0)\ then\ \{s_1 = insert/swap(r_1,r_2) for\ s;\}$
$if\ (f(s_1) \leq f(s'))then\ \{$
$kcount = 0;$
$s' = s_1;$
$for(f_1 = 1; f_1 \leq n-1; f_1++)\{$
$s_1 = sub\_interchange(f_1, f_1+1) for\ s;$
$if\ (f(s_1) \leq f(s'))then\ \{s' = s_1;\}$
$\}$
$\}$
$else\ \{kcount++;\}$
$if(kcount == 1)\ then\ \{\ s_1 = swap/insert(r_1,r_2) for\ s;\}$
$if\ (f(s_1) \leq f(s'))then\ \{$
$kcount = 0;$
$s' = s_1;$
$\}$
$else\ \{kcount++;\}$
$\}while\ (kcount < max\_method)$
$loop++;$
$\}while\ (\ loop\ \leq max - iter);$
$if\ (f(s') \leq f(s))then\{$
$s = s'\}$

The pseudo code of insert neighborhood algorithm:

$s = permutation\ which\ asked\ to\ search\ around$
$s' = s;$
$i = 0;$
$do\{$
$i++;$
$r_1 = rand(1,n); r_2 = rand(1,n); r_1 \neq r_2$
$s_1 = insert(r_1,r_2) for\ s;$
$if\big(f(s_1) \leq f(s')\big)\{$
$s' = s_1;$
$\}$
$\}$
$while(i \leq max - iter)$
$if(f(s') < f(s))$
$s = s'$

Appendix B: The result of paired t-tests for the two PSO algorithms comparison

TABLE XII
PAIRED SAMPLES STATISTICS

	Mean	N	Std. Deviation	Std. Error Mean
PSO_{vns}-2machine	7742.04	27	4332.84	833.86
PSO_{insert}-2machine	7748.96	27	4325.47	832.44
PSO_{vns}-3machine	131248.78	81	98413.35	10934.82
PSOinsert-3machine	134123.23	81	102503.32	11389.26
PSOvns-6machine	2750204.52	27	3420543.50	658283.90
PSOinsert-6machine	2767929.52	27	3422960.28	658749.01

TABLE XIII
PAIRED SAMPLES TEST

	Mean	Std. Deviation	t	df	p-value	95% Confidence Interval of the Difference Lower	Upper
PSOvns2machine - PSOinsert2machine	-6	38	-0.94	26	0.358	-22	8.29
PSOvns3machine - PSOinsert3machine	-2874	6691	-3.87	80	0	-4354	-1394.78
PSOvns6machine- PSOinsert6machine	-17725	20692	-4.45	26	0	-25910	-9539

REFERENCES

[1] N.G.Hall, C. Sriskandarajah,A Survey of Machine Scheduling Problems with Blocking and No-Wait in Process. Operations Research 44 (1996) 510-525.
[2] M. Pinedo,Scheduling theory, algorithms, and systems. 3rd ed., Englewood Cliffs, NJ: Prentice-Hall; 2008, pp. 13-78.
[3] H. Rock, The three-machine no-wait flowshop problem is NP-complete. Journal of the Association for Computing Machinery 31(1984) 336–345.
[4] R. Gangadharan, C. Rajendran, Heuristic algorithms for scheduling in the no-wait flowshop. International Journal of Production Economics 32 (1993) 285–290.
[5] C. Rajendran, A no-wait flowshop scheduling heuristic to minimize makespan. Journal of the Operational Research Society 45 (1994) 472–478.
[6] P. Dileepan, A note on minimizing maximum lateness in a two-machine no-wait flowshop. Computers & Operations Research 31 (2004) 2111–2115.
[7] X. Wang, T.C.E. Cheng, Aheuristic approach for two-machine no-wait flowshopscheduling with due dates and class setups. Computers & Operations Research 33 (2006) 1326–1344.

[8] A. Allahverdi, T. Aldowaisan, No-wait flowshops with bi-criteria of makespanand maximum lateness. European Journal of Operational Research 152 (2004) 132–147.
[9] C. Wang, X. Li and Q. Wang, AcceleratedtabusearchfOr no-wait flowshopschedulingproblem. EuropeanJournalofOperationalResearch206(2010) 64-72.
[10] Q.K. Pan, L. Wang and B. Qian, Anoveldifferentialevolutionalgorithmforbi-criteriano-waitflowshop. Computers&OperationsResearch36 (2009) 2498-2511.
[11] B. Liu, L. Wang and Y.H. Jin, Aneffectivehybridparticleswarmoptimizationforno-waitflowshopscheduling. IntJAdvManufTechnol(2007)31:1001–1011.
[12] Q.K. Pan, L. Wang, Tasgetiren, M.F. Ahybriddiscreteparticleswarmoptimizationalgorithmfortheno-waitflowshopschedulingproblemwithmakespancriterion. IntJAdvManufTechnol(2008)38:337–347.
[13] Q.K. Pan, M.F. Tasgetiren, Y.C. Liang, A discrete particle swarm optimization algorithm for the no-waitflowshop scheduling problem. Computers & Operations Research 35 (2008) 2807 – 2839.
[14] Q.K. Pan, L. Wang, B.Qian, Anovelmulti-objectiveparticleswarmoptimizationalgorithmforno-waitflowshopschedulingproblems. JEM 989-IMechE2008Proc.IMechEVol.222PartB:J.EngineeringManufacture.
[15] M.F. Tasgetiren, M. Sevkli, Y.C. Liang and G. Cencylmaz,Particle Swarm Optimization Algorithm forSingle Machine Total Weighted TardinessProblem. Proceeding of the 2004 congress on evolutionary computation(CEC2004), Portland, 2004; 1412-9.
[16] M.F. Tasgetiren, Y.C. Liang, M. Sevkli and G. Cencylmaz, Particle Swarm Optimization Algorithm for Makespan and Maximum LatenessMinimization in Permutation Flowshop Sequencing Problem.
[17] RC. Eberhart,J. Kennedy, A new optimizer using particle swarm theory. Proceedings of the sixth international symposium on micro machineand human science, Nagoya, Japan; 1995. p. 39–43.
[18] W. Szwarc, S.K. Mukhopadhyay, Optimal Timing Schedules in Earliness-Tardiness Single Machine Sequencing. Naval Research Logistics (1995) 42:1109–1114.
[19] N. Salmasi, R. Logendran and M.R. Skandari,Total flow time minimization in a flowshop sequence dependent group scheduling problem. Computers & Operations Research 37 (2010) 199 -212.
[20] R.J. Ross, Taguchi techniques for quality engineering, McGraw-Hill, New York;1989
[21] R. Poli, K. Kennedy and T. Blackwell, Particle swarm optimizationAn overview. Swarm Intell (2007) 1: 33–57.