Journal of Engineering Science and Technology
Vol. 14, No. 2 (2019) 609 - 628

EVOLUTIONARY HYBRID PARTICLE SWARM OPTIMIZATION ALGORITHM TO MINIMIZE MAKESPAN TOSCHEDULE A FLOW SHOP WITH NO WAIT

LAXMI A. BEWOOR[1,*], V. CHANDRAPRAKASH[1], SAGAR U. SAPKAL[2]

[1]Department of Computer Sci. and Engineering, K.L. University, Guntur, A.P., India
[2]Department of Mechanical Engineering, Shivaji University, Sangli, M.S., India
*Corresponding Author: laxmiabewoor@gmail.com

Abstract

A flow shop with no-wait schedules jobs continuously through all machines without any wait at consecutive machines. This scheduling problem is combinatorial optimization problem and observed as NP-hard as appropriate sequence of jobs for scheduling from all possible combination of sequences is to be determined for reducing total completion time (makespan). This paper presents an effective hybrid Particle Swarm Optimization algorithm for solving no wait flow shop scheduling problem with the objective of minimization of makespan. This Proposed Hybrid Particle Swarm Optimization Makespan (PHPSOM) algorithm represents discrete job permutation by converting the continuous position information values of particles with random key representation rule. The proposed algorithm balances global exploration and local exploitation with evolutionary search guided by the mechanism of PSO, and local search by the mechanism of Simulated Annealing (SA) along with efficient population initialization with Nawaz-Enscore-Ham (NEH) heuristic. The effectiveness of the proposed method is validated by extensive computational experiments based on Reeve's and Taillard's benchmark suite. Computational results and comparisons with best known solutions for makespan confirm that the proposed algorithm's performance is better than the existing methods in terms of searching quality and robustness. Statistical tests of significance validate the improvement in the solution quality.

Keywords: Makespan, No-wait flow shop, Particle swarm optimization, Scheduling, Simulated annealing.

1. Introduction

No-wait flow shop is one of the variant of flowshop in which number of jobs (n) process on number of machines (m) in the same way as in a general flow shop but with an added constraint that there should not be any waiting time between consecutive operations of the jobs. No wait Flow Shop Scheduling Problem (NWFSSP) finds sequence of given jobs on given machines to minimize (or maximize) the given objective function. NWFSSP has wide applications in many industries such as chemical processing to prevent product degradation [1], food processing to ensure food freshness [2], concrete ware production for proper moulding of concrete blocks [3] and pharmaceutical processing for guarantying appropriate chemical reactions [4], etc. NWFSSP addresses various optimality criterions like total completion time (makespan), total flow time (TFT), tardiness, lateness, number of tardy jobs etc. However, the major interest inclined towards minimizing makespan as it determines total processing time of entire pool of jobs [5]. Additionally, makespan is considered as an important performance measure which, when optimized, evaluate the performance of the solutions [6]. Therefore, this paper addresses makespan as an objective function for solving NWFSSP.

The processing time which includes setup times of each job on each machine is given, and for minimization of makespan, a job sequence is to be detected. Exact algorithms can solve only small-sized instances of NWFSSP optimally with reasonable computational time but as the problem size increases these algorithms grows exponentially in their computational time. Many researchers [7-9] concluded that NWFSSP with more than two machines is NP-hard problem because of its computational complexity. Thus, because of strong theoretical and practical significance, NWFSSP has got major attention by many researchers from past few years for the development of an effective and efficient novel approach for solving the NWFSSP. In order to reduce the exponential computational complexity of exact algorithms to polynomial time complexity various efforts are leading towards development of approximate algorithms.

The approximate algorithmic solutions are broadly classified as ‘constructive methods’ and ‘metaheuristic methods’. Generally, heuristic algorithms can obtain near-optimal solution in reasonable time but compromise on quality of solution. Earlier researchers [1, 10-13] had developed efficient constructive heuristic algorithms for minimizing makespan. But, these methods face the problem when there is large sequence of jobs, and continue till all the jobs are incorporated [13] which may not generate efficient solution for large size problem. As the heuristic methods are problem dependent and adopt greedy method, they usually get stuck in local optima and hence unable to obtain global optimal solution. So, the application of metaheuristics for solving real world combinatorial optimization problems is rapidly growing nowadays [14].

Due to advent of computation technique, metaheuristics can solve the problem in less time, so the limitation of computational complexity can be resolved through metaheuristic application. The hybridization of metaheuristics for optimization is explored in recent past; eventually the trend focuses more towards problem specific approaches leading towards hybridization [15]. So, in this work an attempt is made to use metaheuristic technique and their hybridization for solving NWFSSP.

2. Literature Survey

Several noteworthy metaheuristics have been proposed for solving NWFSSP for different objective criterion. Table 1 provides a decade comparative review of various metaheuristics for solving NWFSS for makespan as optimization criteria. The table also summarizes various hybridization techniques for improvement of results obtained through metaheuristics in chronological order.

Table 1. Metaheuristic approaches for solving NWFSSP for makespan.

Year	Author(s)	Acronym	Performance be superior to
2003	Aldowaisan and Allahverdi	SA,SA-1,SA-2,GEN,GEN-1,GEN-2	GA,N-RAJ,RAJ
2005	Grabowskiand Pempera	DS,DS+M,TS,TS+M,TS+MP	RAJ,VNS,GASA
2007	Liu et al.	HPSO	VNS,GASA
2008	Pan et al.	DPSO	HPSO, RAJ, VNS, GASA
2008	Pan et al.	HDPSO	HDPSO,DPSO
2008	Pan et al.	IIG	RAJ, TS, TS+M, TS+MP, DPSO
2009	Qian et al.	HDE	HPSO
2010	Tseng and Lin	HGA	RAJ, VNS, GASA, TS, HPSO
2011	Jarboui et al.	GA-VNS	SA, TS, VNS, DPSO
2012	Samarghandi and ElMekkaw	TS-PSO	VNS, GASA, DS, DS+M, TS, TS+M, TS+MP
2015	Ding et al.	TMIIG	DPSO, IIG, HDE, HGA, GA-VNS, TS-PSO
2015	Lin et al.	$MNEH_1$+LKH $MNEH_2$+LKH	GA-VNS, TMIIG,LC, DPSO, IIGA, HDE, HGA, GA-VNS,TS-PSO, TMIIG

Aldowisen and Allvarhdi [11] proposed improved versions of Simulated Annealing(SA) and Genetic algorithm(GA) and developed six metaheuristics viz. SA, SA-1, SA-2, GA, GA-1, GA-2 with block concept as insertion technique along with NEH [16] for improving the performance of SA and GA. The computational experiments proved that the improved algorithm efficiently finds the solutions over famous RAJ algorithm [1]. The use of metaheuristic for NWFSSP was further attempted by Grabowski and Pempera [12] and proposed several variants of Descending Search (DS) and Tabu Search (TS) algorithms for generating permutations using local search algorithms with multimoves for converging towards good solution. The proposed algorithm also used dynamic tabu to avoid trapping at a local optimum.

Eventually, hybridization of metaheuristics was also attempted. Liu et al. [17] proposed a hybrid algorithmic approach for solving NWFSSP by hybridizing the searching ability of population-based PSO with adaptive local search heuristics to balance local exploration and exploitation. In the proposed method, local search was based by Simulated Annealing (SA) while adaptive local search with meta-Lamarckian learning strategy was incorporated in PSO. Extending the idea of hybridization further, Pan et al. [18] developed Discrete Particle Swarm Optimization (DPSO) by considering both makespan and total flowtime

minimization as optimization criteria for solving the no-wait flowshop scheduling problem. Hybridization of PSO with Variable Neighbourhood Descent (VND) algorithm was proposed for improving the solution quality. Swap and insert neighbourhood structures were used as speed-up methods. Subsequently, Pan et al. [19] proposed an effective hybrid discrete particle swarm optimization (HDPSO) algorithm NWFSSP which stressed the appropriate balance between not only global exploration and also local exploitation neighbourhood searching using insertion based local search technique.

Further, Pan et al. [5] proposed an improved iterated greedy algorithm (IIGA) along with modified NEH (M_NEH). M-NEH was employed for constructing an initial sequence in IIGA. SA type an acceptance criterion was applied to avoid to get trapped in local optima. Qian et al. [20] attempted use of Differential Evolution (DE) in NWFSSP context and proposed an Effective Hybrid Differential Evolution (HDE). The proposed algorithm applied largest-order-value (LOV) rule for converting the continuous values of individuals in DE to job permutations. A fast Insert-based neighbourhood method was used to reduce the computation complexity.

GA as a metaheuristic technique was quite popular for solving optimization problem and it has been observed that the solution quality was much better with hybridization of GA. Tseng and Lin [4] proposed Hybrid genetic algorithm and a local search method with combination of the Insertion Search (IS) and the Insertion Search with Cut-and-Repair (ISCR) as two local search techniques. The genetic algorithm was used for the global search and two local search methods were used for the local search. Further, Jarboui et al. [21] used GA for flowshop sequencing context and proposed hybrid GA (HGA) algorithm. The proposed HGA has Variable Neighbourhood Search (VNS) for local exploration and GA for global exploitation. Samarghandi and ElMekkaw [22] proposed hybrid of Tabu Search (TS) and Particle Swarm Optimisation (PSO) with new coding and decoding technique based on Factoradic number system. The proposed PSO explores solution space by coding technique and moves from one solution to a neighbourhood solution. Afterwards, decoding of the solutions to its respective feasible solutions was returned to the TS. Ding et al. [23] proposed improved iterated greedy (IG) method with Tabu-mechanism (TMIIG) to solve the NWFSSP with a makespan criterion. IG was improved by utilizing Tabu based reconstruction strategy for enhancing exploration ability.

Recently, Lin and Ying [24] modelled NWFSSP as an Asymmetric Travelling Salesman Problem (ATSP), and developed two metaheuristics which comprises of three phases. In the first phase, initial seed sequence was generated with MNEH1 and MNEH2 as constructive heuristics. In the second phase, NWFSSP was converted to ATSP and LKH heuristic was applied to obtain near optimal solution. In the third phase, BIP mathematical model was used for obtaining optimal results. The proposed model was tested on many dataset including very hard and large problem instances and simulation results demonstrated that the proposed metaheuristic outperform over all existing algorithms.

The following insights were obtained from this concise literature review:

- NWFSSP for finding an optimal solution for makespan is extensively studied but taking into account its practical application and theoretical background of NP hard nature, still has a challenge of developing an efficient and effective algorithm.

- According to computational results reported through the literature synthesis, metaheuristic methods are found to be the most promising solution methods. It is observed that the DPSOVND [18], IIGA [5], HDE [20], HGA [4], GA-VNS [21], TS/PSO [22], TMIIG [23] and MNE_{H1}/MNE_{H2}/LKH [24] methods are the state-of-the-art algorithms for solving NWFSSP for makespan criterion. Further, only TMIIG [23] and MNE_{H1}/MNE_{H2}/LKH [24] method provides Objective Function Value (OFV) to large problem instances up to 500 jobs. This necessitates developing a new hybridized method which can find out optimal solution for large problem instances.
- In spite of the fact that PSO is widely used metaheuristic for continuous optimization problem, application of PSO for discrete optimization problem like NWFSSP is less attempted. Earlier hybrid PSO attempts by Liu et al. [17] require adaptive learning strategy additionally. So, designing of an algorithm which can make use of PSO for global exploration along with some local search algorithm for local exploitation with reasonable computation time is necessary to be attempted.

Investigation of efficient algorithm for solving this problem has been done by most of the researchers [4, 18-20, 21, 22] for problem instances up to 100 jobs. Recently, some of the researchers Ding et al. [23] and Lin and Ying [24] had developed metaheuristic methods and provided solutions for Taillard benchmark instances for Permutation Flowshop Scheduling Problem (PFSSP) addressing large size problem up to 500 jobs.

Lin and Ying [24] compared performance of the proposed algorithm with algorithms developed by researchers Pan et al. [18], Qian et al. [20], Tseng and Lin [4], Jarboui et al. [21], Samarghandi and ElMekkaw [22] and also reported improvement in best known solutions for very large problem instances. This motivates further development of an efficient algorithm which can find out solution to small (n=20, 50, 100) as well as for large size problems (n=200, 500). Therefore, this work presents hybridization of metaheuristic to solve this problem optimally. The results produced by Lin and Ying [24], Ding et al. [23], Tseng and Lin [4], Quain et al. [20], Pan et al. [18, 19], and Liu et al.[17] are used for comparison with proposed algorithm. Carlier [25] , Heller [26], Revees [27], Taillard [28] have been used as test set for NWFSS makespan and are used here on the computational experiments to evaluate the performance of proposed approach in comparison to the state-of-art methods.

The remainder of this paper is organized as follows. Section 3 formally defines and formulates NWFSSP .Section 4 describes metaheuristics PSO and SA along with detailed procedure for implementing the proposed metaheuristics, PHPSOM. Section 5 describes the values obtained by PHPSOM on Reeve's benchmark suit and Taillard's benchmark suits and then compares the performance of the proposed PHPSOM with that of the best-so-far algorithms. Finally, section 6 discusses concluding remarks.

3. No-Wait Flow Shop Scheduling Problem (NWFSSP)

A No Wait Flowshop has set of N = {1, 2, 3,.. n} jobs and M=(1, 2, 3,…m} machines and the processing time p(i, j) of every 'i' job on each 'j' machine is given. Every job 'i' is processed by every machine 'j' following the same technological order. The processing of every job is continuous without any

interruption or preemption for additional constraint of no wait. To meet this constraint, job may be delayed at the beginning. So, in order to solve this type of problem a delay matrix (δ) needs to be calculated [29].

Let $\sigma = \{\sigma_1, \sigma_2, \ldots, \sigma_n\}$ correspond to the sequence of 'n' jobs to be processed on 'm' machines, and δ(i, s) correspond to the minimum delay introduced at first machine between the start of job 'i' and job 's' for meeting constraint with no wait. Also, let $p(\sigma_i, j)$ correspond to time of processing by machine j for the job at i^{th} position for a given sequence, and let $\delta(\sigma_{i-1}, \sigma_i)$denote the minimum delay occurred at first machine for the start of two successive jobs available at $(i-1)^{th}$ and i^{th} position for a given sequence. Let $C_{max}(\sigma i)$, i.e., makespan is completion time for the job available at i^{th} position in the sequence, which is given as:

$$C(\sigma 1) = \sum_{j=1}^{m} P(\sigma 1, j) \tag{1}$$

$$C(\sigma_2) = \delta(\sigma_1, \sigma_2) + \sum_{j=1}^{m} P(\sigma_2, j) \tag{2}$$

$$C_{\max}(\sigma_i) = \sum_{k=2}^{i} \delta(\sigma_{k-1}, \sigma_k) + \sum_{j=1}^{m} P(\sigma_n, j) \tag{3}$$

where

$$\delta_{i,k} = \max_{1 \le j \le m} \sum_{h=1}^{j} t_{i,h} - \sum_{h=2}^{j} t_{k(h-1)} \tag{4}$$

Finally, minimum completion time, i.e., makespan is given as:

$$C_{\max}(\sigma) = \min\{C_{\max}(\sigma_i)\} \tag{5}$$

4. Proposed Hybrid-PSO-Makespan (PHPSOM) for NWFSSP

PHPSOM essentially differs from the standard PSO in some characteristics. Suitable particles are selected from the present population. Afterwards, the selected particles are searched locally for finding effective neighbourhood when we design a PHPSOM. The proposed algorithm consists of the following three phases: Ranked- order-value (ROV) encoding scheme for representing solution of a particle, initial seed solution generation with NEH algorithm, a PSO based global search technique to avoid getting trapped in local minima, along-with SA based local search method to improve the present solution. A pre-defined termination condition is used to terminate the global search and the local search. In this section, each phase of the proposed algorithm is described followed by the detailed steps of PHPSOM algorithm.

4.1. Particle swarm optimization

This section briefs about PSO, SA and provides the details regarding use of PSO for developing PHPSOM algorithm for solving NWFSSP for makespan objective criterion. PSO simulates its behaviour from the biological example of flock of birds and used widely as an optimization algorithm. The idea is inspired from searching strategy adopted by birds for their food. PSO algorithm works on similar principle and find the best solution in the given search space.

Particle in PSO is used to represent a single solution. The fitness value of each particle is evaluated by the objective function. The velocity of each particle provides

flying direction for food, in this context reaches towards approximate solution for the given objective function. Standard theory and procedure of PSO is well defined by Kennedy and Eberhart [30]. Further, the performance of PSO for solving NWFSSP problem is compared with other metaheuristics viz. GA and TS by Bewoor et al. [31, 32]. They concluded that PSO gives better solution than other metaheuristics.

In PSO solution in the search space is called as particle and each particle is initialized with random positions and later explores the entire search space for finding better solution [33]. In each iteration, velocity is adjusted by the particle in order to pursue two best solutions. The first case in which own best solution found so far is followed by particle called as *pbest* and observed as cognitive part and other is current best solution of swarm called *gbest*. PSO can be presented with global version and the local version based on learning approaches. In global PSO version, learning takes place by each particle from the best particle from entire swarm. Whereas, in the local version, learning is inspired from best particle in its neighbourhood. Among these two versions, slower convergence speed is observed in the local PSO version and hence, it can be used for a varying environment more easily and which is exactly needed in NWFSS context. The new velocity denoted by V_{new} and new position denoted by X_{new} given in Eqs. (6) and (7).

$$V_{new} = w * V_{curr} + c1 * r1 * (pbest - X_{curr}) + c2 * r2 * (gbest - X_{curr}) \quad (6)$$

$$X_{new} = X_{curr} + V_{new} \quad (7)$$

where '*w*' is inertia weight which provides balance between local and global search capabilities. The acceleration constants c_1 and c_2 in Eq. (6) are cognitive parameter which develops the bird's own confidence (cognitive behaviour) and confidence in swarm (social behaviour) respectively. The low values of c_1, c_2 may direct particles to roam far from target regions whereas high values may lead towards hasty movement from target regions so these acceleration coefficients should be appropriately adjusted. In this paper the value of w, c1 and c2 is set with trial and error method. X_{new} and V_{new} is new position and velocity of particle respectively. X_{curr} and V_{curr} are current position and velocity of particle respectively. In the standard PSO, new velocity of the particle is found by Eq. (6) considering its earlier velocity and its distance from current position to its own best historical position and distance from current position to its neighbours' best position. Generally, the velocity value is set between (V_{max}, $-V_{max}$) due to which, particles cannot roam excessively outside the search space. With this new velocity, the particle moves towards a new position according to Eq. (7). This process stops when user-defined terminating criterion is met.

Further, convergence of PSO was observed on benchmark dataset and it was observed that the algorithm get stucked in local optima as velocity is the only parameter for divergence of the swarm's particles. So, the basic algorithm does not offer a mechanism of escape from local optima. Still if such particles are allowed to continue in this state may lead to exploitation following the initial phase of exploration. The empirical results show that after few iterations velocities are so small that even the nearest solution may be eliminated from the search space. Hence it is concluded that traditional PSO alone can't be considered for local and global search.

4.2. Solution representation

Solution representation is one of the most important issues while designing PSO algorithm. For representing the solution, earlier researchers [34] adopted encoding scheme which was a job-permutation-based method for solving NWFSSP. Generally, in PSO technique the position of particles is following continuous characters, hence, a classical encoding method of PSO can't be adopted for NWFSSP directly. In order to effectively apply PSO for solving NWFSS, '*n*' numbers of dimensions are considered for 'n' number of jobs. Each dimension represents a typical job and related particle information X_i= {x_1, x_2, x_3,…….x_n) for '*n*' number of jobs in the NWFSS.

In PSO, a permutation cannot be presented with the particle itself, which necessitates finding appropriate mapping between the job's sequence and particles' position. So, in this paper the ranked-order-value (ROV) rule based on random key value [35] is used to determine the permutation implied by the position values x_{ij} of particle X_i. The ROV rule converts the continuous position values of particle to a discrete job permutation. This enables to apply the continuous PSO algorithms to discrete sequencing problems, which in turn evaluates the performance of a particle.

Specifically, particle information is represented as X_i={x_1, x_2,…..x_n}, which cannot represent a sequence of job. So, permutations of jobs are constructed by considering a job index, which is the rank of each particle's position. ROV rule used in PHPSOM handles the particle with smallest position value first and rank value 1 is assigned to the position of the particle. In case of two or more particles with same position values, the position having the smallest dimension number is given first priority and assigned a rank value first. The remaining values of position are incremented by 1 and subsequently assigned the next rank values are set with respect to dimension number. Further, the next value of smallest position is handled in the similar way. Thus, the particle's position information is changed to corresponding job permutation σ_{ij}= [j_1, j_2, j_3, …j_n].To demonstrate the scheme of the ROV rule, we provide a simple example in Table 2.

Table 2.Representation of Solution of particle's position information and its corresponding ROV with corresponding job permutation.

Dimension	**1**	**2**	**3**	**4**	**5**
Xij	5.45	4.22	4.37	5.47	4.37
Job Permutation	4	1	2	5	3

Let's consider the random position values of particle (for *n*=5) observed initially are X_i= {5.45, 4.22, 4.37, 5.47, 4.37}.As $x_{1,2}$=4.22 has small value of particle's position so, $x_{1,2}$ is ranked first with value 1. Next two particles $x_{1,3}$ and $x_{i,5}$ has equal position value, i.e., 4.37. But, index of $x_{1,3}$ is smaller as compared to $x_{1,5}$, so $x_{1,3}$ is assigned the next rank value as 2 and $x_{1,5}$ gets next incremented ranked value as 3. Finally, rank values of 4 and 5, respectively are assigned to $x_{1,1}$ and $x_{1,4}$. Thus, the job permutation obtained considering position information value of particle and corresponding rank assignment based on the ROV rule is, σ_{ij}=[4, 1, 2, 5, 3].

PHPSOM apply, job permutation based local search approaches rather than direct consideration of the particle's position information. Hence, it is necessary to

convert particle's position information to corresponding job permutation as per ROV rule after completing a local search. Because of the simple mechanism of the ROV rule, adjustment for new particle position will be very easy. Local search methods using the position information are handled in the same way as the process adopted for permutation of jobs. For instance, as shown in Table 3, if job 2 and job 4 are swapped using a local search based SWAP operator [34], it lead to swapping of position value 4.22 and 4.37.

Table 3. Representation of solution after swapping job2 and job4.

Dimension	**1**	**2**	**3**	**4**	**5**
Xij	4.37	4.22	5.45	5.47	4.37
Job Permutation	2	1	4	5	3

4.3. Population initialization

The initial swarm generation is random often in the standard PSO. In order to get an efficient solution, initial population should have a certain quality along with diversity. In this paper, NEH heuristic [16] is used as an efficient population initialization procedure. In order to find NEH based seed sequence, jobs are ordered according to descending makespan values. The partial schedules depending on the initial order are taken into account for constructing a job sequence. Consider a current sequence σ_{ij} = [4, 1, 2, 5, 3] is determined and job 4 at index 'i' is the first job then partial sequences are constructed by inserting job 4 at all indexes where 'i = i+1' of the current sequence. Among all these sequences, the sequence generating the minimum makespan is kept as the current sequence for the next iteration. The procedural steps are given in Algorithm 1.Thus, initial population generation with NEH technique helps to find better quality of a job permutation as compared to a random initial population.

Algorithm 1 PSO_NEH Algorithm

Step 1: Order the jobs by non increasing order of makespan.
Step 2: Consider the first sequence, (σ_1) of job and find makespan. Swap first position of particle with next position of particle and compute makespan value for new sequence(new_seq).
Step 3: for i=1 to n do
    3.1 Swap $\sigma_i$with σ_{i+1} and find makespan
    3.2 if makespan (σ_i)> makespan(σ_{i+1}) set new_seq=σ_i
        else new_seq=σ_{i+1}
Step 4: End for
Step 5: return new_seq

4.4. SA based local search

In metallurgy, annealing process is the process where metals are cooled slowly and are reached to a state of low energy where they are very strong [21]. At high temperature the movements are random whereas at low temperature, little randomness is observed. This paper reports, the use of SA as a local search method for finding neighbourhood that is, possible job sequences leading towards minimum makespan in the context of NWFSS.

SA starts a random search at high temperature eventually the temperature is reduced slowly, and becoming pure greedy descent as it approaches to zero temperature. Random changes in the temperature not only help to escape from local minima but also help to find low heuristic value regions. The results may be worst initially at high temperatures, but better improvement can be observed gradually at lower temperatures. For minimization of given objective function, temperature should be reduced according the probability given by the Boltzmann factor given in Eq. (8).

$$P = e^{\frac{-\Delta E}{\alpha T}} \tag{8}$$

where α is the Boltzmann constant and T is the current temperatureΔE is change in energy.The Boltzmann probability is a random number between 0 and 1 drawn from a uniform distribution; if the Boltzmann probability is more than the random number, the configuration is accepted. This allows the algorithm to escape from local minima. The initial value of temperature should be maintained high in order to visit all system states with an equal probability.

In order to visit all the system states appropriately a proper initial temperature should be high enough but at the same time, it should not be too high which may lead to lot of unnecessary searches. In this paper, we tuned an initial temperature by trial and error. Similarly cooling rate should be in between 0 to 1 and hence trails are made and finally set the value for cooling rate.

Algorithm 2: SA Algorithm

Step 1. Initialize temperature init_temp as 3.0 and final _temp as 0.9 and cooling rate α as 0.99
Step 2: Initialize Best_found to current state
Step 3: while init_temp < final_temp do
3.1 for i= 0 to n-1
3.1.1: Randomly perturb from the current state to a new state and calculate corresponding objective function value, i.e., makespan.
3.1.2:Update gbest depending on best particle.
3.1.3: Calculate the difference in makespan value between current and new state and set it as ΔE.
3.1.4: If $\Delta E < 0$, i.e., new state has minimum makespan, accept new state as current state. Set Best_ found_ Solution to this new state
3.1.5: If $\Delta E \geq 0$, consider new state as current state with probability by invoking random number between range(0,1).
3.1.6: Prob (accepted) = exp (- ΔE / α.init_temp)
3.1.7: Revise init_temp as necessary according to annealing schedule.
Step 3.2: End for
Step 4: End while
Step 5: set gbest to Best_found
Step 6: return gbest

4.5. Computational procedure

PHPSOM algorithm is based on the solution representation by ROV rule, population initialization with NEH-based local search and neighbourhood searching through SA-based local search. The complete computational procedure of PHPSO framework for the NWFSS can be summarized as follows:

Procedure: *PHPSOM*

Step 1: Input the total no. of jobs (n), total no. of machines (m) and processing time matrix (arrPT). Calculate delay matrix (del_mat) as per equation 1.
Step 2:
For *i* :=0 to n -1 do
2.1 Initialize particle *i* with random value (particle_pval) and velocity (particle_velocity). Set the acceleration constants c_1 and c_2 each equal to1.65 and 1.75 respectively; r_1 and r_2 are set to the value 0.5 each.
2.2 Apply ROV rule to represent random value of particle to position of particle (particle_pbest).
2.3 Calculate the processing sequence of job (job_seq) based on particle's position as shown in Table 2.
2.4 Evaluate objective function value makespan as per equation2.
End for
Step 3: Sort the particles with decreasing order of makespan value.
Step 4: Generate initial seed sequence with NEH algorithm as per Algorithm 2.
Step 5: Calculate pbest (mpbest) of particle and gbest (pgbest) of swarm for generated the initial seed sequence.
Step 6: Repeat
Step 7: Select particle for local refinement from present population
repeat
7.1 for *i* :=0 to n -1 do
7.1.1 Particle's velocity and position is updated according to equation (10) and equation (11), respectively
7.1.2 Update value of particle (particle_pval) and apply ROV rule to find next job permutation.
7.1.3 Calculate makespan value for the updated particle.
7.1.4 If updated_makespan_value > current_makespan_value and gbest (pgbest) then
7.1.5 update pbest of particle (mpbest) and gbest (pgbest)
end for
until maximum iteration reached
Step 8: Select minimum pgbest value as best_particle from the population for further local refinement;
Step 9: Explore local search space with SA based local search as per Algorithm 2 with initial solution as best_particle returned from PSO.
until pgbest of consecutive iterations are same.
Step 10: Output gbest
End Procedure

Thus, it can be observed that the Proposed Hybrid PSO Makespan (PHPSOM) can effectively explore promising solution within entire region along with

exploitation for solution improvement in sub regions. Because of NP-Hard nature of NWFSSP, PHPSOM applies local search methods which includes NEH-based local search and SA based local search. Since both exploration and exploitation are stressed and balanced, it is expected to achieve good results for the NWFSSP. The next section, investigates the performance of PHPSOM based on numerical simulations.

5. Computational Results

This section provides details about computational experiments performed by PHPSOM to evaluate its performance for solving the NWFSSP for makespan objective function. The following sub- sections provide the details about experimental setup and then compare the computational results obtained by applying PHPSOM to the test problems with those obtained using other state-of-the-art algorithms.

5.1. Experimental setup

PHPSOM is coded in java and run on Intel Core i5, 8 GB RAM, 2.20 GHz PC with Windows 7 operating environment. The effectiveness and efficiency of PHPSOM is verified with two sets of benchmark problem instances. The first benchmark suit consists of 31 problem instances from OR library provided by Carlier [25] abbreviated as car1-car8, Heller [26] abbreviated as hel1-hel2 and Reeves [27] abbreviated as reC01-reC42 with different problem sizes (number of jobs X number of machines).

As the above problem instances are observed as small scale, the algorithm is tested for large scale problem instances of Taillard benchmark suit [28]. The second benchmark problem suit comprises of 120 problem instances contributed to Taillard dataset. The Taillard benchmark set is composed of 12 groups of the problems of size ranging from 20 jobs and 5 machines to 500 jobs and 20 machines with 10 instances of each problem size.

Further, these subsets are denoted as 20×5 (ta001-ta010), 20×10 (ta011-ta020), 20×20 (ta021-ta030), 50×5 (ta031-ta040), 50×10 (ta041-ta050), 50×20 (ta051-ta060), 100×5 (ta061-ta070), 100×10 (ta071-ta080), 100×20 (ta081-ta090), 200×10 (ta091-ta100), 200×20 (ta101-ta110) and 500×20 (ta111-ta120) for representing the number of jobs and machines, respectively.

5.2. Computational and statistical evaluation

To compare the performance of PHPSOM with the existing metaheuristics, experimentation was carried out by running each instance independently 10 times. 'Average Relative Percentage Deviation' (*ARPD*) is used as a performance measure, which is popular in the scheduling literature [13, 19]. *ARPD* is given by,

$$\mathrm{ARPD} = \frac{100}{k}\sum_{i=1}^{k}\frac{Heuristic_i - BestH_i}{BestH_i} \tag{9}$$

where $\mathrm{Heuristic_i}$ is the makespan obtained by any of four algorithms and the $\mathrm{BestH_i}$ was the lowest makespan obtained for that specific instance. Table 4 displays comparative evaluation of PHPSOM with MNEH+LKH [24], TMIIG [23], HGA [4] based on *ARPD* for Taillard benchmark data suite [28].

Table 4. Comparison of Performance of the existing metaheuristics and the proposed method for Taillard's Benchmark suit.

Inst-ances	MNEH+ LKH	TMI IG	HGA	PHP SOM	Inst-ances	MNEH+ LKH	TMI IG	HGA	PHP SOM
ta001	0.22	0.22	0.22	0.00	**ta061**	0.09	0.09	0.13	0.00
ta002	0.23	0.23	0.21	0.00	**ta062**	0.08	0.08	0.12	0.00
ta003	0.26	0.26	0.23	0.00	**ta063**	0.21	0.21	0.25	0.00
ta004	0.18	0.18	0.17	0.00	**ta064**	0.08	0.09	0.12	0.00
ta005	0.16	0.16	0.15	0.00	**ta065**	0.09	0.09	0.13	0.00
ta006	0.21	0.21	0.20	0.00	**ta066**	0.07	0.08	0.11	0.00
ta007	0.23	0.23	0.24	0.00	**ta067**	0.09	0.10	0.14	0.00
ta008	0.15	0.15	0.15	0.00	**ta068**	0.22	0.22	0.27	0.00
ta009	0.13	0.13	0.10	0.00	**ta069**	0.03	0.03	0.00	0.00
ta010	0.16	0.16	0.16	0.00	**ta070**	0.08	0.08	0.11	0.00
ta011	0.36	0.36	0.33	0.00	**ta071**	0.07	0.08	0.10	0.00
ta012	0.29	0.29	0.29	0.00	**ta072**	0.02	0.01	0.01	0.00
ta013	0.21	0.21	0.21	0.00	**ta073**	0.09	0.09	0.11	0.00
ta014	0.16	0.16	0.16	0.00	**ta074**	0.10	0.10	0.13	0.00
ta015	0.37	0.37	0.38	0.00	**ta075**	0.07	0.07	0.09	0.00
ta016	0.27	0.27	0.26	0.00	**ta076**	0.04	0.03	0.02	0.00
ta017	0.53	0.53	0.54	0.00	**ta077**	0.07	0.07	0.10	0.00
ta018	0.29	0.29	0.30	0.00	**ta078**	0.08	0.08	0.11	0.00
ta019	0.24	0.24	0.24	0.00	**ta079**	0.09	0.09	0.11	0.00
ta020	0.40	0.40	0.38	0.00	**ta080**	0.02	0.02	0.01	0.00
ta021	0.55	0.55	0.55	0.00	**ta081**	0.02	0.02	0.00	0.00
ta022	0.71	0.71	0.70	0.00	**ta082**	0.12	0.12	0.16	0.00
ta023	0.44	0.44	0.43	0.00	**ta083**	0.11	0.11	0.13	0.00
ta024	0.54	0.54	0.54	0.00	**ta084**	0.24	0.24	0.26	0.00
ta025	0.56	0.56	0.57	0.00	**ta085**	0.10	0.11	0.13	0.00
ta026	0.44	0.44	0.43	0.00	**ta086**	0.09	0.09	0.11	0.00
ta027	0.42	0.42	0.41	0.00	**ta087**	0.01	0.01	0.03	0.00
ta028	0.46	0.46	0.45	0.00	**ta088**	0.23	0.23	0.25	0.00
ta029	0.50	0.50	0.50	0.00	**ta089**	0.09	0.09	0.11	0.00
ta030	0.45	0.45	0.45	0.00	**ta090**	0.23	0.24	0.27	0.00
ta031	0.13	0.13	0.16	0.00	**ta091**	0.06	0.05	0.02	0.06
ta032	0.17	0.17	0.18	0.00	**ta092**	0.12	0.08	0.04	0.13
ta033	0.15	0.15	0.17	0.00	**ta093**	0.03	0.03	0.08	0.00
ta034	0.04	0.04	0.05	0.00	**ta094**	0.07	0.06	0.00	0.07
ta035	0.29	0.29	0.31	0.00	**ta095**	0.02	0.03	0.07	0.00
ta036	0.04	0.04	0.05	0.00	**ta096**	0.07	0.07	0.03	0.08
ta037	0.15	0.15	0.17	0.00	**ta097**	0.02	0.03	0.07	0.00
ta038	0.14	0.14	0.16	0.00	**ta098**	0.02	0.03	0.07	0.00
ta039	0.13	0.13	0.15	0.00	**ta099**	0.15	0.16	0.21	0.00
ta040	0.14	0.14	0.17	0.00	**ta100**	0.16	0.17	0.22	0.00
ta041	0.06	0.06	0.06	0.00	**ta101**	0.07	0.06	0.02	0.07
ta042	0.05	0.05	0.07	0.00	**ta102**	0.03	0.04	0.08	0.00
ta043	0.15	0.15	0.16	0.00	**ta103**	0.08	0.07	0.03	0.08
ta044	0.07	0.07	0.11	0.00	**ta104**	0.08	0.08	0.04	0.09
ta045	0.34	0.34	0.35	0.00	**ta105**	0.02	0.03	0.07	0.00
ta046	0.04	0.04	0.05	0.00	**ta106**	0.04	0.04	0.09	0.00
ta047	0.06	0.06	0.07	0.00	**ta107**	0.02	0.03	0.07	0.00
ta048	0.09	0.09	0.09	0.00	**ta108**	0.03	0.04	0.08	0.00
ta049	0.12	0.12	0.13	0.00	**ta109**	0.06	0.06	0.02	0.07
ta050	0.16	0.16	0.17	0.00	**ta110**	0.02	0.03	0.07	0.00
ta051	0.17	0.17	0.18	0.00	**ta111**	0.09	0.10	0.17	0.00
ta052	0.08	0.08	0.09	0.00	**ta112**	0.08	0.10	0.17	0.00
ta053	0.09	0.09	0.10	0.00	**ta113**	0.05	0.04	0.02	0.00
ta054	0.33	0.33	0.34	0.00	**ta114**	0.04	0.03	0.03	0.00
ta055	0.12	0.12	0.13	0.00	**ta115**	0.04	0.03	0.03	0.00
ta056	0.10	0.10	0.11	0.00	**ta116**	0.08	0.09	0.16	0.00
ta057	0.19	0.19	0.19	0.00	**ta117**	0.08	0.09	0.16	0.00
ta058	0.23	0.23	0.24	0.00	**ta118**	0.13	0.12	0.06	0.14
ta059	0.36	0.36	0.37	0.00	**ta119**	0.09	0.10	0.17	0.00
ta060	0.10	0.10	0.11	0.00	**ta120**	0.08	0.09	0.17	0.00

Table 5 displays comparative evaluation of the PHPSOM with MNEH+LKH [24], TMIIG [23], HGA [4], HDE [20], IIG [5], HPSO [17] based on *ARPD* for Reeve's dataset [27].

Table 5. Comparison of performance of the existing metaheuristics and the proposed method for Reeve's Benchmark suit.

Instances	MNEH+ LKH	IMIIG	HGA	HDE	IIG	DPSO	HPSO	PHPSOM
Rec001	0.31	0.31	0.31	0.31	0.31	0.31	0.32	0.00
Rec003	0.14	0.14	0.14	0.14	0.14	0.14	0.15	0.00
Rec005	0.18	0.18	0.18	0.18	0.18	0.18	0.19	0.00
Rec007	0.43	0.43	0.43	0.43	0.43	0.43	0.43	0.00
Rec009	0.27	0.27	0.27	0.27	0.27	0.27	0.27	0.00
Rec011	0.26	0.26	0.26	0.26	0.26	0.26	0.27	0.00
Rec013	0.50	0.50	0.50	0.50	0.50	0.50	0.51	0.00
Rec015	0.35	0.35	0.35	0.35	0.35	0.35	0.36	0.00
Rec017	0.33	0.33	0.33	0.33	0.33	0.33	0.33	0.00
Rec019	0.15	0.15	0.15	0.15	0.15	0.15	0.16	0.00
Rec021	0.21	0.21	0.22	0.22	0.22	0.22	0.23	0.00
Rec023	0.25	0.25	0.25	0.25	0.25	0.25	0.27	0.00
Rec025	0.22	0.22	0.22	0.22	0.22	0.22	0.23	0.00
Rec027	0.31	0.31	0.31	0.32	0.31	0.32	0.33	0.00
Rec029	0.19	0.19	0.19	0.19	0.19	0.19	0.20	0.00
Rec031	0.31	0.31	0.32	0.32	0.31	0.32	0.34	0.00
Rec033	0.31	0.31	0.32	0.32	0.32	0.32	0.36	0.00
Rec035	0.16	0.16	0.17	0.17	0.17	0.17	0.19	0.00
Rec037	0.16	0.16	0.01	0.23	0.01	0.01	0.03	0.00
Rec039	0.15	0.15	0.16	0.16	0.15	0.16	0.18	0.00
Rec041	0.17	0.17	0.17	0.17	0.17	0.17	0.20	0.00
Rec001	0.31	0.31	0.31	0.31	0.31	0.31	0.32	0.00
Rec003	0.14	0.14	0.14	0.14	0.14	0.14	0.15	0.00
Rec005	0.18	0.18	0.18	0.18	0.18	0.18	0.19	0.00
Rec007	0.43	0.43	0.43	0.43	0.43	0.43	0.43	0.00
Rec009	0.27	0.27	0.27	0.27	0.27	0.27	0.27	0.00
Rec011	0.26	0.26	0.26	0.26	0.26	0.26	0.27	0.00
Rec013	0.50	0.50	0.50	0.50	0.50	0.50	0.51	0.00
Rec015	0.35	0.35	0.35	0.35	0.35	0.35	0.36	0.00
Rec017	0.33	0.33	0.33	0.33	0.33	0.33	0.33	0.00
Rec019	0.15	0.15	0.15	0.15	0.15	0.15	0.16	0.00
Rec021	0.21	0.21	0.22	0.22	0.22	0.22	0.23	0.00
Rec023	0.25	0.25	0.25	0.25	0.25	0.25	0.27	0.00
Rec025	0.22	0.22	0.22	0.22	0.22	0.22	0.23	0.00
Rec027	0.31	0.31	0.31	0.32	0.31	0.32	0.33	0.00
Rec029	0.19	0.19	0.19	0.19	0.19	0.19	0.20	0.00
Rec031	0.31	0.31	0.32	0.32	0.31	0.32	0.34	0.00
Rec033	0.31	0.31	0.32	0.32	0.32	0.32	0.36	0.00
Rec035	0.16	0.16	0.17	0.17	0.17	0.17	0.19	0.00
Rec037	0.16	0.16	0.01	0.23	0.01	0.01	0.03	0.00
Rec039	0.15	0.15	0.16	0.16	0.15	0.16	0.18	0.00

Table 6 displays comparative evaluation of the PHPSOM with HDE [20], IIG [18], and DPSO [18] for Heller's dataset [26].

Table 6. Comparison of Performance of the existing metaheuristics and the proposed method PHPSOM for Heller's Benchmark suit for makespan.

Instances	HDE	IIG	DPSO	PHPSOM
Hel1	0.04	0.04	0.06	0.04
Hel2	0.34	0.34	0.41	0.00

Table 7 displays comparative evaluation of the PHPSOM with HGA [4], HPSO [17] for Carlier dataset [25].

Table 7. Comparison of Performance of the existing metaheuristics and the proposed method PHPSOM for Carlier's Benchmark suit for makespan.

Instances	HGA	HPSO	PHPSOM	Instances	HGA	HPSO	PHPSOM
car1	0.32	0.32	0.00	**car5**	0.50	0.50	0.00
car2	0.20	0.20	0.00	**car6**	0.68	0.68	0.00
car3	0.23	0.23	0.00	**car7**	0.71	0.71	0.00
car4	0.34	0.34	0.00	**car8**	0.75	0.75	0.00

Tables 4 to 7 indicate that PHPSOM improves 111 out of 120 instances of Taillard dataset and 28 out of 30 instances of Reeve's, Carlier and Heller dataset respectively. This shows that, the optimal results obtained by the PHPSOM are better than values obtained by various metaheuristic till date which exhibits the effective searching quality of PHPSOM.

Compared with the results by the HGA, TMIIG and MNEH+LKH method, for large size problem PHPSOM could improve the results to a great extent, which demonstrates the noteworthy improvement by PHPSOM over HGA, TMIIG and MNEH+LKH metaheuristics. Values obtained by PHPSOM are found better than the values reported by DPSO, HDPSO, IIG and HPSO for nearly all small size problem instances. So, it is concluded that PHPSOM is more effective than IITG [23] and MNEH and LKH [24], HGA [4] especially for large problem size.

Figures 1 and 2 illustrate that the performance of algorithm PHPSOM is independent on number of machines and is largely dependent on number of jobs to be sequenced for scheduling. Further, graphs also show that, makespan values given by all existing and proposed algorithms are increasing with increase in number of jobs.

In addition to the pair-wise comparison of the metaheuristics, statistical significance can be observed with the differences between the heuristics, the means of each metaheuristic with its 95% confidence level are shown in Figs. 3 and 4.

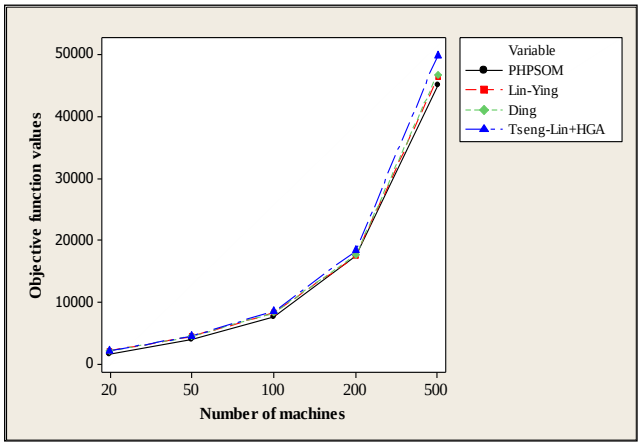

Fig.1. Comparative analysis of objective function values of makespan for Taillard's dataset.

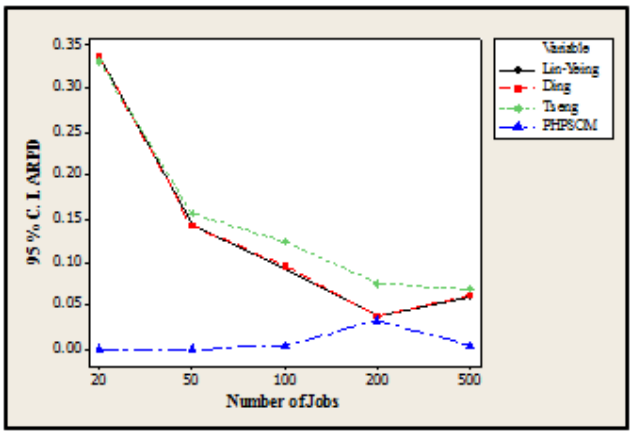

Fig.2. Comparative analysis of *ARPD* values of different algorithms for Taillard's dataset for makespan.

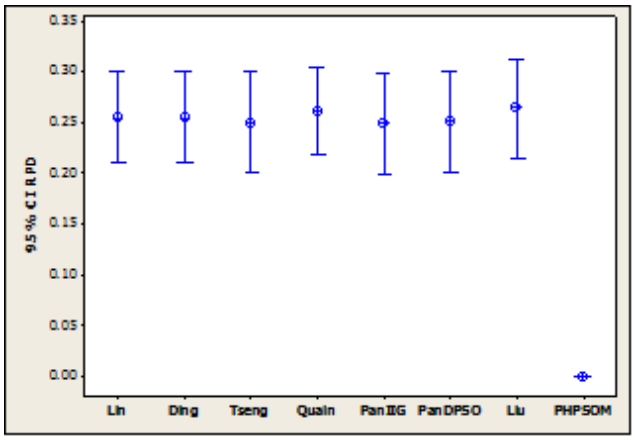

Fig. 3. Means and 95% confidence level intervals. for various algorithms for Reeve's dataset.

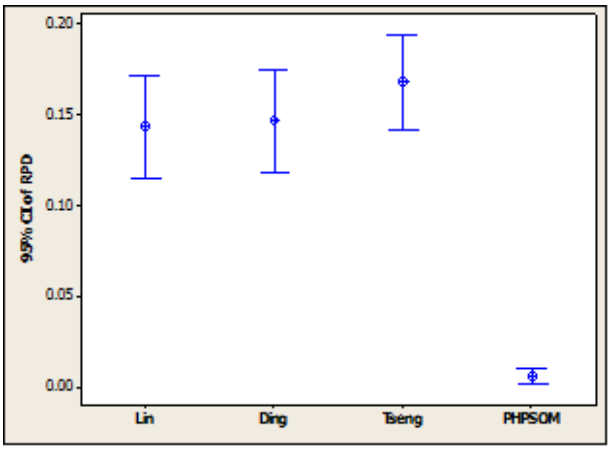

Fig. 4. Means and 95% confidence level intervals for various algorithms for Taillard's dataset.

6. Conclusion

This paper proposes hybridization of PSO with SA for the scheduling of flow shop with no-wait constraint. The developed PHPSOM algorithm not only applies evolutionary search guided by PSO mechanism, but also applies SA based local search with NEH based initial population. Thus, appropriate balance of exploration and exploitation is ensured globally and locally. The results and comparisons on various datasets used for simulation, exhibit the supremacy of PHPSOM in terms of searching quality and robustness of solutions.

The effectiveness of the proposed method was measured by using *ARPD*, which is widely used performance measure. We have carried out the extensive experimental and statistical analysis; and found that, 28 out of 30 of Reeve's dataset and 111 out of 120 of Taillard's benchmark suite objective function values are modified with PHPSOM. Comparative analysis of the solutions produced by PHPSOM and those available from literature (viz. MNEH+LKH, TMIIG, HGA, HDE, IIG, DPSO, HPSO) reveals that PHPSOM algorithm outperforms the existing methods. Hence, it can be concluded that, the PHPSOM algorithm is an improved hybrid algorithm for application of (PSO) to (NWFSSP) with makespan criterion.

Future, research directions may consider more constrained, complex, real life application based scheduling problems. It seems worthwhile to investigate and apply the applications of PHPSOM algorithm for real time and multi-objective scheduling problems.

Nomenclatures

$C(\sigma_i)$	Finishing time of the job at i^{th} position of a given sequence
c_1, c_2	Cognitive parameters
M	Number of machines $\{m_1, m_2, m_3, \ldots m_m\}$
N	Number of jobs $\{n_1, n_2, \ldots n_n\}$
$p(\sigma_i, j)$	processing time on machine j of the job at the i^{th} position of a given sequence
$p(i,j)$	Processing time of i^{th} job on j^{th} machine
V_{curr}	current velocity of particle
V_{new}	Updated velocity of particle
w	inertia weight
X_{curr}	Current position of particle
X_{new}	Updated position of particle

Greek Symbols

α	Boltzmann constant
ΔE	Difference in energy
δ	delay matrix
$\delta(i,s)$	delay introduced on the first machine for the start of jobs i and s
$\delta(\sigma_{i-1}, \sigma_i)$	delay occurred on the first machine for the start of two successive jobs found at $(i-1)^{th}$ and i^{th} position of the sequence
σ	Sequence of n jobs

Abbreviations

ARPD	Average Relative Percentage Deviation

References

1. Rajendran, C. (1994). A no-wait flowshop scheduling heuristic to minimize makespan . *Journal of Operation Research Society*, 45, 472-478.
2. Grabowski, J.; and Pempera, J. (2000). Sequencing of jobs in some production system. *European Journal of Operational Research*, 125(3), 535-550.
3. Raaymakers, W.; and Hoogeveen J. (2000). Scheduling multipurpose batch process industries with no-wait restrictions by simulated annealing. *European Journal of Operational Research*, 12(6),131-151.
4. Tseng, L. Y; and Lin, Y.T. (2010). A hybrid genetic algorithm for no -wait flow shop scheduling problem. *International Journal of Production Economics*, 128(1),144-152.
5. Pan, Q.K.; Wang, L., and Zhao, B.H. (2008). An improved iterated greedy algorithm for the no - wait flow shop scheduling problem with makespan criterion. *International Journal of Advanced Manufacturing Technology*, 38(7-8), 778-786.
6. Pinedo, M. (2009). *Scheduling: Theory, algorithms and systems* (4th ed.).New Jersey:Prentice-Hall.
7. Rock, H. (1984). The three-machine no-wait flow shop is NP-complete. *Journal of the Association for Computing Machinery*, 31(2), 336-345.
8. Garey, M.; and Johnson, D. (1979).*Computers and intractability, a guide to the theory of NP -completeness* (4th Ed.). New York: Freeman.
9. Graham, R.L. (1979). Optimization and Approximation in Deterministic Sequencing and Scheduling. *Annals of Discrete Mathematics*, 5, 287-326.
10. Schuster, C.J.; and Framinan, J.M. (2003). Approximative procedure for no -wait job shop scheduling.*Operation Research Letter*, 31, 308-318.
11. Aldowaisan, T.; and Allahverdi, A. (2003). New heuristics for no-wait flowshops to minimize makespan. *Computers & Operation Research*, 30, 1219-1231.
12. Grabowski, J.; and Pempera, J. (2005). Some local search algorithms for no-wait flow-shop problem with makespan criterion. *Computers & Operation Research*, 32(8), 2197-2212.
13. Laha, D.; and Chakraborty, U.K. (2009). A constructive heuristic for minimizing makespan in no -wait flow shop scheduling. *International Journal of Advanced Manufacturing Technology*, 41(1-2),97-109.
14. Blum, C.; and Roli, A. (2003). Metaheuristics in combinatorial optimization: Overview and conceptual comparison. *ACM Computing Surveys*, 35(3), 268-308.
15. Blum, C.; Puchinger, J.; Raidl, G. R.; and Roli, A. (2011). Hybrid metaheuristics in combinatorial optimization : A survey. *Applied Soft Computing Journal*, 11(6), 4135-4151.
16. Nawaz, M.; Enscore, Jr. E.; Ham, I. (1983). A Heuristic Algorithm for the m-Machine, n-Job Flow-shop Sequencing Problem.*OMEGA, The International Journal. of Management Science*, 11(1),91-95.
17. Liu, B.; Wang, L.; and JinY.H. (2007). An effective hybrid particle swarm optimization for no - wait flow shopscheduling. *International Journal of Advanced Manufacturing Technology*, 31,1001-1011.

18. Pan, Q.K.; Tasgetiren, M.F.; and Liang, Y.C. (2008). A discrete particle swarm optimization algorithm for the no -wait flowshop scheduling problem. *Computers& Operations Research*, 35(9), 2807-2839.
19. Pan, Q.K.; Liang, Y.C.; Tasgetiren M.F.; and Zhao B.H. (2008). A hybrid discrete particle swarm optimization algorithm for the no-wait flow shop scheduling problem with makespan criterion. *International Journal of Advanced Manufacturing Technology*,38(3-4), 337-347.
20. Qian, B.; Wang, L.; Hu, R.; Huang, D.X. ; and Wang X. (2009). A DE-based approach to no-wait flow-shop scheduling.*Computers & Industrial Engineering*, 57(3), 787- 805.
21. Jarboui, B.; Ibrahim, S.; Siarry, P.; and Rebai, A. (2008). A combinatorial particle swarm optimization for solving permutation flowshop problems.*Computers and Industrial Engineering*, 54(3), 526-538.
22. Samarghandi, H; and ElMekkawy T.Y. (2012). A metaheuristic approach for solving the no-wait flowshop problem. *International Journal of Production Research*, 50 (24),7313-7326.
23. Ding, J.Y.; Song, S.; Gupta, J.N.D.; Zhang, R..; Chiong, R.; and Wu C. (2015). An improved iterated greedy algorithm with a Tabu-based reconstruction strategy for the no-wait flowshop scheduling problem. *Applied Soft Computing*, 30(5),604-613.
24. Lin, S.; and Ying, K. (2015). Optimization of makespan for no-wait flowshop scheduling problems using efficient metaheuristics. *Omega*, 64, 116-125.
25. Carlier, J. (1978). Ordonnancements a contraintes disjonctives. *Radio Operation Research*, 12(4), 333-350.
26. Heller, J. (1960). Some numerical experiments for an M × J flow shop and its decision -theoretical aspects. *Operations Research*, 8, 178-84.
27. Reeves, C.R. (1994). Genetic algorithms and neighbourhood search. *Evolutionary computing: Lecture Notes in Computer Science*, 865, 115-130.
28. Taillard, E. (1993). Benchmarks for basic scheduling problems. *European Journal Operational Research*, 64(2), 278-285.
29. Sapkal, S.U.; and Laha, D. (2013). A heuristic for no-wait flow shop scheduling. *International Journal of Advanced Manufacturing Technology*, 68(5-8), 1327-1338.
30. Kennedy, J.; and Eberhart, R. (1995). Particle swarm optimization. *Proceedings ofICNN'95-IEEE International Conference on Neural Networks*, IEEE CS Press, Perth, WA, Australia, 1942-1948.
31. Bewoor, L.A.; Chandra Prakash, V.; and Sapkal, S.U. (2016).Comparative analysis of metaheuristic approaches for m-Machine no wait flow shop scheduling for minimizing total flow time with stochastic input. *International Journal of Engineering and Technology*, 8(6), 3021-3026.
32. Bewoor, L.; Chandra Prakash V.; and Sapkal S. (2017). Comparative analysis of metaheuristic approaches for makespan minimization for no wait flow shop scheduling problem. *International Journal of Electrical and Computer Engineering*,7, 31-37.
33. Tasgetiren, M. F.; Liang, Y. C.; Sevkli, M.; and Gencyilmaz, G. (2007). A particle swarm optimization algorithm for makespan and total flowtime

minimization in the permutation flowshop sequencing problem. *European Journal of Operational Research*, 177(3), 1930-1947.

34. Wang, L.; and Zheng, D.Z. (2003). An effective hybrid heuristic for flow shop scheduling. *International Journal of Advanced Manufacturing Technology*, 21(1), 38-44.
35. Bean, J.C. (1994). Genetic algorithms and random keys for sequencing and optimization. *ORSA Journal of Computing*, 6(2), 154-160.