AN IMPROVED EVOLUTIONARY HYBRID PARTICLE SWARM OPTIMIZATION ALGORITHM TO MINIMIZE MAKESPAN FOR NO WAIT FLOW SHOP SCHEDULING

[1]LAXMI A. BEWOOR*, [2]V.CHANDRAPRAKASH2, [3]SAGAR U.SAPKAL

[1]Research Scholar, Computer Sci. and Engineering, K.L.University, Guntur, A.P. India;

[2] Professor , Computer Sci. and Engineering, K.L.University, Guntur, A.P. India;

[3] Associate Professor, Mechanical Engineering, Shivaji University, Sangli, Maharashtra, India

E-mail: [1] laxmiabewoor@gmail.com, [2]vchandrap@kluniversity.in, [3]sagarus1201@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 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: *No-wait flow shop, Scheduling, Particle Swarm Optimization, Simulated Annealing, Makespan*

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 (Rajendran, 1994), food processing to ensure food freshness (Grabowski and Pempera, 2000), concrete ware production for proper molding of concrete blocks (Raaymakers and Hoogeveen, 2000] and pharmaceutical processing for guarantying appropriate chemical reactions (Tseng and Lin, 2010) 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 (Pan et al., 2008). Additionally, makespan is considered as an important performance measure which, when optimized, evaluate the performance of the solutions (Pinedo, 2002). 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 (Rock, 1984; Garey, 1979; Graham, 1979) 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 viz. (Rajendran, 1994; Schuster and Framinan, 2003; Aldowaisan and Allahvardi, 2003; Grabowski and Pempera, 2005; Laha and Chakraborty, 2009), had developed efficient constructive heuristic algorithms for minimizing makespan. But, these methods face the problem for large sequence of jobs, and continue processing till all the jobs are incorporated (Laha and Chakraborty, 2009) 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 (Blum and Roli, 2003). 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 (Blum et al., 2011). 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 chronological order.

Table 1. Metaheuristic Approaches For Solving NWFSSP For Makespan

Year	Author(s)	Acronym	Performance be superior to
2003	Aldowaisan & Allahverdi	SA,SA-1,SA-2,GEN,GEN-1, GEN-2	GA,N-RAJ,RAJ
2005	Grabowski and 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 (2003) 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 proposed by Nawaz et al. (1983) for improving the performance of SA and GA. The computational experiments proved that the improved algorithm efficiently finds the solutions over famous RAJ (Rajendran, 1994) algorithm. The use of metaheuristic for NWFSSP was further attempted by Grabowski and Pempera (2005) 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. (2007) 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. (2008) 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 Neighborhood Descent (VND) algorithm was proposed for improving the solution quality. Swap and insert neighborhood structures were used as speed-up methods. Subsequently, Pan et al. (2008) 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 neighborhood searching using insertion based local search technique. Further, Pan et al. (2008) 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. (2009) 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 neighborhood 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 (2010) 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. (2008) 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 (2012) 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 neighborhood solution. Afterwards, decoding of the solutions to its respective feasible solutions was returned to the TS. Ding et al. (2015) 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 (2015) modeled 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.

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 (Pan et al., 2008), IIGA (Pan et al., 2008), HDE (Quian et al., 2009), HGA (Tseng and Lin, 2010), GA-VNS (Jarboui et al., 2008), TS/PSO (Samargandhi and ElMekkawy, 2012), TMIIG (Ding et al., 2015) and $MNE_{H1}/MNE_{H2}/LKH$ (Lin and Ying, 2015) methods are the state-of-the-art algorithms for solving NWFSSP for makespan criterion. Further, only TMIIG (Ding et al., 2015) and $MNE_{H1}/MNE_{H2}/LKH$ (Lin and Ying, 2015) 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. (2007) 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 (Pan et al., 2008; Quian et al., 2009; Tseng and Lin, 2010; Jarboui et al., 2008; Samargandhi and ElMekkawy, 2012) for problem instances up to 100 jobs. Recently, some of the researchers Ding et al. (2015) and Lin and Ying(2015) 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 et al. (2015) compared performance of the proposed algorithm with algorithms developed by researchers Pan et al. (2008), Qian et al. (2009), Tseng and Lin (2010), Jarboui et al. (2008), Samarghandi and ElMekkaw (2012) 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 (2015) and Ding et al.(2015) are used for comparing results produced by proposed algorithm. Taillard (1993) benchmark suite is used as test set for NWFSS makespan and also is used here for conducting computational experiments for evaluating the performance of proposed approach with state-of-art methods available in literature.

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 Taillard's benchmark suits and then compares the performance of the proposed PHPSOM with that of the best-so-far algorithms. Finally, section 6 provides 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 (Sapkal and Laha, 2013).

Let $\sigma = \{\sigma_1, \sigma_2, . . . , \sigma_n\}$ correspond to the sequence of 'n' jobs to be processed on 'm' machines, and $\delta(i,s)$ 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:
For i = 1,2,. . .,n and j = 1,2,. . .,m

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

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

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

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

Finally, 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 neighborhood 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.

5.   PARTICLE SWARM OPTIMIZATION (PSO)

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 (1995). Bewoor et al. (2016; 2017) compared the performance of PSO for solving NWFSSP problem with other metaheuristics viz. GA and TS and proved that PSO gives better solution than other metaheuristics. The particles are initialized with random positions and later explore the entire search space for finding better solution (Tasgetiren, 2007). 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 followed by particles. The learning by each particle from the best particle from entire swarm is observed in global PSO version, while learning from the best particle available in its neighbourhood is observed in the local version. 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 (6) and (7)

$$V_{new} = w*V_{curr} + c_1*r_1*(pbest-X_{curr}) + c_2*r_2*(gbest-X_{curr}) \quad (6)$$

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

where w is inertia weight which provides balance between local and global search capabilities. The acceleration constants c_1 and c_2 in (6) are cognitive parameter which develops the bird's own confidence (cognitive behavior) and confidence in swarm (social behavior) 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. X_{new} and V_{new} is new position and velocity of particle respectively. X_{curr} and V_{curr} is current position and velocity of particle respectively. In the standard PSO, new velocity of the particle is found by (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 equation (7). This process stops when user-defined terminating criterion is met.

5.1 Solution Representation

Solution representation is one of the most important issues while designing PSO algorithm. For representing the solution, earlier researchers (Wang and Zheng, 2003) 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 (Bean, 1994) 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 $\sigma_{ij}= [j_1, j_2, j_3, …j_n]$.To demonstrate the scheme of the ROV rule, we provide a simple example in Table 2.

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, $\sigma_{ij}=[4,1,2,5,3]$. In PHPSOM, job permutation based local search approaches are applied rather than direct consideration of the particle's position information. So, 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 (Wang and Zheng, 2003), lead to swapping of position value 4.22 and 4.37.

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

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

5.2 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 [Nawaz et al., 1983] 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 $\sigma_{ij}=[4,1,2,5,3]$ 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 σ_i with σ_{i+1} and find makespan

    3.2 if makespan $(\sigma_i) >$ makespan(σ_{i+1}) set new_seq=σ_i else new_seq=σ_{i+1}

Step 4: End for

Step 5: return new_seq

5.3 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 [Jarboui et al., 2008]. 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 neighborhood 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 equation (8).

$$P = e^{\frac{-\Delta E}{\alpha T}} \qquad (8)$$

where α is the Boltzmann constant and T is the current temperature ΔE 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.

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

5.4 Proposed Hybrid PSO Makespan (PHPSOM) for NWFSSP

PHPSOM algorithm is based on the solution representation by ROV rule, population initialization with NEH-based local search and neighborhood 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: Select particle for local refinement from present population

    repeat

    6.1 for *i* :=0 to n -1 do

      6.1.1 Particle's velocity and position is updated according to equation (10) and equation (11), respectively

      6.1.2 Update value of particle (particle_pval) and apply ROV rule to find next job permutation.

      6.1.3 Calculate makespan value for the updated particle.

      6.1.4 If updated_makespan_value > current_makespan_value and gbest (pgbest) then

      6.1.5 update pbest of particle (mpbest) and gbest (pgbest)

      end for

    until maximum iteration reached

Step 8: Select best_particle from the population for global refinement;

Step 9: Explore local serach space with SA based local search as per Algorithm 2.

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.

6. 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.

6.1 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 benchmark problem instances. The algorithm is tested for large scale problem instances of Taillard benchmark suit (1993). The 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), representing the number of jobs and machines, respectively.

6.2 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,

$$\text{ARPD} = \frac{100}{k} \sum_{i=1}^{k} \frac{Heuristic_i - BestH_i}{BestH_i} \quad (9)$$

where *Heuristic_i* is the makespan obtained by any of four algorithms and the *BestH_i* was the lowest makespan obtained for that specific instance. Table 4 displays comparative evaluation of PHPSOM with MNEH+LKH (Lin and Ying, 2015), TMIIG (Ding et al., 2015), HGA (Tseng and Lin, 2010) based on ARPD for Taillard benchmark data suite.

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

Instances	MNEH+ LKH	TMI IG	HGA	PHP SOM
ta001	0.22	0.22	0.22	0.00
ta002	0.23	0.23	0.21	0.00
ta003	0.26	0.26	0.23	0.00
ta004	0.18	0.18	0.17	0.00
ta005	0.16	0.16	0.15	0.00
ta006	0.21	0.21	0.20	0.00
ta007	0.23	0.23	0.24	0.00
ta008	0.15	0.15	0.15	0.00
ta009	0.13	0.13	0.10	0.00
ta010	0.16	0.16	0.16	0.00
ta011	0.36	0.36	0.33	0.00
ta012	0.29	0.29	0.29	0.00

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

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. So, it can be concluded that PHPSOM is more effective than IITG (Ding et al., 2015), MNEH and LKH (Lin and Ying, 2015) and HGA (Tseng and Lin, 2010) especially for large problem size.

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 Figure 1.
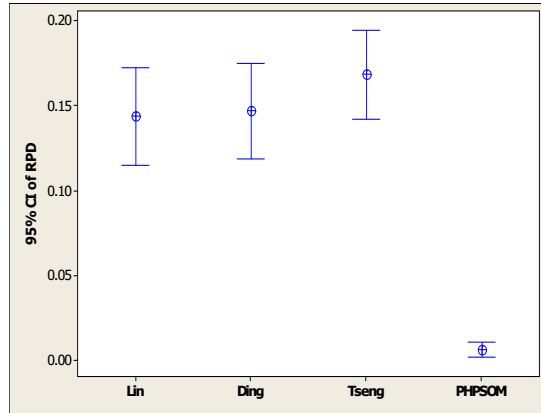

Figure 1: Means and 95% confidence level intervals for various algorithms for Taillard's dataset.

7. 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, 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) 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.

Author's Contributions

Laxmi A. Bewoor has designed and developed the algorithm under the supervision and guidance of Sagar U. Sapkal and V. Chandra Prakash. They explored the applicability of this algorithm to manufacturing scheduling problem in general and no wait flow shop in particular. Accordingly, Laxmi A. Bewoor has tested the algorithm and obtained the results which are incorporated in this paper. All of them contributed for writing this paper.

Ethics

Authors should address any ethical issues that may arise after the publication of this manuscript. There are no ethical issues that can be arisen after the publication of this work, as it is novel and covers materials that are not published before. The corresponding author endorses that the other co-author has acknowledged the contents of the manuscript.

REFERENCES

[1] Aldowaisan T. and Allahverdi,"New heuristics for no -wait flowshops to minimize makespan", *Comput Oper Res*, Vol.30, 2003, pp.1219–1231.

[2] Bean J.C., "Genetic algorithms and random keys for sequencing and optimization", *ORSA J Comput.*Vol.6, No.2, 1994, pp.154–160.

[3] Bewoor L. A. , Chandra Prakash V. and Sapkal S. U. "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*, Vol.8, No.6, 2016, pp.3021–3026.

[4] Bewoor L. , Chandra Prakash V. and Sapkal S. "Comparative Analysis of Metaheuristic approaches for Makespan Minimization for No Wait Flow Shop Scheduling Problem", *Int. J. Elec.*,Vol. 7, 2017,pp.31-37.

[5] Blum C.and Roli A. "Metaheuristics in Combinatorial Optimization: Overview and Conceptual Comparison", *ACM Computing Surveys*, Vol.35, No.3, 2003,pp.268–308.

[6] Blum C.; Puchinger J. Raidl G. R. and Roli A. "Hybrid metaheuristics in combinatorial optimization: A survey",Applied Soft Computing Journal,Vol.11, No.6, 2011, pp. 4135–4151.

[7] Ding J.Y.; Song S.; Gupta J.N.D.; Zhang R..; Chiong R. and Wu C, "An improved iterated greedy algorithm with a Tabu-based

reconstruction strategy for the no-wait flowshop scheduling problem”, Applied Soft Computing; vol.30, no.5, 2015, pp.604–613.

[8] Garey M.; and Johnson D.,“Computers and intractability a guide to the theory of NP – completeness”, 4th Ed. 1979, New York: Freeman.

[9] Grabowski J.and Pempera J. “Sequencing of jobs in some production system”, European Journal of Operational Research, vol. 125, no. 3, 2000, pp. 535–550.

[10] Grabowski J.and Pempera J.,“Some local search algorithms for no-wait flow-shop problem with makespan criterion”, Comput. Oper. Res.,vol.32, No.8, 2005, pp. 2197–2212.

[11] Graham R.L.“Optimization and Approximation in Deterministic Sequencing and Scheduling”, Annals of Discrete Mathematics, Vol.5,1979, pp.287–326.

[12] Jarboui B.; Ibrahim S.; Siarry P.; and Rebai A., “A combinatorial particle swarm optimization for solving permutation flowshop problems”, Computers and Industrial Engineering,vol. 54, No.3, 2008, pp. 526–538.

[13] Kennedy J.; and Eberhart R., “Particle swarm optimization”,Proceedings of ICNN’95–IEEE International Conference on Neural Networks, IEEE CS Press Perth WA, Australia, 1995,pp.1942–1948.

[14] Laha D.; Chakraborty U. K., “A constructive heuristic for minimizing makespan in no -wait flow shop scheduling”, International Journal of Advanced Manufacturing Technology, vol.41, No.1, 2009, pp.97–109.

[15] Lin S. and Ying K. “Optimization of makespan for no-wait flowshop scheduling problems using efficient metaheuristics”, Omega, vol.64, 2015, pp.116-125.

[16] Liu B., Wang L., and JinY. H., “An effective hybrid particle swarm optimization for no - wait flow shopscheduling”, International Journal of Advanced Manufacturing Technology, vol.31, 2007,pp.1001–1011.

[17] Nawaz M.; Enscore Jr. E.; Ham I. “A Heuristic Algorithm for the m- Machine n-Job Flow-shop Sequencing Problem”, The Int. J. of Mgmt Sci.,vol. 11, No.1, 1983, pp. 91–95.

[18] Pan Q.K.; Wang L.and Zhao B. H.“An improved iterated greedy algorithm for the no - wait flow shop scheduling problem with makespan criterion”,International Journal of Advanced Manufacturing Technology, vol.38, No.7, 2008, pp.778–786.

[19] Pan Q.K.,Tasgetiren M. F.and Liang Y.C. “A discrete particle swarm optimization algorithm for the no –wait flowshop scheduling problem”, Computers & Operations Research, vol.35, No.9, 2008,pp.2807–2839.

[20] Pan Q.K.,Liang Y.C.,Tasgetiren M.F. and Zhao B.H. “A hybrid discrete particle swarm optimization algorithm for the no-wait flow shop scheduling problem with makespan criterion”, International Journal of Advanced Manufacturing Technology,vol.38, No.3, 2008, pp.337–347.

[21] Pinedo M. “Scheduling: Theory Algorithms and Systems”, 2002, Prentice-Hall New Jersey.

[22] Qian B. Wang L., Hu R. Huang D.X. and Wang X., “A DE-based approach to no-wait flow-shop scheduling”,Computers & Industrial Engineering, vol.57, No.3, 2009, pp.787– 805.

[23] Raaymakers W. and Hoogeveen J.“ Scheduling multipurpose batch process industries with no-wait restrictions by simulated annealing”,Eur J Oper Res, vol.12, No.6, 2000, pp.131–151.

[24] Rajendran C., “A no-wait flowshop scheduling heuristic to minimize makespan”,J Oper Res Soc, 45, 1994, pp.472– 478.

[25] Evolutionary computing, Springer-Verlag Berlin. Heidelberg 115–130.

[26] Rock H., “The Three-Machine No-Wait Flow Shop is NP-Complete”, Journal of the Association for Computing Machinery,vol. 31, No.2, 1984, pp. 336-345.

[27] Samarghandi H and ElMekkawy T. Y., “A metaheuristic approach for solving the no-wait flowshop problem”, International Journal of Production Research, vol.50, No.24, 2012, pp. 7313–7326.

[28] Sapkal S. U. and Laha D. “A heuristic for no-wait flow shop scheduling”, International Journal of Advanced Manufacturing Technology, vol. 68, No. 5–8, 2013, pp.1327–1338.

[29] Schuster C. J. and Framinan J.M. “ Approximative procedure for no -wait job shop scheduling”, Oper Res Lett 31,2003, pp. 308–318.

[30] Taillard E., “ Benchmarks for basic scheduling problems”, European Journal Operational Research, Vol.64, No.2, 1993,pp.278–285.

[31] Tasgetiren M. F.; Liang Y. C., Sevkli M., and Gencyilmaz G. “A particle swarm optimization algorithm for makespan and total flowtime minimization in the permutation flowshop sequencing problem”,European

Journal of Operational Research, Vol.177, No.(3), 2007, pp. 1930–1947.

[32] Tseng L. Y; and Lin Y. T."A hybrid genetic algorithm for no -wait flow shop scheduling problem",International Journal of Production Economics, vol.128, No.1, 2010, pp. 144–152.

[33] Wang   L.and Zheng D. Z., " An effective hybrid heuristic for flow shop scheduling", International   Journal   of   Advanced Manufacturing Technology, Vol. 21, No.1, 2003, pp. 38–44.