algorithms

MDPI

Article

Evolutionary Hybrid Particle Swarm Optimization Algorithm for Solving NP-Hard No-Wait Flow Shop Scheduling Problems

Laxmi A. Bewoor [1,*], V. Chandra Prakash [1] and Sagar U. Sapkal [2]

[1] Department of Computer Science and Engineering, K. L. University, Andhra Pradesh, Guntur 522502, India; vchandrap@kluniversity.in

[2] Department of Mechanical Engineering, Walchand College of Engineering, Maharashtra, Sangli 416415, India; sagarus1201@gmail.com

* Correspondence: laxmiabewoor@gmail.com; Tel.: +91-976-653-1977

Received: 29 July 2017; Accepted: 19 October 2017; Published: 28 October 2017

Abstract: The no-wait flow shop is a flowshop in which the scheduling of jobs is continuous and simultaneous through all machines without waiting for any consecutive machines. The scheduling of a no-wait flow shop requires finding an appropriate sequence of jobs for scheduling, which in turn reduces total processing time. The classical brute force method for finding the probabilities of scheduling for improving the utilization of resources may become trapped in local optima, and this problem can hence be observed as a typical NP-hard combinatorial optimization problem that requires finding a near optimal solution with heuristic and metaheuristic techniques. This paper proposes an effective hybrid Particle Swarm Optimization (PSO) metaheuristic algorithm for solving no-wait flow shop scheduling problems with the objective of minimizing the total flow time of jobs. This Proposed Hybrid Particle Swarm Optimization (PHPSO) algorithm presents a solution by the random key representation rule for converting the continuous position information values of particles to a discrete job permutation. The proposed algorithm initializes population efficiently with the Nawaz-Enscore-Ham (NEH) heuristic technique and uses an evolutionary search guided by the mechanism of PSO, as well as simulated annealing based on a local neighborhood search to avoid getting stuck in local optima and to provide the appropriate balance of global exploration and local exploitation. Extensive computational experiments are carried out based on Taillard's benchmark suite. Computational results and comparisons with existing metaheuristics show that the PHPSO algorithm outperforms the existing methods in terms of quality search and robustness for the problem considered. The improvement in solution quality is confirmed by statistical tests of significance.

Keywords: NP-hard; no-wait flow shop; metaheuristic; scheduling; particle swarm optimization; simulated annealing; total flow time

1. Introduction

Scheduling is an integral part of advanced manufacturing systems. Production scheduling is the arrangement of jobs to be processed on available machines under some constraints. Flow Shop, Job Shop, and Open Shop are the classical models used to solve scheduling problems. The Flow Shop Scheduling Problem (FSSP) addresses most famous machine scheduling problems of many manufacturing systems, assembly lines, and information service facilities [1,2]. Sometimes Flow Shops have no delay situations that occur in the production environment in many real-life situations where a job must be processed continuously, without any interruption, from beginning to end, in order to follow the technological order of a process, which leads to a variant with the added constraint of "no-wait" [3]. In order to maintain continuous processing of a job in No-Wait Flow Shop Scheduling (NWFSS),

the start of a job by the first machine is delayed, if required, and the scheduling of such a "no-wait" constraint has attracted many researchers. A No-Wait Flow Shop Scheduling Problem (NWFSSP) has found applications in various processing industries, such as the chemical industry [4], food [5], concrete ware production [6], pharmaceuticals [7], etc. Allahverdi [8] reviewed scheduling problems with the no-wait constraint with respect to different shop environments, performance measures, setup types, and optimal scheduling criteria. Among various optimality criteria, viz. makespan, Total Flow Time (TFT), tardiness, lateness, number of tardy jobs, etc., makespan [9] and TFT [7,10] are of major interest for solving scheduling problems of no-wait type of flow shops, because makespan and TFT determine the total processing time for an entire pool of jobs and the total processing times for individual jobs respectively.

This paper addresses TFT as an objective function for solving NWFSSP. TFT is considered to be an important performance measure that, when optimized, reflects a stable or uniform utilization of resources, a rapid turn-around on jobs, and the minimization of in-process inventory [4]. The main objective of planning a production schedule is to discover the sequence of jobs, which minimizes TFT. The classical brute force method for finding such job sequences fails for large-sized problems, as computational complexity rises exponentially as n!, where "n" is number of jobs; thus, NWFSSP is treated as combinatorial optimization problem. Because of computational complexity, researchers [11–13] have concluded that NWFSSP with more than two machines is NP-hard. The solutions to solve such NP-hard problems consist of an approximate algorithm which uses constructive heuristics, local search methods, and metaheuristics. Generally, heuristic algorithms can obtain near-optimal solutions in an acceptable amount of time. Earlier researchers [14–18] have developed efficient constructive heuristic algorithms for TFT minimization; however, these attempts are not useful for identifying near optimal solutions for larger-sized problems, as these developed algorithms usually get trapped in local optima for large-problem sizes [15]. Local search methods can find the solutions, but the quality of a solution and computational time depends to a great extent on appropriate initial populations [19]. Due to the advent of computation techniques, metaheuristics can be used to solve problems in less time so that the limitation of computational complexity can be resolved through metaheuristic applications.

The field of metaheuristics, for application in combinatorial optimization problems of the scientific and industrial worlds, is growing rapidly [20]; on the other hand, attempts to use metaheuristics for solving combinatorial optimization problems began late. The recent past has witnessed a remarkable shift towards the hybridization of metaheuristics for optimization. The current trend focuses more on problem-specific approaches that lead to hybridization [21]. This paper attempts to use a metaheuristic technique, viz. Particle Swarm Optimization (PSO) algorithm, and its hybridization with Simulated Annealing (SA) to solve NWFSSP with a consideration of the Total Flow Time (TFT) of jobs as an objective criterion. Through extensive computational analysis using the well-known Taillard benchmark suite, we demonstrate that the Proposed Hybrid PSO (PHPSO) algorithm outperforms the recent best-performing algorithms available in the literature.

The remainder of this paper is organized as follows. Section 2 provides a comparative review of various metaheuristics for solving NWFSSP. Section 3 formally defines and formulates NWFSSP. Section 4 describes metaheuristic PSO and SA along with a detailed procedure for implementing the proposed metaheuristic PHPSO. Section 5 describes PHPSO on Taillard benchmark suites, and then compares the performance of the proposed metaheuristics with that of the best-so-far algorithms. Finally, concluding remarks are given in Section 6.

2. Literature Review

Various metaheuristics have been proposed for solving NWFSSP for different objective criteria. This section provides a comparative review of various metaheuristics, used by earlier researchers, for solving NWFSSP for TFT as an optimization criterion, along with various hybridization techniques for the improvement of results obtained via metaheuristics over the last decade.

Fink and Vob [3] applied different kinds of metaheuristics and constructive heuristics, such as nearest neighbor, cheapest insertion, and the pilot method, along with steepest descent (SD), Iterated Steepest Descent (ISD), Simulated Annealing (SA), and Tabu Search (TS), and examined tradeoffs between solution quality and running time. Implementation efforts showed that high-quality results were obtained in an efficient way by applying metaheuristics. Later, Gao et al. [22] proposed the Hybrid Harmony Search (HHS) algorithm using Nawaz-Enscore-Ham (NEH) [23] heuristics, and provided a solution with an appropriate balance between global exploration and local exploitation. Gao et al. [24] attempted to use the Enhanced Migrating Birds Algorithm (EMBO), based on neighborhood search heuristics, to avoid local optima. In order to improve the quality of solutions, Filho et al. [25] came up with a novel Evolutionary Clustering Search (ECS) metaheuristic approach, and found it to have better results than the method of Fink and Vob [3], Discrete Particle Swarm Optimization (DPSO) [9]. Genetic Algorithm (GA) as a metaheuristic technique was quite popular for solving optimization problems, and, later, it was observed that the solution quality was improved using a hybridization technique. Tseng and Lin [7] proposed the hybrid genetic algorithm and a novel local search scheme for improving solution qualities. Zang et al. [26] also proposed the Hybrid-Genetic Algorithm (HGA) using a new crossover operator, which helped them to argue that metaheuristics always yield better solutions than heuristics. Further, the Asynchronous Genetic Algorithm (AGA) proposed by Xu et al. [27] provided a solution for avoiding gene diversity in a short amount of time. Recently, Wang et al. [28] used a constraint-simplified mixed integer programming model and proposed the Hybridization of GA with the Neighborhood Search (H&NSGA). Although GA yields better solutions, it requires the appropriate tuning of parameters; thus, a new population-based methodology working on the principle of social behavior was introduced.

PSO is a population-based metaheuristic technique that is quite popular nowadays, because of its better solution quality achieved with less parameter tuning efforts. The comparative analysis of various metaheuristics for NWFSSP, by Bewoor et al. [29,30], advocated the effectiveness of PSO. Some remarkable metaheuristics were developed by Pan et al. [9,10] for solving NWFSSP. Pan et al. [9] developed Discrete Particle Swarm Optimization (DPSO) by considering both the makespan and total flow time minimization as optimization criteria for solving the no-wait flowshop scheduling problem. Akhshabi et al. [31] proposed a Hybrid Particle Swarm Optimization (HPSO) algorithm, based on the Memetic Algorithm (MA), and provided better solutions. On the basis of a detailed literature review, and a study of the status of research work related to use of metaheuristics for solving NWFSSP for TFT as optimization criteria, the following research gaps were identified:

- The published literature, thus far, has primarily addressed permutation-type flow shop scheduling problems, but fewer attempts were found in the study of the "no-wait" type variant of flow shop scheduling problems (NWFSSP).
- Earlier researchers used PSO, variants of PSO, DPSO, and the hybridization of PSO with MA for solving NWFSSP with TFT as an optimality criterion; however, the development of an efficient algorithm using the hybridization of PSO with SA and TFT as an optimality criterion for NWFSSP has not been reported, neither for small-sized jobs (n = 20, 50, and 100) nor for large-sized jobs (n = 200 and 500).
- Investigations done by most of the researchers for solving NWFSSP have been limited to 100/200 jobs [3,9,15–17]. Recently, Pan-Ruiz et al. [10] tried to solve large-sized problems up to 500 jobs for the Permutation Flow Shop Scheduling Problem (FSSP) and Akhshabi et al. [31] considered NWFSSP for solving large-sized problems up to 500 jobs. Hence, the scope for further research can be clearly sensed to develop improved metaheuristics.

This provided the impetus to study the hybridization of PSO with SA in order to solve NWFSSP optimally. To know the effectiveness and efficiency of the newly-proposed algorithm, the results produced by earlier researchers, Fink and Vob [3], DPSO [9], Pan and Ruiz [10], and HPSO [31], are used for comparison with the PHPSO algorithm.

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

No-wait Flowshop scheduling has set of "n" jobs and "m" machines. The processing time p(i, j) of every job "i" on each machine "j" is given. NWFSS has additional constraint of "no-wait", which meansthat once a job starts at the first machine, it will be processed entirely through all "m" machines without waiting in between and without any preemption. To meet this constraint, a job may be delayed at the beginning. So, in order to solve this type of problems, a delay matrix (δ) needs to be calculated [17].

Let $\sigma = \{\sigma_1, \sigma_2, \ldots, \sigma_n\}$ represent the sequence of "n" jobs to be processed on "m" machines, and δ(i, s) represent the minimum delay on the first machine between the start of job "i" and the start of job "s". Also, let $p(\sigma_i, j)$ represent the processing time on machine "j" of the job at the "i" position of a given sequence, and let $\delta(\sigma_{i-1}, \sigma_i)$ denote the minimum delay on the first machine between the start of two consecutive jobs found in the "(i − 1)" and "i" position of the sequence. Let $C(\sigma_i)$ be the completion time of the job in the ith position of a given sequence.

For i = 1, 2, . . . , n and j = 1, 2, . . . , m

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

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

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

The formula for total flow time (TFT) is given as:

$$TFT = \sum_{i=1}^{n} C(\sigma 1) \tag{4}$$

$$= \sum_{i=2}^{n} \{ \sum_{k=2}^{i} \delta(\sigma_{k-1}, \sigma_k) + \sum_{j=1}^{m} p(\sigma_i, j) \} + \sum_{j=1}^{m} p(\sigma_1, j) \tag{5}$$

$$= \sum_{i=2}^{n} (n + 1 - i)\ \delta\ (\sigma_{i-1}, \sigma_i) + \sum_{i=2}^{n} \sum_{j=1}^{m} p(i, j) \tag{6}$$

The delay matrix of size n × n provides all the δ(i,k) values between the start of any two consecutive jobs i and k, where $i \neq k$ in a given sequence of n jobs to determine the objective function value. The delay matrix δ(i,k) values are obtained from the following equation:

$$\delta(i, k) = p(i, 1) + \max\{ \sum_{h=2}^{r} p(i, h) - \sum_{h=1}^{m} p(k, h), 0 \} \qquad 2 \leq r \leq m \tag{7}$$

Given the matrix of size "n" (jobs) × "m" (machines) with processing time p(i,j), it is possible to generate (n!) number of feasible sequence of solutions, denoted as F(σ), from which the optimal sequence, denoted as F(σ*), is to be chosen and can be stated as per the equation given below:

$$F\ (\sigma^*) \leq F\ (\sigma) \tag{8}$$

The problem is to determine a sequence of "n" jobs which gives minimum total flow time (TFT).

4. Proposed Hybrid PSO (PHPSO) for NWFSSP

In this paper, we propose an extension of the PSO algorithm to solve NWFSSP. PHPSO essentially differs from the standard PSO in some characteristics. While designing PHPSO, suitable particles from the current population are selected and effective local search for the selected particles are carried out. Taillard benchmark suite [32] is used as the input dataset for validating results produced by PHPSO. The procedural steps for designing PHPSO are explained in detail in the subsequent sections.

4.1. Particle Swarm Optimization (PSO)

PSO is an optimization algorithm that simulates its behavior from the biological example of a flock of birds searching for food in a defined area [33]. Birds do not know where the food is, but they know at each time how far the food is, by following the nearest food strategy. PSO simulates this behavior and finds the best solution in the search space. Each 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, the particle reaches towards the approximate solution for the given objective function. The standard theory and procedure of PSO is well defined by Eberhard and Kennedy [34].

The algorithm is initialized with particles at random positions, and then it explores the search space to find a better solution [18]. For each iteration, each particle adjusts its velocity to follow two best solutions. The first is the cognitive part where a particle follows its own best solution found so far, called "*pbest*", and the other is the current best solution of swarm, called "*gbest*". On the basis of the different learning approaches of particles, PSO presents with two versions viz. the global version and the local version. In the global PSO, each particle learns from the best particle in the whole swarm, while in the local version each particle learns from the best particle in its neighborhood. Out of these two versions, the local PSO has a slower convergence speed; thus, it may adapt to a changing environment more easily which is followed exactly in the NWFSS. The new velocity is denoted by V_{new} and the new position is denoted by X_{new}, as stated in Equations (9) and (10):

$$V_{new} = w*V_{curr} + c_1*r_1*(pbest - X_{curr}) + c_2*r_2*(gbest - X_{curr}) \quad (9)$$

$$X_{new} = X_{curr} + V_{new} \quad (10)$$

where w is the inertia weight, which provides balance between local and global search capabilities. The acceleration constants c_1 and c_2 in Equation (9) are cognitive parameters which develop the bird's own confidence (cognitive behavior) and its confidence in the swarm (social behavior), respectively. Low values of c_1 and 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} are the new position and velocity of the particle, respectively. X_{curr} and V_{curr} are the current position and velocity of the particle, respectively. In the standard PSO, the new velocity of the particle is found by Equation (9), considering its previous velocity and the distance of its current position from both its own best historical position and its neighbors' best position. Generally, the value of each component in velocity is set to the range (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 (10). This process stops when the user-defined terminating criterion is met.

4.2. Solution Representation

Solution representation is one of the most important issues in designing a PSO algorithm. To represent the solution, a job-permutation-based encoding scheme [2] has been used very often by earlier researchers for solving NWFSSP. However, as the position of particles in PSO is a continuous character, a standard encoding scheme of PSO cannot be adopted directly for solving NWFSSP. PSO can

be effectively applied by considering dimension size as "n" for representing "n" jobs, and related particle information is represented as $X_i = \{x_1, x_2, x_3, \ldots, x_n\}$. As permutations of jobs cannot be presented with the particle alone, it is necessary to find suitable mapping between the job sequence and the position of particles in PSO. So, in this paper, the Ranked-Order-Value (ROV) rule based on the random key value [35,36] 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 one to convert the continuous nature of PSO algorithms to apply to the determination of the discrete nature of problems, such as sequencing, which in turn evaluates the performance of a particle. Moreover, permutations of jobs are constructed by considering a job index, which is the rank of each position value of a particle. The ROV rule used in PHPSO handles the particle with the smallest position value first and assigns a rank value i.e., 1, and that which is observed as the smallest is assigned to that position of the particle. In the case of two or more particles with same position values, the position with the smallest dimension number is given priority and assigned a rank value first. The remaining position values are incremented by 1 and subsequently assigned the next rank values as per the dimension number. Then, the second smallest position value will be handled in the same manner. Thus, the position information of a particle is converted to corresponding job permutation $\sigma_{ij} = [j_1, j_2, j_3, \ldots, j_n]$. To demonstrate the scheme of the ROV rule, we provide a simple example in Table 1.

Let us consider that 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 the smallest position value of the particle, $x_{1,2}$ is prioritized first by being assigned the rank value of 1. Next, two particles $x_{1,3}$ and $x_{1,5}$ have equal position value i.e., 4.37. Yet, index of $x_{1,3}$ is smaller as compared to $x_{1,5}$. So, $x_{1,3}$ is assigned the next rank value of 2 and $x_{1,5}$ is incremented to rank value 3. Finally, rank values 4 and 5 are respectively assigned to $x_{1,1}$ and $x_{1,4}$. Thus, the job permutation $\sigma_{ij} = [4, 1, 2, 5, 3]$ is obtained considering the position information value of each particle and the corresponding rank assignment based on the ROV rule. In the Proposed Hybrid PSO, job permutation based local search approaches are applied rather than a direct consideration of the particle's position information. So, it is necessary to convert the particle's position information to a corresponding job permutation as per the ROV rule when a local search is completed. Because of the simple mechanism of the ROV rule, adjustment for a new particle position is very easy. Local search methods using the position information are handled in the same way as the process adopted for job permutation. For example, in Table 2, if the SWAP operator [2] is used as a local search operator for job permutation; the swapping of job 2 and job 4 corresponds to the swap of position values 4.22 and 4.37.

Table 1. Representation of the solution of position information of each particle and the corresponding ROV for the corresponding job permutation.

Dimension	**1**	**2**	**3**	**4**	**5**
x_{ij}	5.45	4.22	4.37	5.47	4.37
Job Permutation	4	1	2	5	3

Table 2. Job permutations and the corresponding position information after swapping job2 and job4 for a swap-based local search.

Dimension	**1**	**2**	**3**	**4**	**5**
x_{ij}	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 often random in the standard PSO. An initial population with a certain quality and diversity provides an efficient solution. In this paper, we propose the NEH

heuristic technique [23] as an efficient population initialization procedure. In order to find NEH-based seed sequence, jobs are ordered in ascending sums of their total flow times. The partial schedules depending on the initial order are taken into account to construct a job sequence. Consider a current sequence σ_{ij} = [4, 1, 2, 5, 3]; if 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 which may appear as [1, 4, 2, 5, 3], [1, 2, 4, 5, 3], [1, 2, 5, 4, 3], and [1, 2, 5, 3, 4]. Among all these sequences, the sequence generating the minimum TFT is chosen as the current sequence for the next iteration. Thus, initial population generation with the NEH technique helps job permutation as compared to a random initial population.

4.4. Simulated Annealing(SA)

In metallurgy, the annealing process is the process where metals are cooled slowly to reach a state of low energy where they are very strong [37]. At high temperatures, the movements are random, whereas at low temperatures, little randomness is observed. Khamlichi et al. [38] used SA as a local search method for finding neighborhoods for optimizing the number of sensors and their positions in order to achieve the desired application requirements. Here, SA is used for possible job sequences leading towards minimum TFT in the context of NWFSS. SA starts a random search at a high temperature, and eventually the temperature is reduced slowly, becoming a greedy descent as it approaches to zero degrees. 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 worse initially at high temperatures, but improvements can be observed gradually at lower temperatures. For the minimization of a given objective function, temperature should be reduced according the probability (P) given by the Boltzmann factor given in Equation (11):

$$P = e^{-\Delta E / \alpha T} \tag{11}$$

where α is the Boltzmann constant, T is the current temperature, and ΔE is the 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. A proper initial temperature should be maintained high so that all states of the system have an equal probability of being visited.

4.5. Proposed Hybrid PSO (PHPSO) Algorithm

PHPSO algorithm is based on the solution representation by the ROV rule, population initialization with NEH-based local search and neighborhood searching through SA-based local search. The complete computational procedure of the PHPSO framework for the NWFSS PHPSO algorithm for NWFSSP (Algorithm 1) can be summarized as follows:

Algorithm 1: PHPSO for NWFSSP

Step 1: Input the total no. of jobs (n), total no. of machines (m) and processing time matrix (p).
Calculate delay matrix (δ) as per Equation (7).
Step 2:
for i = 0 to n − 1 do

2.1 Initialize particle iwith random value (mpval) and velocity (mvelocity). Set the acceleration constants c_1 and to 1.65 and 1.75 respectively; r_1 and r_2 both are set to the value 0.5 and inertia weight w to 0.65.
2.2 Apply ROV rule to represent random value of particle to position of particle (mpbest).
2.3 Calculate the processing sequence of job (σ) as per ROV rule.
2.4 Evaluate objective function value TFT as per Equation (6).

end for
Step 3:
Sort the particles with increasing order of TFT score.
Step 4: Generate initial seed sequence with NEH algorithm by following:

4.1 Consider the first sequence(σ_1) of job and find TFT. Swap first position with next and compute TFT for new sequence(σ_1).

4.2 for i = 1 to n do

4.2.1 Swap σ_i with σ_{i+1} and find TFT

4.2.2 if TFT(σ_i) < TFT(σ_{i+1}) set $f_{seq} = \sigma_i$

else $f_{seq} = \sigma_{i+1}$
Step 5: minArr = f_{seq}
Step 6: Calculate pbest (mpbest) of particle and gbest (pgbest) of swarm for generating the initial seed sequence.
Step 7: Select particle from the current population for local refinement;
repeat

7.1 for i = 0 to n − 1 do

7.1.1 Update velocity and position of particle according to Equations (9) and (10), respectively.

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

7.1.3 Calculate TFT value for the updated particle.

7.1.4 If updated_TFT_value > current_TFT_value and gbest (pgbest) then

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

end for
until maximum iteration count is reached.
Step 8: Select best_particle from the population for global refinement;
Step 9: Initialize initial_temperature, T as 3.0 and final_temperature, F as 0.9, and cooling rate α as 0.99.
Step 10: Initialize Best_So_Far to current state.
Step 11: while T < final_temperature do

11.1 for i = 0 to n − 1

11.1.1 Randomly perturb from the current state to a new state and calculate corresponding objective function value.

11.1.2 Update gbest depending on best particle.

11.1.3 Calculate the difference in objective function value between current and new state i.e., ΔE

11.1.4 If $\Delta E < 0$ i.e., new state has minimum TFT, accept new state as current state. Set Best_So_Far to this new state

11.1.5 If $\Delta E \geq 0$, consider new state as current state with probability by invoking random number between range (0, 1).

11.1.6 Prob (accepted) = exp $(-\Delta E/\alpha \cdot T)$.

11.1.7 Revise T as necessary according to annealing schedule

end for
end while
Step 12: set gbest to Best_So_Far.
End Procedure

Thus, it can be observed that the PHPSO effectively provides a promising solution within the entire region, along with exploitation for solution improvement in sub-regions. Because of the NP-hard nature of NWFSSP, PHPSO applies local search methods which include NEH-based local search and SA-based local search. Since both exploration and exploitation are used in this algorithm, it is expected to achieve good results for NWFSSP. The results obtained through various numerical simulations and their comparisons are demonstrated in the next section.

5. Numerical Tests and Comparisons

5.1. Experimental Setup

To test the performance of the PHPSO algorithm, a computational simulation is carried out with some well-studied benchmarks. In this paper, 120 problem instances that were contributed by the Taillard dataset are selected. The Taillard benchmark dataset is composed of 12 groups containing the problems of size ranging from 20 jobs and five 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. In this paper, we used this dataset to test our PHPSO algorithm, and this test bed is treated for NWFSSP with TFT as the optimization criterion. PHPSO is coded in Java and run on Intel Core i5, 8 GB RAM, 2.20 GHz PC.

5.2. Computational and Statistical Evaluation

To compare the proposed heuristic with the existing heuristics, we carried out the experimentation by running each instance independently 10 times; for each replication we used "Average Relative Percentage Deviation" (ARPD) as a performance measure, which is popular in the scheduling literature [9,10,14,16,17]. ARPD is given by:

$$ARPD = \frac{100}{k}\sum_{i=1}^{k}\frac{(HeuristiCi - BestHi)}{BestHi} \tag{12}$$

where $HeuristiC_i$ is the total flowtime obtained by any of four algorithms, and the $BestH_i$ is the lowest total flowtime obtained for that specific instance. Table 3 displays a comparative evaluation of the proposed metaheuristic, F&V [3], DPSO [9], Pan-Ruiz [10], and HPSO [31] based on ARPD for the Taillard benchmark data suite for 500 jobs.

Table 3. Comparison of performance of the existing metaheuristics and PHPSO.

Instances	F&V	DPSO	PAN-RUIZ	HPSO	PHPSO	Instances	F&V	DPSO	PAN-RUIZ	HPSO	PHPSO
ta001	0.6602	0.6602	0.4864	0.4864	0	ta062	0.3453	0.3243	0.0779	0.0779	0
ta002	0.8378	0.8378	0.6142	0.6142	0	ta063	0.341	0.3253	0.0802	0.0802	0
ta003	0.3002	0.3002	0.0931	0.0931	0	ta064	0.3614	0.353	0.112	0.112	0
ta004	0.5711	0.5711	0.3505	0.3505	0	ta065	0.2568	0.2431	0.0316	0.0316	0
ta005	0.6642	0.6642	0.4699	0.4699	0	ta066	0.2517	0.2355	0	0	0.6682
ta006	0.8226	0.8226	0.543	0.543	0	ta067	0.41	0.399	0.1294	0.1294	0
ta007	0.7412	0.7412	0.5032	0.5032	0	ta068	0.2638	0.246	0	0	0.578
ta008	0.2086	0.2086	0.0566	0.0566	0	ta069	0.2751	0.2606	0.0276	0.0276	0
ta009	0.6369	0.6369	0.4281	0.4284	0	ta070	0.2429	0.2272	0	0	0.5659
ta010	0.2729	0.2729	0.0747	0.0747	0	ta071	0.5933	0.5818	0.152	0.1519	0
ta011	0.449	0.449	0.2021	0.2021	0	ta072	0.6913	0.6736	0.1762	0.1762	0
ta012	1.4844	1.4844	1.1164	1.1164	0	ta073	0.6287	0.6143	0.1562	0.1562	0
ta013	1.4634	1.4634	1.1326	1.1326	0	ta074	0.6034	0.586	0.143	0.143	0
ta014	0.4945	0.4945	0.2571	0.2571	0	ta075	0.7386	0.725	0.2368	0.2368	0
ta015	1.2817	1.2817	0.8373	0.8373	0	ta076	0.5632	0.5474	0.0772	0.0772	0
ta016	1.7866	1.7866	1.4364	1.4364	0	ta077	0.4528	0.4413	0.03	0.03	0
ta017	1.8615	1.8615	1.3951	1.3951	0	ta078	0.7447	0.7325	0.2625	0.2625	0
ta018	0.9409	0.9409	0.6262	0.6262	0	ta079	0.6651	0.6526	0.2063	0.2063	0
ta019	0.6716	0.6716	0.446	0.446	0	ta080	0.6151	0.5987	0.1464	0.1464	0
ta020	1.1856	1.1856	0.8944	0.8944	0	ta081	1.2881	1.272	0.4875	0.4875	0
ta021	3.459	3.459	2.8844	2.8844	0	ta082	1.1395	1.1295	0.413	0.413	0
ta022	3.0842	3.0842	2.4337	2.4337	0	ta083	1.2602	1.2526	0.487	0.487	0
ta023	2.7716	2.7716	2.3392	2.3392	0	ta084	1.3522	1.3387	0.5561	0.5561	0
ta024	2.4208	2.4208	1.7912	1.7912	0	ta085	1.0442	1.0261	0.3556	0.3556	0

Table 3. *Cont.*

Instances	F&V	DPSO	PAN-RUIZ	HPSO	PHPSO	Instances	F&V	DPSO	PAN-RUIZ	HPSO	PHPSO
ta025	1.2228	1.2228	0.9689	0.9689	0	ta086	1.1755	1.1604	0.4352	0.4352	0
ta026	2.9389	2.9389	2.3263	2.3263	0	ta087	1.2491	1.2396	0.4628	0.4628	0
ta027	1.5579	1.5579	1.1232	1.1232	0	ta088	0.5035	0.4935	0	0	0
ta028	3.1438	3.1438	2.63	2.63	0	Ta089	1.0852	1.0701	0.3824	0.3824	0
ta029	3.0663	3.0663	2.4829	2.4829	0	ta090	0.3783	0.5	0	0	0.4575
ta030	2.6776	2.6776	2.128	2.128	0	ta091	0.6029	0.5761	0.1025	0.1025	0
ta031	0.5309	0.5242	0.305	0.305	0	ta092	0.4659	0.428	0	0	0.7522
ta032	0.3904	0.3816	0.1345	0.1345	0	ta093	0.4476	0.4266	0	0	0.7405
ta033	0.3633	0.3602	0.1	0.1	0	ta094	0.4936	0.4664	0.0334	0.0334	0
ta034	0.3682	0.3638	0.1283	0.1283	0	ta095	0.4635	0.4351	0	0	0.7278
ta035	0.432	0.425	0.1837	0.1837	0	ta096	0.499	0.4649	0	0	0.7416
ta036	0.4044	0.4014	0.16	0.16	0	ta097	0.4638	0.4366	0	0	0.5638
ta037	0.3379	0.3347	0.125	0.125	0	ta098	0.5586	0.523	0.0621	0.0621	0
ta038	0.3947	0.391	0.1357	0.1357	0	ta099	0.5659	0.5388	0.066	0.066	0
ta039	0.3935	0.3932	0.1572	0.1572	0	ta100	0.5911	0.558	0.0779	0.0779	0
ta040	0.4571	0.4501	0.1953	0.1953	0	ta101	0.9873	0.9636	0.2122	0.2122	0
ta041	0.3132	0.3092	0	0	0.1271	ta102	0.3667	0.6266	0	0	0.7785
ta042	1.1402	1.1346	0.5724	0.5724	0	ta103	0.6347	0.6084	0.0124	0	0
ta043	0.636	0.6346	0.2403	0.2403	0	ta104	1.0478	1.0169	0.2425	0.2425	0
ta044	0.9319	0.929	0.4709	0.4709	0	ta105	0.6518	0.6373	0	0	0.7306
ta045	1.1979	1.1951	0.6446	0.6446	0	ta106	1.0966	1.0666	0.2576	0.2576	0
ta046	0.8167	0.8137	0.3965	0.3965	0	ta107	0.645	0.6233	0	0	0
ta047	0.3134	0.3142	0	0	0.1592	ta108	0.9611	0.9464	0.1955	0.1852	0
ta048	0.9951	0.9953	0.5039	0.5039	0	ta109	1.0853	1.0552	0.2642	0.2642	0
ta049	0.9334	0.9314	0.4952	0.4952	0	ta110	0.6368	0.6159	0	0	0
ta050	0.8461	0.8455	0.4318	0.4318	0	ta111	-	-	0.07	0.07	0
ta051	1.7498	1.752	1.0019	1.0019	0	ta112	-	-	0.05	0.05	0
ta052	0.3752	0.3731	0.018	0.018	0	ta113	-	-	0.05	0.05	0
ta053	1.592	1.5903	0.8841	0.8841	0	ta114	-	-	0.05	0.05	0
ta054	1.6893	1.6915	1.0019	1.0019	0	ta115	-	-	0.07	0.07	0
ta056	1.2087	1.2087	0.6463	0.6463	0	ta116	-	-	0.06	0.06	0
ta057	0.3609	0.3596	0	0	0.0642	ta117	-	-	0.02	0.02	0
ta058	1.8276	1.823	1.0604	1.0604	0	ta118	-	-	0.11	0.1	0
ta059	0.4815	0.4807	0.0923	0.0923	0	ta119	-	-	0	0	0.99
ta060	0.3934	0.3934	0.0257	0.0257	0	ta120	-	-	0	0	1.09
ta061	0.3236	0.3051	0.0882	0.0882	0	*Avg.*	0.9011	0.8955	0.4244	0.4241	0.0999

Table 3 exhibits the results showing that the ARPD of PHPSO is significantly less compared to existing algorithms. It is also observed that, with respect to ARPD, the proposed method performs better than either of the existing methods for 103 problems out of the 120 under consideration.

To validate the significance of the proposed algorithm statistically, the results of PHPSO are compared with the results obtained by earlier developed metaheuristics viz. F&V (2003), DPSO (2008), Pan-Ruiz (2012), and HPSO (2014). To test the performance of the proposed algorithm and the best-known solutions of earlier algorithms published in the literature, a series of the paired *t*-test at the 95% significance level was carried out by Devore [39]. Paired *t*-test analyzes the differences in two observations of the mean of the results of PHPSO and the mean of existing metaheuristics. Let $\mu_D = \mu_1 - \mu_2$ denote the true difference between the ARPD generated by two different algorithms. The null hypothesis is given by H_0: $\mu_D = \mu_1 - \mu_2 = 0$, saying that there is no difference between the ARPD generated by two algorithms when compared. The alternative hypothesis is given by H_1: $\mu_D = \mu_1 - \mu_2 \neq 0$, saying that there is a difference between the ARPD generated by two algorithms when compared. The paired *t*-test results on the Taillard instances are shown in Tables 4–9.

The *p*-value is zero. Thus, the null hypothesis was rejected on behalf of the PHPSO algorithm. This indicates that the difference between TFTs generated using both algorithms are meaningful at the confidence interval (CI) of 95%. For this reason, it can be concluded that the PHPSO algorithm is superior to F&V, DPSO, Pan-Ruiz, and HPSO.

In addition to the pair-wise comparison of the metaheuristics, to observe the statistical significance of the differences between the heuristics, the means of each metaheuristic and the corresponding 95% confidence intervals are plotted in Figures 1 and 2.

Table 4. Paired *t*-test for H_0 = PHPSO = HPSO vs. H_1 = PHPSO ≠ HPSO on the best-known solutions.

Algorithm	N	Mean	StDev	SE Mean
PHPSO	110	282,949	380,236	36,254
HPSO-2014	110	319,512	405,337	38,647
Difference	110	−36,563.3	60,259.7	5745.5

95% CI for mean difference: (−47,950.8, −25,175.9); *t*-test of mean difference = 0 (vs. not = 0): *t*-value = −6.36, *p*-value = 0.000.

Table 5. Paired *t*-test for H_0 = PHPSO = Pan-Ruiz vs. H_1 = PHPSO ≠ Pan-Ruiz on the best-known solutions.

Algorithm	N	Mean	StDev	SE Mean
PHPSO	110	282,949	380,236	36,254
Pan+Ruiz-2012	110	319,750	405,889	38,700
Difference	110	−36,801.5	60,548.4	5773.1

95% CI for mean difference: (−48,243.5, −25,359.5); *t*-test of mean difference = 0 (vs. not = 0): *t*-value = −6.37, *p*-value = 0.000.

Table 6. Paired *t*-test for H_0 = PHPSO = DPSO vs. H_1= PHPSO ≠ DPSO on the best-known solutions

Algorithm	N	Mean	StDev	SE Mean
PHPSO	110	282,949	380,236	36,254
DPSO-2008	110	472,275	637,377	60,771
Difference	110	−189,326	272,438	25,976

95% CI for mean difference: (−240,810, −137,843); *t*-test of mean difference = 0 (vs. not = 0): *t*-value = −7.29, *p*-Value = 0.000.

Table 7. Paired *t*-test for H_0 = PHPSO = F&V vs. H_1 = PHPSO ≠ F&V on the best-known solutions.

Algorithm	N	Mean	StDev	SE Mean
PHPSO	110	282,949	380,236	36,254
F&V-2003	110	478,400	647,517	61,738
Difference	110	−195,451	267,281	25,484

95% CI for mean difference: (−248, 238, −141, 755); *t*-test of mean difference = 0 (vs. not = 0): *t*-value = −7.26, *p*-Value = 0.000.

Table 8. Paired *t*-test for H_0 = PHPSO = HPSO vs. H_1 = PHPSO ≠ HPSO on the best-known solutions.

Algorithm	N	Mean	StDev	SE Mean
PHPSO	120	795,657	1,746,744	159,455
HPSO-2014	120	853,494	1,820,273	166,167
Difference	120	−57,837.6	106,077.3	9683.5

95% CI for mean difference: (−77,011.8, −38,663.3); *t*-test of mean difference = 0 (vs. not = 0): *t*-value = −5.97, *p*-Value = 0.000.

Table 9. Paired *t*-test for H_0 = PHPSO = Pan-Ruiz vs. H_1 = PHPSO ≠ Pan-Ruiz on the best-known solutions.

Algorithm	N	Mean	StDev	SE Mean
PHPSO	120	795,657	1,746,744	159,455
Pan+Ruiz-2012	120	854,696	1,823,535	166,465
Difference	120	−59,039.3	109,396.3	9986.5

95% CI for mean difference: (−78, 813.5, −39,265.1); *t*-test of mean difference = 0 (vs. not = 0): *t*-value = −5.91, *p*-Value = 0.000.

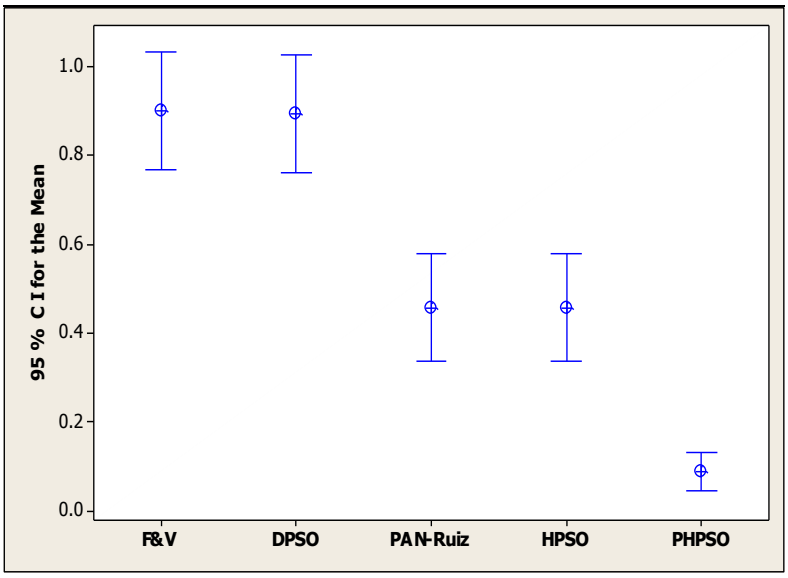

Figure 1. Means and 95% confidence intervals for different algorithms for ta001 to ta110.

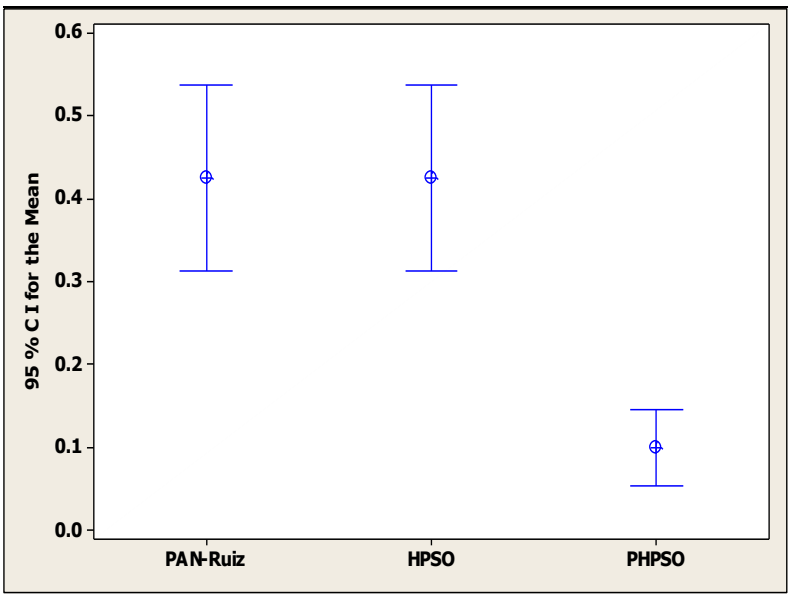

Figure 2. Means and 95% confidence intervals for different algorithms for ta001 to ta120.

5.3. Comparison of Proposed Hybrid PSO(PHPSO) with Fink and Vob, DPSOVND, Pan-Ruiz, and HPSO

We report the best-known solutions termed as objective function values found so far for NWFSSP with TFT criterion for Taillard's benchmark suite in Table 10. First, we carried out a simulation for the effectiveness of the PHPSO algorithm, and later, we compared our PHPSO with four existing metaheuristics viz. HPSO, which is an MA-based PSO by Akhshabi [31]; PRA, which is a local search-based algorithm by Pan and Ruiz [10]; the hybrid PSO based on the variable neighborhood search (DPSO) by Pan et al. [8]; and the F&V algorithm by Fink and Vob [3]. The best solution for each of the 120 Taillard dataset available in Operation Research (OR) library [32] was considered by closely examining all existing results. They are applied to these 120 benchmark instances ranging from ta001 to ta120 (i.e., for 20–500 jobs).

Table 10. New objective function values for Taillard's benchmarks treated as NWFSS with TFT criterion.

Instance	F&V	DPSO	Pan+Ruiz	HPSO	PHPSO	Instance	F&V	DPSO	Pan+Ruiz	HPSO	PHPSO
ta001	15,674	15,674	14,033	14,033	**10,841**	ta061	308,052	303,750	253,266	253,266	**232,745**
ta002	17,250	17,250	15,151	15,151	**11,386**	ta062	302,386	297,672	242,281	242,281	**224,780**
ta003	15,821	15,821	13,301	13,301	**12,168**	ta063	295,239	291,782	237,832	237,832	**220,164**
ta004	17,970	17,970	15,447	15,447	**11,438**	ta064	278,811	277,093	227,738	227,738	**204,798**
ta005	15,317	15,317	13,529	13,529	**10,204**	ta065	292,757	289,554	240,301	240,301	**232,933**
ta006	15,501	15,501	13,123	13,123	**11,505**	ta066	290,819	287,055	**232,342**	232,342	232,342
ta007	15,693	15,693	**13,548**	13,548	**13,548**	ta067	300,068	297,731	240,366	240,366	**212,821**
ta008	15,955	15,955	13,948	13,948	**11,394**	ta068	291,859	287,754	**230,945**	230,945	230,945
ta009	16,385	16,385	14,295	14,298	**12,010**	ta069	307,650	304,131	247,921	247,921	**241,266**
ta010	15,329	15,329	**12,943**	12,943	**12,943**	ta070	301,942	298,119	**242,933**	242,933	242,933
ta011	25,205	25,205	20,911	20,911	**17,395**	ta071	412,700	409,715	298,385	298,358	**259,015**
ta012	26,342	26,342	22,440	22,440	**20,603**	ta072	394,562	390,417	274,384	274,384	**233,285**
ta013	22,910	22,910	19,833	19,833	**15,300**	ta073	405,878	402,274	288,114	288,114	**249,201**
ta014	22,243	22,243	18,710	18,710	**14,883**	ta074	422,301	417,733	301,044	301,044	**263,386**
ta015	23,150	23,150	18,641	18,641	**16,146**	ta075	400,175	397,049	284,681	284,681	**230,167**
ta016	22,011	22,011	19,245	19,245	**17,899**	ta076	391,359	387,398	269,686	269,686	**250,354**
ta017	21,939	21,939	18,363	18,363	**17,667**	ta077	394,179	391,057	279,463	279,463	**271,318**
ta018	24,158	24,158	20,241	20,241	**19,447**	ta078	402,025	399,214	290,908	290,908	**230,425**
ta019	23,501	23,501	20,330	20,330	**20,059**	ta079	416,833	413,701	301,970	301,970	**250,337**
ta020	24,597	24,597	21,320	21,320	**21,254**	ta080	410,372	406,206	291,283	291,283	**254,082**
ta021	38,597	38,597	33,623	33,623	**29,656**	ta081	562,150	558,199	365,463	365,463	**245,683**
ta022	37,571	37,571	31,587	31,587	**29,199**	ta082	563,923	561,305	372,449	372,449	**263,582**
ta023	38,312	38,312	33,920	33,920	**30,158**	ta083	562,404	560,530	370,027	370,027	**248,834**
ta024	38,802	38,802	31,661	31,661	**31,343**	ta084	562,918	559,690	372,393	372,393	**239,313**
ta025	39,012	39,012	34,557	34,557	**29,551**	ta085	556,311	551,388	368,915	368,915	**272,137**
ta026	38,562	38,562	32,564	32,564	**29,790**	ta086	562,253	558,356	370,908	370,908	**258,445**
ta027	39,663	39,663	32,922	32,922	**25,506**	ta087	574,102	571,680	373,408	373,408	**255,264**
ta028	37,000	37,000	32,412	32,412	**28,929**	ta088	578,119	574,269	**384,525**	384,525	**384,525**
ta029	39,228	39,228	33,600	33,600	**29,647**	ta089	564,803	560,710	374,423	374,423	**270,858**
ta030	37,931	37,931	32,262	32,262	**29,314**	ta090	522,798	568,927	**379,296**	379,296	379,296
ta031	76,016	75,682	64,802	64,802	**49,655**	ta091	1,521,201	1,495,730	1,046,314	1,046,314	**949,025**
ta032	83,403	82,874	68,051	68,051	**59,984**	ta092	1,516,009	1,476,863	**1,034,195**	1,034,195	1,034,195

Table 10. *Cont.*

Instance	F&V	DPSO	Pan+Ruiz	HPSO	PHPSO	Instance	F&V	DPSO	Pan+Ruiz	HPSO	PHPSO
ta033	78,282	78,103	63,162	63,162	**57,420**	ta093	1,515,535	1,493,502	**1,046,902**	1,046,902	1,046,902
ta034	82,737	82,467	68,226	68,226	**60,470**	ta094	1,489,457	1,462,300	1,030,481	1,030,481	**997,214**
ta035	83,901	83,493	69,351	69,351	**58,590**	ta095	1,513,281	1,483,894	**1,034,027**	1,034,027	1,034,027
ta036	80,924	80,749	66,841	66,841	**57,620**	ta096	1,508,331	1,474,000	**1,006,195**	1,006,195	1,006,195
ta037	78,791	78,604	66,253	66,253	**58,893**	ta097	1,541,419	1,512,861	**1,053,051**	1,053,051	1,053,051
ta038	79,007	78,796	64,332	64,332	**56,646**	ta098	1,533,397	1,498,330	1,044,875	1,044,875	**983,816**
ta039	75,842	75,825	62,981	62,981	**54,424**	ta099	1,507,422	1,481,283	1,026,137	1,026,137	**962,641**
ta040	83,829	83,430	68,770	68,770	**57,533**	ta100	1,520,800	1,489,218	1,030,299	1,030,299	**955,843**
ta041	114,398	114,051	**87,114**	87,114	87,114	ta101	2,012,785	1,988,772	1,227,733	1,227,733	**949,025**
ta042	112,725	112,427	**82,820**	82,820	**82,820**	ta102	2,057,409	2,025,561	**1,245,271**	1,245,271	1,245,271
ta043	105,433	105,345	79,931	79,931	**64,446**	ta103	2,050,169	2,017,216	1,269,673	1,254,162	**1,254,162**
ta044	113,540	113,367	86,446	86,446	**68,770**	ta104	2,040,946	2,010,121	1,238,349	1,238,349	**1,238,349**
ta045	115,441	115,295	86,377	86,377	**62,523**	ta105	2,027,138	2,009,299	**1,227,214**	1,227,214	1,227,214
ta046	112,645	112,459	86,587	86,587	**62,005**	ta106	2,046,542	2,017,240	1,227,604	1,227,604	**976,118**
ta047	116,560	116,631	**88,750**	88,750	88,750	ta107	2,045,906	2,018,945	**1,243,707**	1,243,707	1,243,707
ta048	115,056	115,065	86,727	86,727	**67,669**	ta108	2,044,218	2,028,861	1,246,123	1,235,460	**983,816**
ta049	110,482	110,367	85,441	85,441	**67,144**	ta109	2,037,040	2,007,678	1,234,936	1,234,936	**962,641**
ta050	113,462	113,427	87,998	87,998	**61,460**	ta110	2,046,966	2,020,806	1,250,596	1,250,596	**955,843**
ta051	172,845	172,981	125,831	125,831	**62,857**	ta111	–	–	6,698,656	6,698,656	**6,263,859**
ta052	161,092	160,836	119,247	119,247	**117,137**	ta112	–	–	6,770,735	6,723,548	**6,413,646**
ta053	160,213	160,104	116,459	116,459	**101,810**	ta113	–	–	6,739,645	6,739,645	**6,437,528**
ta054	161,557	161,690	120,261	120,261	**100,074**	ta114	–	–	6,785,991	6,743,598	**6,432,538**
ta055	167,640	167,336	118,184	118,184	**114,468**	ta115	–	–	6,729,468	6,729,468	**6,312,830**
ta056	161,784	161,784	120,586	120,586	**103,248**	ta116	–	–	6,724,085	6,724,085	**6,361,035**
ta057	167,233	167,064	**122,880**	122,880	**122,880**	ta117	–	–	6,691,468	6,691,468	**6,539,854**
ta058	168,100	167,822	122,489	122,489	**119,449**	ta118	–	–	6,783,916	6,755,489	**6,126,127**
ta059	165,292	165,207	121,872	121,872	**111,571**	ta119	–	–	**6,711,305**	6,711,305	6,711,305
ta060	168,386	168,386	123,954	123,954	**120,849**	ta120	–	–	**6,755,722**	6,755,722	6,755,722

Bold values indicate improvement over existing metaheuristics.

Table 10 shows that the proposed algorithm improves 103 out of 120 instances of the Taillard dataset. This shows that the optimal results obtained by the PHPSO are better than values obtained by various metaheuristics to date, exhibiting the effective searching quality of PHPSO.

Compared with the results by the F & V and DPSO methods, PHPSO could improve the results to a great extent, which demonstrates the noteworthy improvement by PHPSO over F & V and $DPSO_{VND}$ metaheuristics. Values obtained by PHPSO are better than those obtained by HPSO for almost all of the instances. So, it is concluded that our proposed NEH- and SA-based local search methods, especially their utilizations in a hybrid sense, are more effective than the variable neighborhood-based [9] and MA-based local search methods [31], especially for large-sized problems.

6. Conclusions and Future Research

In this paper, we proposed a hybridization of PSO with SA for flow shop scheduling with a no-wait constraint. The PHPSO algorithm not only applies an evolutionary search guided by the mechanism of PSO, but also it applies a local search guided by the NEH-based initial population and the mechanism of SA. Thus, both global exploration and local exploitation are balanced. The results and comparisons of the simulation demonstrate the supremacy of PHPSO in terms of searching quality and robustness of solution.

The effectiveness of the proposed method was measured by using ARPD, which is a widely used performance measure. We carried out an extensive experimental and statistical analysis and found that PHPSO has improved objective function values for 103 out of the 120 best-known solutions for Taillard's benchmark suite. After comparing the solutions obtained through PHPSO with the solutions provided by other algorithms reported in the literature (viz. HPSO, Pan-Ruiz, DPSO, F & V algorithms), it is clearly seen that the PHPSO algorithm outperforms the existing algorithms. Hence, to the best of our knowledge, it is concluded that the PHPSO algorithm is the improved hybrid algorithm for the application of PSO to NWFSSP with a TFT criterion.

Author 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.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Pinedo, M. *Scheduling: Theory, Algorithms and Systems*, 2nd ed.; Prentice-Hall: Upper Saddle River, NJ, USA, 2002.
2. Wang, L.; Zheng, D.Z. An effective hybrid heuristic for flow shop scheduling. *Int. J. Adv. Manuf. Technol.* **2003**, *21*, 38–44.
3. Fink, A.; Vob, S. Solving the continuous flow-shop scheduling problem by metaheuristics. *Eur. J. Oper. Res.* **2003**, *151*, 400–414. [CrossRef]
4. Rajendran, C. A no-wait flowshop scheduling heuristic to minimize makespan. *J. Oper. Res. Soc.* **1994**, *45*, 472–478. [CrossRef]
5. Grabowski, J.; Pempera, J. Sequencing of jobs in some production system. *Eur. J. Oper. Res.* **2000**, *125*, 535–550. [CrossRef]
6. Raaymakers, W.; Hoogeveen, J. Scheduling multipurpose batch process industries with no-wait restrictions by simulated annealing. *Eur. J. Oper. Res.* **2000**, *12*, 6131–6151. [CrossRef]
7. Tseng, L.; Lin, Y. A genetic local search algorithm for minimizing total flow time in the permutation flowshop scheduling problem. *Int. J. Prod. Econom.* **2010**, *127*, 121–128. [CrossRef]
8. Allahverdi, A. A survey of scheduling problems with no-wait in process. *Eur. J. Oper. Res.* **2016**, *255*, 665–686. [CrossRef]
9. Pan, Q.; Tasgetiren, F.; Liang, Y. A discrete particle swarm optimization algorithm for the no-wait flowshop scheduling problem. *Comput. Oper. Res.* **2008**, *35*, 2807–2839. [CrossRef]

10. Pan, Q.; Ruiz, R. Local search methods for the flowshop scheduling problem with flowtime minimization. *Eur. J. Oper. Res.* **2012**, *222*, 31–43. [CrossRef]
11. Rock, H. The three-machine no-wait flow shop is NP-complete. *J. ACM* **1984**, *31*, 336–345. [CrossRef]
12. Garey, M.; Johnson, D. *Computers and Intractability, a Guide to the Theory of NP-Completeness*, 4th ed.; Freeman: New York, NY, USA, 1979.
13. Graham, R. Optimization and approximation in deterministic sequencing and scheduling. *Ann. Discret. Math.* **1979**, *5*, 287–326.
14. Rajendran, C. A heuristic for scheduling in flowshop and flowline-based manufacturing cell with multicriteria. *Int. J. Prod. Res.* **1994**, *32*, 2541–2558. [CrossRef]
15. Bertolissi, E. Heuristic algorithm for scheduling in the no-wait flow-shop. *J. Mater. Process. Technol.* **2000**, *107*, 459–465. [CrossRef]
16. Aldowaisan, T.; Allahverdi, A. New heuristics for m -machine no-wait flowshop to minimize total completion time. *Int. J. Manag. Sci.* **2004**, *32*, 345–352. [CrossRef]
17. Sapkal, S.; Laha, D. A heuristic for no-wait flow shop scheduling. *Int. J. Adv. Manuf. Technol.* **2013**, *68*, 1327–1338. [CrossRef]
18. Tasgetiren, M.; Pan, Q.; Kizilay, D.; Gao, K. A variable block insertion heuristic for the blocking flowshop scheduling problem with total flowtime criterion. *Algorithms* **2016**, *9*, 71. [CrossRef]
19. Liu, J.; Reeves, C. Constructive and composite heuristic solutions to the P//∑Ci scheduling problem. *Eur. J. Oper. Res.* **2001**, *132*, 439–452. [CrossRef]
20. Blum, C.; Roli, A. Metaheuristics in combinatorial optimization: Overview and conceptual comparison. *Appl. Soft Comput. J.* **2003**, *35*, 268–308. [CrossRef]
21. Blum, C.; Puchinger, J.; Raidl, G.; Roli, A. Hybrid metaheuristics in combinatorial optimization: A survey. *Appl. Soft Comput. J.* **2011**, *11*, 4135–4151. [CrossRef]
22. Gao, K.; Pan, Q.; Li, J.; Wang, Y.; Liang, J. A hybrid harmony search algorithm for the no-wait flow-shop scheduling problems. *Asia-Pac. J. Oper. Res.* **2012**, *29*, 12500–12512. [CrossRef]
23. Nawaz, M.; Enscore, E., Jr.; Ham, I. A Heuristic Algorithm for the m-Machine, n-Job Flow-shop Sequencing Problem. *Int. J. Manag. Sci.* **1983**, *11*, 91–95. [CrossRef]
24. Gao, K.; Suganthan, P.; Chua, T. An enhanced migrating birds optimization algorithm for no-wait flow shop scheduling problem. In Proceedings of the IEEE Symposium on Computational Intelligence in Scheduling, Singapore, 16–19 April 2013; pp. 9–13.
25. Filho, G.; Nagano, M.; Lorena, L. *Hybrid Evolutionary Algorithm for Flowtime Minimization in No-Wait Flowshop Scheduling*; Lecture Notes in Computer Science; Springer: Berlin, Germany, 2007; Volume 4827, pp. 1099–1109.
26. Zhang, Y.; Li, X.; Wang, Q. Hybrid genetic algorithm for permutation flowshop scheduling problems with total flowtime minimization. *Eur. J. Oper. Res.* **2009**, *196*, 869–876. [CrossRef]
27. Xu, X.; Xu, Z.; Gu, X. An asynchronous genetic local search algorithm for the permutation flowshop scheduling problem with total flowtime minimization. *Expert Syst. Appl.* **2011**, *38*, 7970–7979. [CrossRef]
28. Wang, F.; Rao, Y.; Tang, Q. A hybrid intelligence algorithm for no-wait flow shop scheduling. *Adv. Mater. Res.* **2013**, *712*, 2447–2451. [CrossRef]
29. Bewoor, L.; Chandra Prakash, V.; Sapkal, S. Comparative analysis of metaheuristic approaches for m-machine no-wait flow shop scheduling for minimizing total flow time with stochastic input. *Int. J. Eng. Technol.* **2016**, *8*, 3021–3026. [CrossRef]
30. Bewoor, L.; Chandra Prakash, V.; Sapkal, S. Comparative analysis of metaheuristic approaches for makespan minimization for no-wait flow shop scheduling problem. *Int. J. Electr. Comput. Eng.* **2017**, *7*, 31–37. [CrossRef]
31. Akhshabi, M.; Tavakkoli-Moghaddam, R.; Rahnamay-Roodposhti, F. A hybrid particle swarm optimization algorithm for a no-wait flow shop scheduling problem with the total flow time. *Int. J. Adv. Manuf. Technol.* **2014**, *70*, 1181–1188. [CrossRef]
32. Taillard, E. Benchmarks for basic scheduling problems. *Eur. J. Oper. Res.* **1993**, *64*, 278–285. [CrossRef]
33. Vannucci, P. ALE-PSO: An adaptive swarm algorithm to solve design problems of laminates. *Algorithms* **2009**, *2*, 710–734. [CrossRef]
34. Eberhard, R.; Kennedy, J. A new optimizer using particle swarm theory. In Proceedings of the Sixth International Symposium on Micro Machine and Human Science, Nagoya, Japan, 4–6 October 1995; pp. 39–43.

35. Bean, J. Genetic algorithms and random keys for sequencing and optimization. *ORSA J. Comput.* **1994**, *6*, 154–160. [CrossRef]
36. Liu, B.; Wang, L.; Jin, Y.-H. An effective PSO-based memetic algorithm for flowshop scheduling. *IEEE Trans. Syst. Man Cybern. Part B* **2007**, *37*, 18–27. [CrossRef]
37. Jarboui, B.; Ibrahim, S.; Siarry, P.; Rebai, A. A combinatorial particle swarm optimization for solving permutation flowshop problems. *Comput. Ind. Eng.* **2008**, *54*, 526–538. [CrossRef]
38. Khamlichi, Y.; Tahiri, A.; Abtoy, A.; Bulo, I.; Lozano, F. A hybrid algorithm for optimal wireless sensor network deployment with the minimum number of sensor nodes. *Algorithms* **2017**, *10*, 80. [CrossRef]
39. Devore, J. *Probability and Statistics for Engineering and the Sciences*, 9th ed.; Brooks Cole: Pacific Grove, CA, USA, 2016.