





Article

Hybridization of Particle Swarm Optimization with Variable Neighborhood Search and Simulated Annealing for Improved Handling of the Permutation Flow-Shop Scheduling Problem

Iqbal Hayat ¹, Adnan Tariq ¹, Waseem Shahzad ², Manzar Masud ³, Shahzad Ahmed ⁴,
Muhammad Umair Ali ^{5,*} and Amad Zafar ^{5,*}

¹ Department of Mechanical Engineering, University of Wah, Wah Cantt 47040, Pakistan; iqbalmts@gmail.com (I.H.); adnan.tariq@wecu.edu.pk (A.T.)

² Department of Mechatronics Engineering, University of Wah, Wah Cantt 47040, Pakistan; waseemshahzad@wecu.edu.pk

³ Department of Mechanical Engineering, Capital University of Science and Technology (CUST), Islamabad 44000, Pakistan; manzar.masud@cust.edu.pk

⁴ Department of Electronics Engineering, Hanyang University, Seoul 04763, Republic of Korea; shahzad1@hanyang.ac.kr

⁵ Department of Intelligent Mechatronics Engineering, Sejong University, Seoul 05006, Republic of Korea

* Correspondence: umair@sejong.ac.kr (M.U.A.); amad@sejong.ac.kr (A.Z.)

Abstract: Permutation flow-shop scheduling is the strategy that ensures the processing of jobs on each subsequent machine in the exact same order while optimizing an objective, which generally is the minimization of makespan. Because of its NP-Complete nature, a substantial portion of the literature has mainly focused on computational efficiency and the development of different AI-based hybrid techniques. Particle Swarm Optimization (PSO) has also been frequently used for this purpose in the recent past. Following the trend and to further explore the optimizing capabilities of PSO, first, a standard PSO was developed during this research, then the same PSO was hybridized with Variable Neighborhood Search (PSO-VNS) and later on with Simulated Annealing (PSO-VNS-SA) to handle Permutation Flow-Shop Scheduling Problems (PFSP). The effect of hybridization was validated through an internal comparison based on the results of 120 different instances devised by Taillard with variable problem sizes. Moreover, further comparison with other reported hybrid metaheuristics has proved that the hybrid PSO (HPSO) developed during this research performed exceedingly well. A smaller value of 0.48 of ARPD (Average Relative Performance Difference) for the algorithm is evidence of its robust nature and significantly improved performance in optimizing the makespan as compared to other algorithms.

Keywords: permutation flow-shop scheduling problems (PFSP); particle swarm optimization (PSO); makespan; hybrid particle swarm optimization (HPSO); metaheuristic



Citation: Hayat, I.; Tariq, A.; Shahzad, W.; Masud, M.; Ahmed, S.; Ali, M.U.; Zafar, A. Hybridization of Particle Swarm Optimization with Variable Neighborhood Search and Simulated Annealing for Improved Handling of the Permutation Flow-Shop Scheduling Problem. *Systems* **2023**, *11*, 221. <https://doi.org/10.3390/systems11050221>

Academic Editor: William T. Scherer

Received: 2 March 2023

Revised: 28 March 2023

Accepted: 24 April 2023

Published: 26 April 2023



Copyright: © 2023 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

Scheduling is a fundamental component of advanced manufacturing processes and systems. It efficiently utilizes resources to maximize objectives, e.g., makespan, flow time, average tardiness, etc. It plays a significant role in modern production facilities by optimally organizing and controlling the work and workloads in a manufacturing process, resulting in minimal inventory, processing time, idle time, and product cost [1]. In scheduling resources, such as machines, are allocated to tasks, such as jobs, to ensure the completion of these tasks in a limited amount of time. Scheduling while manufacturing, is basically the arrangement of jobs that can be processed on available machines subjected to different constraints.

Some of the classical models used to solve scheduling problems are job-shop, flow-shop, and open-shop. The Permutation Flow-shop Scheduling Problem (PFSP) addresses

the most important problems related to machine scheduling and involves the sequential processing of n jobs on m machines [2]. Flow-Shop scheduling problems are NP-complete [3,4], for which complete enumeration requires considerable computational effort and time exponentially increasing with the problem size. The intricate nature of these problems renders exact solution methods impractical when dealing with numerous jobs and/or machines. This is the primary rationale behind the utilization of various heuristics, found in the literature.

2. Literature Review

Pioneering efforts by Johnson [5] concluded that PFSP with more than two machines could not be solved analytically. Consequently, researchers focused on other heuristic-based approaches to handle the PFSP of more than two machines. Some notable examples include Palmer's heuristic [6], CDS (Campbell, Dudek & Smith) [7], VNS (Variable Neighborhood Search) [8], Branch & Bound [9], etc. However, with the increase in problem size, even the best heuristics tend to drift away from the optimal solutions and converge to suboptimal solutions. Therefore, the focus of research shifted towards meta-heuristics. Many such approaches, as viable solutions to PFSP, have already been reported in the literature, which includes GA (Genetic Algorithms) [10,11], PSO (Particle Swarm Optimization) [12,13] and ACO (Ant Colony Optimization) [14], Q-Learning algorithms [15], HWOA (Hybrid Whale Optimization Algorithms) [16], CWA (Enhanced Whale Optimization Algorithms) [17], and BAT-algorithms [18]. Metaheuristic-based approaches start with sequences generated randomly by a heuristic and then iterate until a stopping criterion is satisfied [19]. These approaches have been extensively applied to find optimal solutions to flow-shop scheduling problems [20]. Goldberg et al. [21] proposed GA-based algorithms as viable solutions to scheduling problem. A GA-based heuristic for flow-shop scheduling, with makespan minimization as the objective, was presented by Chen et al. [22]. The authors utilized a partial crossover, no mutation, and a different heuristic for the random generation of the initial population. A comparative analysis showed no improvement in results for a population size of more than 60. However, hybrid GA-based approaches have significantly improved results [23–27]. Despite this, the increased computational cost of such approaches is a major limitation. Therefore, comparatively more efficient algorithms such as PSO have recently been opted for more frequently [28].

PSO, initially proposed by Kennedy and Eberhart [29], is based on the “collective intelligence” exhibited by swarms of animals. In PSO, a randomly generated initial population of solutions iteratively propagates toward the optimal solution. PSO has been extensively applied to flow-shop scheduling problems. Tasgetiren et al. [30] implemented the PSO algorithm on a single machine while optimizing the total weighted tardiness problem and developed the SPV (Smallest Position Value) heuristic-based approach for solving a wide range of scheduling and sequencing problems. The authors hybridized PSO with VNS to obtain better results by avoiding local minima. Improved performance of the PSO was reported in comparison to ACO, with ARPD (average relative percent deviation) as the evaluation criteria. Another approach, presented by Tasgetiren et al. [31], utilized PSO with SPV and VNS for solving Taillard's [32] benchmark problems and concluded that PSO with VNS produced the same results as GA with VNS. Moslehi et al. [33] conducted a study on the challenges of a Limited Buffer PSFP (LBPSFP) using a hybrid VNS (HVNS) algorithm combined with SA. Despite the similar performance, the computational efficiency of PSO was found to be exceedingly better than GA [28,34,35]. In addition, Fuqiang et al. [36] proposed a Two Level PSO (TLPSO) to solve the management problem related to credit portfolio. TLPSO included internal search and external search processes, and the experimental results show that the TLPSO is more reliable than the other tested methods.

Hornig et al. [37] presented a hybrid metaheuristic by combining SA (Simulated Annealing) with PSO and compared their results with that of simple GA and PSO. The results for 20 different mathematical optimization functions confirmed the quick convergence and good optimality of SA-PSO compared to standalone GA and PSO. A similar but slightly dif-

ferent metaheuristic compounding PSO, SA, and TS (Tabu Search) was developed by Zhang et al. [38]. The algorithm obtained quality solutions while consuming lesser computational time when tested with 30 instances of 10 different sizes taken from Taillard's [39] benchmark problems for PFSP and performed significantly better than NPSO (Novel PSO) and GA. Given the optimizing capabilities of hybrid approaches, recently, researchers have focused even more on applying hybrid metaheuristics to PFSP to improve the global and local neighborhood search capabilities of the standard algorithms. The optimizing capabilities of the hybrid approaches have invigorated the researchers to apply these techniques to the global and local neighborhood search algorithms. Yannis et al. [40] hybridized his PSO-VNS algorithm with PR (Path Relinking Strategy). A comparative analysis of the technique while solving Taillard's [32] problems yielded a significantly better PSO-VNS-PR algorithm performance than PSO with constant global and local neighborhood searches. The effects of population initialization on PSO performance were studied by Laxmi et al. [41]. The authors hybridized standard PSO with a NEH (Nawaz, Ensore & Ham) heuristic for population initialization and SA for enhanced local neighborhood search. A significantly improved performance of the algorithm was reported compared to other competing algorithms. Fuqiang et al. [42] developed a technique including SA and GA for scheduled risk management of IT outsourcing projects. They concluded that SA, in combination with GA, is the superior algorithm in terms of stability and convergence.

From the literature review presented above, it can be evidently concluded that metaheuristics have increased the capability of handling NP-hard problems. Furthermore, PSO, combined with other heuristics, have performed better than other tools, e.g., GA, ACO, etc. Therefore, to further validate this conclusion, a PSO-based approach was developed during this research in a stepwise manner. First, a standard PSO was developed and validated through Taillard's [32] suggested benchmark problems. This was followed by its gradual hybridization with VNS only and then both with VNS & SA while observing the initial temperature's effect on SA optimality [43]. An internal comparison based on Taillard's benchmark problems was also carried out to justify the effect of hybridization. After validation, the hybrid PSO (HPSO)—developed during this research—was also compared with a recently reported Hybrid Genetic Simulated Annealing Algorithm (HGSA) [26] and other famous techniques based on the ARPD values. The comparison showed the effectiveness and the robust nature of the HPSO (PSO—VNS—SA), developed during this research, as it outperformed all its competitors.

3. Methodology

To execute the proposed research, a stepwise methodology adopted during this research is as follows:

- (1) Formulation of an optimization model for the PFSP.
- (2) Development of standard PSO.
- (3) Hybridization of standard PSO with VNS.
- (4) Further hybridization of PSO—VNS with SA.

3.1. Optimization Model

An optimization model is formulated for n -jobs and m -machines PFSP while having minimization of makespan (C_{max}) as the objective function so that its results can be easily compared with other approaches. To optimize C_{max} , the model needs to fulfill several constraints listed as follows:

1. Each job (j) must start its next operation on the next machine ($i + 1$) in sequence after its previous operation on the previous machine (i) is completed.
2. Each job (j) has an operation scheduled on each machine (i), i.e., each job must visit each machine only once
3. Jobs must not overtake each other in between machines, and the permutation remains the same on each subsequent machine.

The formulated model is presented in Equation (1)

$$\text{Objective} = F = \text{Minimize } C_{max} \quad (1)$$

where $C_{max} = \max (CT_j)$; CT_j is the completion time of job j and $j = 1, 2, 3 \dots \dots, n$; where n is the total number of jobs. In PFSP, the value of makespan is always defined by the last job in each permutation; therefore, to reduce the computational effort, C_{max} is determined only for the last job ($j = L$) and is given by Equation (2).

$$C_{max} = \left[CT_{1L} + \sum_{i=1}^{m-1} (ST_{(i+1)L} - CT_{iL}) + (CT_{(i+1)L} - ST_{(i+1)L}) \right] \quad (2)$$

where:

CT_{1L} = Completion time of the last job ($j = L$) in a permutation on machine1.

$ST_{(i+1)L}$ = Start time of the last job on the next machine.

$ST_{(i+1)L} - CT_{1L}$ = Waiting time of the last job ($j = L$) in a permutation on the next machine.

$CT_{(i+1)L} - ST_{(i+1)L}$ = Processing time of the last job in a permutation on the next machine.

Subject to:

$$ST_{(i+1)j} - CT_{ij} \geq 0 \quad (3)$$

Job (j) must start its next operation on the next machine ($i + 1$) after its previous operation on the previous machine (i) is completed.

$$ST_{(i+1)j} - ST_{ij} \geq 0 \quad (4)$$

This constraint ensures that the processing time of each job must be positive, i.e., each job has an operation scheduled on each machine.

$$CT_{ij} \leq ST_{i(j+1)} \quad (5)$$

This ensures that jobs do not overtake each other in between machines, and the permutation remains the same on each subsequent machine.

$$ST_{ij} \geq 0 \quad (6)$$

$$CT_{ij} \geq 0 \quad (7)$$

3.2. Solution Representation

Since PSO is domain-independent, the most complicated step is the solution representation. There is n number of coordinates or positions for n number of jobs, representing the direct relationship between the problem domain and the PSO particles. In parallel, the particle $X_i^t = x_{i1}^t, x_{i2}^t, x_{i3}^t, \dots, x_{in}^t$ represents the continuous values of positions for n number of jobs in the flow-shop problems. For the determination of sequencing, the smallest position value (SPV) was embedded in the algorithm for finding the position values x^t of the particle x_i^t proposed by Tasgetiren et al. [31]. The solution representation of particle X_i^t with its positions and velocity values, and the sequence, according to the SPV rule, is shown in Table 1.

Table 1. Solution representation of a particle.

Dimension, <i>j</i>	1	2	3	4	5	6
x_{ij}^t	1.80	−0.99	3.01	−0.72	−1.20	2.15
v_{ij}^t	3.89	2.94	3.08	−0.87	−0.20	3.16
Jobs, π_{ij}^t	5	2	4	1	6	3

$x_{i5}^t = -1.20$ is the minimum value in the position row, so the first job assigned in the permutation $j = 5$; the second minimum position value is $x_{i2}^t = -0.99$. Therefore, $j = 2$ is assigned to the next job, and so on. Otherwise stated, permutation is constructed by using the sorted value of positions.

3.3. Algorithm Implementation

The approach is implemented in three different stages using MATLAB programming. Firstly, a standard PSO is encoded, as shown in Figure 1, Part (a). To test its performance, a total number of 120 benchmark problems, devised by Taillard [32], are handled, and their respective outcomes are listed in Table 1. Later on, the same standard PSO is hybridized with VNS by allowing the best solution in each iteration of standard PSO to receive further improvements under the procedure devised by VNS. PSO is a global search algorithm and thus explores the larger search space and identifies the potential region where the optimum may exist. VNS, on the other hand, searches inside this potential space and narrows down the location of a possible optimum. In each iteration this process is repeated until the best solution is identified. This combination of standard PSO with VNS is presented in Figure 1, combination of Parts (a) and (b). The performance of hybridized PSO-VNS is also validated through the same 120 benchmark problems, and the results are listed in Table 1.

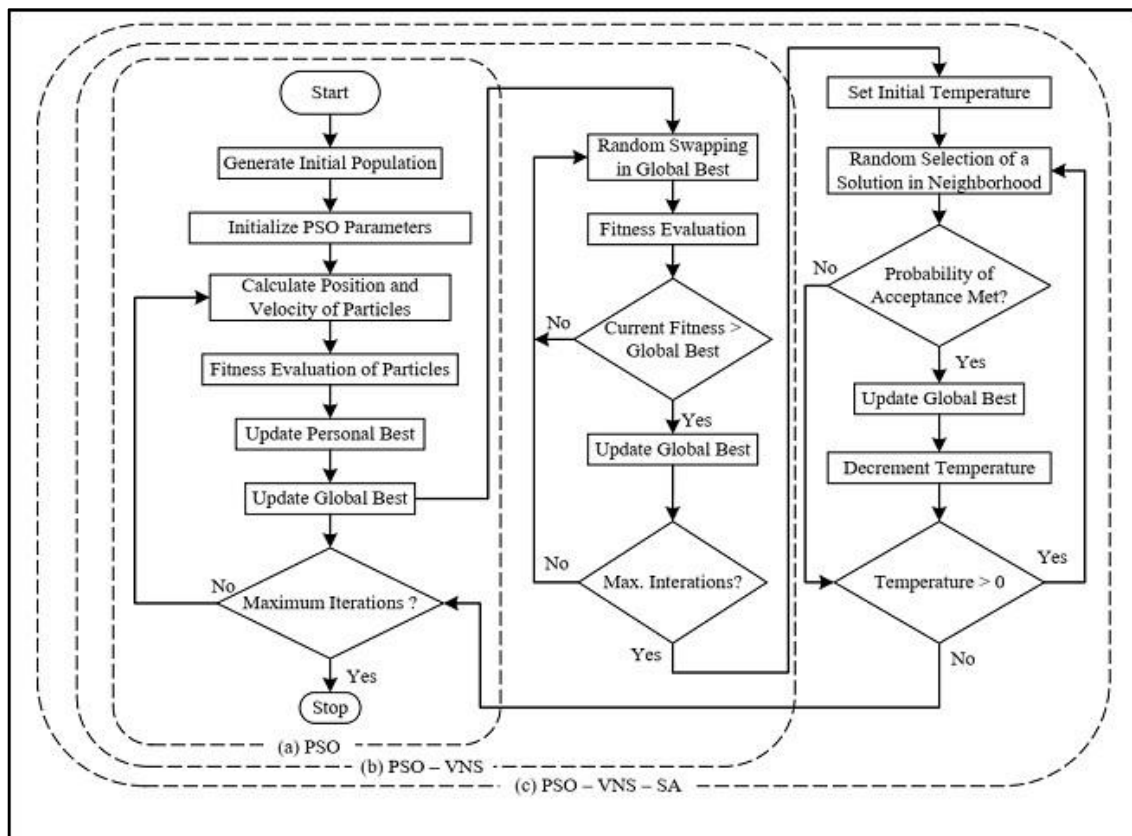


Figure 1. Flow diagram of PSO, PSO-VNS and PSO-VNS-SA.

Finally, the hybrid version of PSO-VNS is further hybridized with SA to intensify local improvement and give further chance to the best solution in each iteration of the standard PSO by searching in its neighborhood for an even better option. The evolution part of the search is taken care of by the PSO whereas both VNS and SA help in locally improving the solution (Best) identified by PSO. VNS, by nature, is a greedy search algorithm and selects only better solutions whereas SA chooses even a slightly inferior solution if it falls within an already calculated range of probability. This helps the overall algorithm in avoiding local minima, reaching global optimum, and maintaining diversity in the population as well. The combined operating strategy of HPSO is shown in Figure 1. Its performance has also been confirmed through the same 120 benchmark problems. All the algorithms were run on a Core-i7 processor with 8 Gb of RAM using Windows 10 as the operating system. For comparison, the results are also listed in Table 1. The statistical comparison of the algorithms was based on a *t*-test, and the level of significance was set to $p = 0.05$.

3.4. Sensitivity Analysis

Like other evolutionary computational algorithms, particle swarm optimization is sensitive to hyperparameters and may suffer from premature convergence due to poorly selected control parameters. This may result in suboptimal performance and convergence at a local minimum. Extensive research has been conducted to characterize the influence of various control parameters on the performance of PSO. Isiet et al. [44] studied the effect of individual parameter variation on the performance of a standard benchmark problem. The authors testified a profound impact of inertia weight and acceleration coefficients on algorithm performance and reported a range of [0, 0.5] for the inertia weight matrix contrary to the previously reported [45,46] range of [0.9, 1.2]. More recent studies have proposed a dynamically changing inertia matrix approach [47] for increased exploration at the initial stage and reduced randomness at the converging stage of the algorithm. The algorithm developed in this research followed a similar approach with an inertia matrix in the range of [0.4, 1.2] and a decrement factor of 0.975. In addition to the inertia matrix, C1 (self-adjustment) and C2 (social adjustment) learning factors significantly impact algorithm's performance. Previous studies have experimented with contrasting values of C1 and C2 and have suggested that the sum of the two variables should not be more than 4 [48], as proposed by [29]. The algorithm, developed during this research, exhibited similar behavior, which utilized a value of 2 for both the social coefficients. The main PSO algorithm was hybridized with Simulated Annealing (SA) to improve its performance against local minima. The computational procedure of SA is analogous to the positioning of molecules to the lowest state of energy as well as the constancy state through a controlled cooling rate for improving material properties. SA's behavior depends on hyperparameters too. These include initial temperature, acceptance criteria, and cooling rate. The cooling rate is dynamically adjusted, starting with a higher value and decreasing it with increasing iterations. The initial temperature is kept at a higher value to ensure maximum variation and is slowly cooled with a specific cooling rate to the final temperature, which is also the stopping criterion for the algorithm. The approach developed during this research utilizes an initial temperature of 100 cooled to a final temperature of 0.5 with a final cooling rate of 0.99.

4. Results and Discussion

The sizes of Taillard's benchmark problems, used for validation of the three algorithms presented in Figure 1, ranged from 20×5 ($n \times m$) to 500×20 . Each problem was given ten runs while using a swarm size of twice the number of jobs, and the inertial weights used, ranged from 1.2 to 0.4, with a decrement factor of 0.975. The cognitive and social acceleration coefficients (C1 and C2) were initialized with a value of 2, and the maximum iterations were kept limited to 100. The results obtained are listed in Tables 2–5 and presented in Figures 2–5. The recently reported results of the Q-Learning algorithm [15], though limited to a maximum problem size of 50×20 , have also been analyzed for

performance comparisons. The data in Table 2 and Figure 2 show that the algorithm achieved the upper bound in all instances of the 20-job problems resulting in an ARPD value of zero. A similar trend can be observed for the 50-job problems (Table 3, Figure 3), where the HPSO results in an ARPD value of 0.80, which is significantly better than the PSO-VNS and PSO with ARPD values of 3.21 and 4.20, respectively. The performance consistency is also apparent from the 100- and 200-job instances results, where the HPSO achieved ARPD values of 0.48 and 0.63 compared to 3.03 and 3.06 for PSO-VNS and 6.30 and 8.49 for the PSO. Figures 2–5 clearly depict that the performance of HPSO in comparison to standard PSO and PSO–VNS was consistently superior as it returned improved solutions for the entire set of 120 benchmark problems.

Table 2. Results of PSO, PSO-VNS, and HPSO for Taillard’s benchmark problems.

Taillard’s Problems		Makespan			RPD		
Problem Instance	Upper Bound	PSO	PSO-VNS	HPSO	PSO	PSO-VNS	HPSO
20 × 5							
1	1278	1294	1297	1278 ± 0.00	1.25	1.49	0.00
2	1359	1365	1366	1359 ± 0.00	0.44	0.52	0.00
3	1081	1108	1100	1081 ± 1.79	2.50	1.76	0.00
4	1293	1311	1311	1293 ± 0.00	1.39	1.39	0.00
5	1235	1248	1248	1235 ± 1.00	1.05	1.05	0.00
6	1195	1217	1210	1195 ± 0.00	1.84	1.26	0.00
7	1234	1251	1251	1234 ± 2.50	1.38	1.38	0.00
8	1206	1228	1218	1206 ± 0.00	1.82	1.00	0.00
9	1230	1264	1261	1230 ± 0.00	2.76	2.52	0.00
10	1108	1135	1135	1108 ± 0.00	2.44	2.44	0.00
ARPD					1.69	1.48	0.00
20 × 10							
1	1582	1632	1625	1582 ± 0.00	3.16	2.72	0.00
2	1659	1710	1708	1659 ± 3.00	3.07	2.95	0.00
3	1496	1551	1542	1496 ± 2.94	3.68	3.07	0.00
4	1377	1418	1417	1377 ± 3.85	2.98	2.90	0.00
5	1419	1488	1479	1419 ± 3.00	4.86	4.23	0.00
6	1397	1443	1449	1397 ± 2.50	3.29	3.72	0.00
7	1484	1535	1531	1484 ± 0.00	3.44	3.17	0.00
8	1538	1575	1560	1538 ± 5.82	2.41	1.43	0.00
9	1593	1638	1636	1593 ± 0.92	2.82	2.70	0.00
10	1591	1637	1637	1591 ± 4.41	2.89	2.89	0.00
ARPD					3.26	2.98	0.00
20 × 20							
1	2297	2355	2380	2297 ± 7.03	2.53	3.61	0.00
2	2099	2168	2144	2099 ± 4.58	3.29	2.14	0.00
3	2326	2375	2360	2326 ± 5.88	2.11	1.46	0.00
4	2223	2291	2291	2223 ± 4.50	3.06	3.06	0.00

Table 2. Cont.

Taillard's Problems		Makespan			RPD		
Problem Instance	Upper Bound	PSO	PSO-VNS	HPSO	PSO	PSO-VNS	HPSO
5	2291	2364	2361	2291 ± 4.58	3.19	3.06	0.00
6	2226	2288	2276	2226 ± 2.40	2.79	2.25	0.00
7	2273	2332	2332	2273 ± 3.38	2.60	2.60	0.00
8	2200	2260	2248	2200 ± 3.41	2.73	2.18	0.00
9	2237	2304	2304	2237 ± 2.50	3.00	3.00	0.00
10	2178	2271	2253	2178 ± 2.94	4.27	3.44	0.00
ARPD					2.95	2.68	0.00
Average					2.63	2.38	0.00

Table 3. Results of PSO, PSO-VNS, and HPSO for Taillard's 50-job benchmark problems.

Taillard's Problems		Makespan			RPD		
Problem Instance	Upper Bound	PSO	PSO-VNS	HPSO	PSO	PSO-VNS	HPSO
50 × 5							
1	2724	2740	2735	2724 ± 0.00	0.59	0.40	0.00
2	2834	2882	2882	2834 ± 6.08	1.69	1.69	0.00
3	2621	2664	2628	2621 ± 0.00	1.64	0.27	0.00
4	2751	2795	2782	2751 ± 0.80	1.60	1.13	0.00
5	2863	2864	2864	2863 ± 0.00	0.03	0.03	0.00
6	2829	2848	2848	2829 ± 1.99	0.67	0.67	0.00
7	2725	2774	2758	2725 ± 4.18	1.80	1.21	0.00
8	2683	2719	2707	2683 ± 8.40	1.34	0.89	0.00
9	2552	2589	2585	2552 ± 5.96	1.45	1.29	0.00
10	2782	2786	2782	2782 ± 0.00	0.14	0.00	0.00
ARPD					1.10	0.76	0.00
50 × 10							
1	2991	3140	3105	3018 ± 8.45	4.98	3.81	0.90
2	2867	3040	2980	2890 ± 11.77	6.03	3.94	0.80
3	2839	3015	2950	2860 ± 15.32	6.20	3.91	0.74
4	3063	3242	3180	3063 ± 2.00	5.84	3.82	0.00
5	2976	3140	3095	2995 ± 14.72	5.51	4.00	0.64
6	3006	3148	3106	3043 ± 2.50	4.72	3.33	1.23
7	3093	3225	3195	3115 ± 6.00	4.27	3.30	0.71
8	3037	3158	3090	3048 ± 2.80	3.98	1.75	0.36
9	2897	3030	2995	2909 ± 8.00	4.59	3.38	0.41
10	3065	3160	3140	3099 ± 1.50	3.10	2.45	1.11
ARPD					4.92	3.37	0.69
50 × 20							
1	3850	4216	4107	3910 ± 8.33	9.51	6.68	1.56
2	3704	4052	4009	3765 ± 13.12	9.40	8.23	1.65
3	3640	3899	3860	3709 ± 23.83	7.12	6.04	1.90
4	3723	3960	3930	3792 ± 13.80	6.37	5.56	1.85
5	3611	3850	3780	3675 ± 15.69	6.62	4.68	1.77
6	3681	3930	3890	3743 ± 26.00	6.76	5.68	1.68
7	3704	3915	3860	3762 ± 24.47	5.70	4.21	1.57
8	3691	3920	3890	3753 ± 25.47	6.20	5.39	1.68
9	3743	3960	3925	3805 ± 20.16	5.80	4.86	1.66
10	3756	3955	3896	3822 ± 3.60	5.30	3.73	1.76
ARPD					6.88	5.51	1.71
Average					4.30	3.21	0.80

Table 4. Results of PSO, PSO-VNS and HPSO for Taillard’s 100 job benchmark problems.

Taillard’s Problems		Makespan			RPD		
Problem Instance	Upper Bound	PSO	PSO-VNS	HPSO	PSO	PSO-VNS	HPSO
100 × 5							
1	5493	5523	5495	5493 ± 1.00	0.55	0.04	0.00
2	5268	5302	5290	5268 ± 7.84	0.65	0.42	0.00
3	5175	5225	5213	5175 ± 2.29	0.97	0.73	0.00
4	5014	5035	5023	5014 ± 3.63	0.42	0.18	0.00
5	5250	5311	5260	5250 ± 1.83	1.16	0.19	0.00
6	5135	5161	5160	5135 ± 0.98	0.51	0.49	0.00
7	5246	5292	5261	5246 ± 0.00	0.88	0.29	0.00
8	5094	5130	5120	5094 ± 2.45	0.71	0.51	0.00
9	5448	5485	5475	5448 ± 3.00	0.68	0.50	0.00
10	5322	5360	5342	5322 ± 6.50	0.71	0.38	0.00
ARPD					0.72	0.37	0.00
100 × 10							
1	5770	6112	5879	5770 ± 6.42	5.93	1.89	0.00
2	5349	5654	5455	5364 ± 6.80	5.70	1.98	0.28
3	5676	5945	5797	5680 ± 5.88	4.74	2.13	0.07
4	5781	6204	5940	5810 ± 11.27	7.32	2.75	0.50
5	5467	5880	5619	5485 ± 8.65	7.55	2.78	0.33
6	5303	5625	5368	5303 ± 4.00	6.07	1.23	0.00
7	5595	5830	5727	5595 ± 2.80	4.20	2.36	0.00
8	5617	5985	5747	5624 ± 10.80	6.55	2.31	0.12
9	5871	6164	5990	5898 ± 4.80	4.99	2.03	0.46
10	5845	6074	5922	5860 ± 10.08	3.92	1.32	0.26
ARPD					5.70	2.08	0.20
100 × 20							
1	6202	7003	6678	6308 ± 6.75	12.92	7.67	1.71
2	6183	7035	6566	6246 ± 27.00	13.78	6.19	1.02
3	6271	7017	6726	6361 ± 3.60	11.90	7.26	1.44
4	6269	7031	6651	6331 ± 0.00	12.16	6.09	0.99
5	6314	7131	6685	6403 ± 11.76	12.94	5.88	1.41
6	6364	7142	6781	6440 ± 9.62	12.23	6.55	1.19
7	6268	7092	6722	6342 ± 18.99	13.15	7.24	1.18
8	6404	7233	6852	6475 ± 8.33	12.95	7.00	1.11
9	6275	7072	6726	6342 ± 10.50	12.70	7.19	1.07
10	6434	7080	6777	6510 ± 16.23	10.04	5.33	1.18
ARPD					12.47	6.64	1.23
Average					6.30	3.03	0.48

Table 5. Results of PSO, PSO-VNS, and HPSO for Taillard’s 200- and 500-job benchmark problems.

Taillard’s Problems		Makespan			RPD		
Problem Instance	Upper Bound	PSO	PSO-VNS	HPSO	PSO	PSO-VNS	HPSO
200 × 10							
1	10,862	11,224	10,993	10,862 ± 16.17	3.33	1.21	0.00
2	10,480	11,194	10,628	10,480 ± 14.18	6.81	1.41	0.00
3	10,922	11,435	11,122	10,922 ± 11.42	4.70	1.83	0.00
4	10,889	11,240	11,025	10,889 ± 0.00	3.22	1.25	0.00
5	10,524	11,145	10,650	10,550 ± 11.22	5.90	1.20	0.25
6	10,329	10,929	10,468	10,365 ± 8.97	5.81	1.35	0.35
7	10,854	11,409	11,087	10,880 ± 14.11	5.11	2.15	0.24
8	10,730	11,292	10,745	10,746 ± 5.88	5.24	0.14	0.15
9	10,438	11,098	10,515	10,438 ± 7.20	6.32	0.74	0.00
10	10,675	11,152	10,922	10,724 ± 6.86	4.47	2.31	0.46
ARPD					5.09	1.36	0.14

Table 5. Cont.

Taillard's Problems		Makespan			RPD		
Problem Instance	Upper Bound	PSO	PSO-VNS	HPSO	PSO	PSO-VNS	HPSO
200 × 20							
1	11,195	11,517	11,625	11,310 ± 11.31	2.88	3.84	1.03
2	11,203	12,685	11,850	11,326 ± 9.17	13.23	5.78	1.10
3	11,281	12,715	11,887	11,404 ± 17.76	12.71	5.37	1.09
4	11,275	12,596	11,836	11,380 ± 11.00	11.72	4.98	0.93
5	11,259	12,658	11,780	11,394 ± 14.10	12.43	4.63	1.20
6	11,176	12,640	11,702	11,289 ± 20.38	13.10	4.71	1.01
7	11,360	12,805	11,936	11,487 ± 8.71	12.72	5.07	1.12
8	11,334	12,679	11,832	11,464 ± 11.99	11.87	4.39	1.15
9	11,192	12,642	11,768	11,287 ± 0.00	12.96	5.15	0.85
10	11,288	12,760	11,923	11,415 ± 9.31	13.04	5.63	1.13
ARPD					11.66	4.95	1.06
500 × 20							
1	26,059	28,524	26,808	26,220 ± 27.92	9.46	2.87	0.62
2	26,520	29,096	27,177	26,684 ± 84.41	9.71	2.48	0.62
3	26,371	28,810	27,276	26,546 ± 34.74	9.25	3.43	0.66
4	26,456	27,895	27,178	26,640 ± 71.73	5.44	2.73	0.70
5	26,334	28,646	27,028	26,516 ± 47.5	8.78	2.64	0.69
6	26,477	28,750	27,263	26,674 ± 18.79	8.58	2.97	0.74
7	26,389	28,540	27,116	26,642 ± 31.41	8.15	2.75	0.96
8	26,560	28,890	27,348	26,743 ± 42.81	8.77	2.97	0.69
9	26,005	28,582	26,760	26,195 ± 30.62	9.91	2.90	0.73
10	26,457	28,844	27,204	26,604 ± 62.60	9.02	2.82	0.56
ARPD					8.71	2.86	0.70
Average					8.49	3.06	0.63

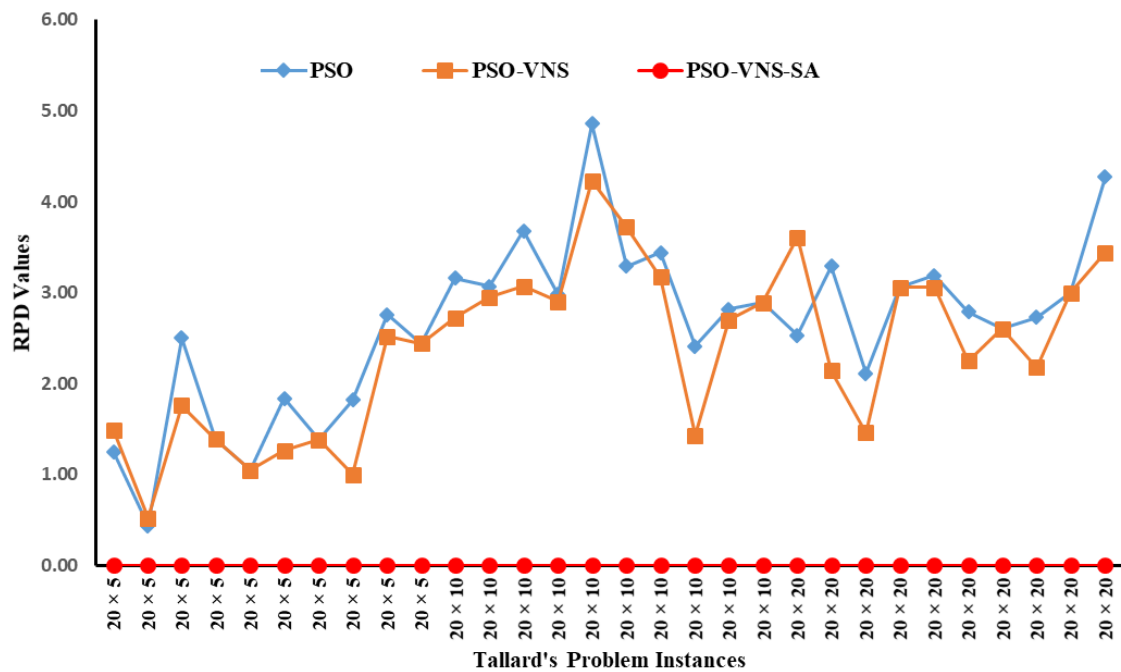


Figure 2. Performance comparison of the three algorithms for the 20-Job.

presented in this paper performed comparatively better with an initial temperature setting of 100° and a cooling rate of 0.95° . A possible reason for this deviation from other algorithms is the comparatively wider initial search space that increased the probability acceptance level of SA.

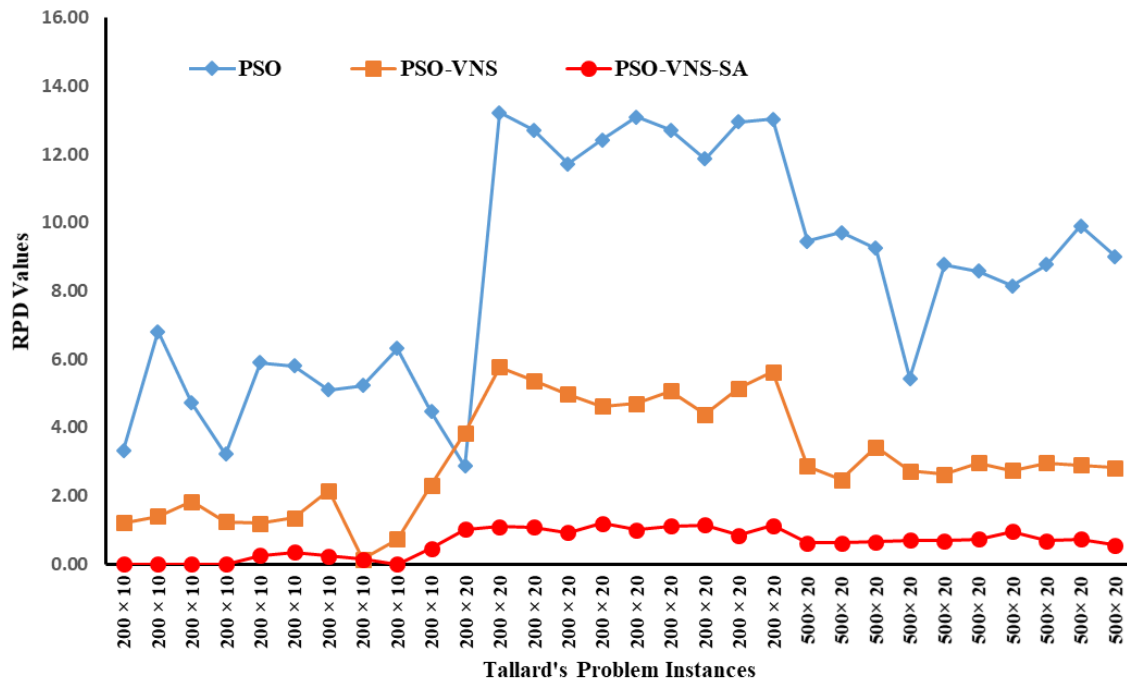


Figure 5. Performance comparison of the RPD values of the three PSO algorithms for the 200- and 500-Job Taillard's instances.

To further elaborate on the HPSO's effectiveness, its performance has also been compared against hybrid GA (HGA)-based approaches. Since HGAs have been extensively reported in the literature and are widely regarded as the best metaheuristic for these sorts of problems, to justify the robust behavior of HPSO developed during this research, its performance was also compared with HGA by Tseng et al. [27] and HGSA by Wei et al. [26] (hybrid GA with SA). The algorithm performed significantly better than HGA ($p = 0.05$) and HGSA ($p = 0.05$), as evident from Figure 6.

Comparisons were also carried out with four different versions of GA, i.e., SGA (Simple Genetic Algorithm) [24], MGGA (Mining Gene Genetic Algorithm) [23], ACGA (Artificial Chromosome with Genetic Algorithm) [23], and SGGA (Self-Guided Genetic Algorithm) [24], as presented in Figure 7. The deviation of the GA from the upper bound, which increases in magnitude with the increasing problem size, was significantly more than HPSO. Thus, it can be concluded that the algorithm developed during this research is comparatively more robust and performs better than other hybrid GA techniques even while handling larger problem sizes.

Once the internal comparison of HPSO with standard PSO, PSO-VNS, and validation against HGA and HGSA was completed, the last part of validation was against other notable techniques already reported in the literature. For this purpose, a more detailed comparison was carried out with WOA [16], Chaotic Whale Optimization (CWA) [17], the BAT-algorithm [18], NEHT (NEH algorithm together with the improvement presented by Taillard) [31], ACO [50], CPSO (Combinatorial PSO) [51], PSOENT (PSO with Expanding Neighborhood Topology) [40], and HAPSO (Hybrid Adaptive PSO) [52]. This comparison was solely based on ARPD values and is shown in Table 6 and graphically illustrated in Figure 8. The improved performance of HPSO, developed during this research, is evident as compared to other reported hybrid heuristics.

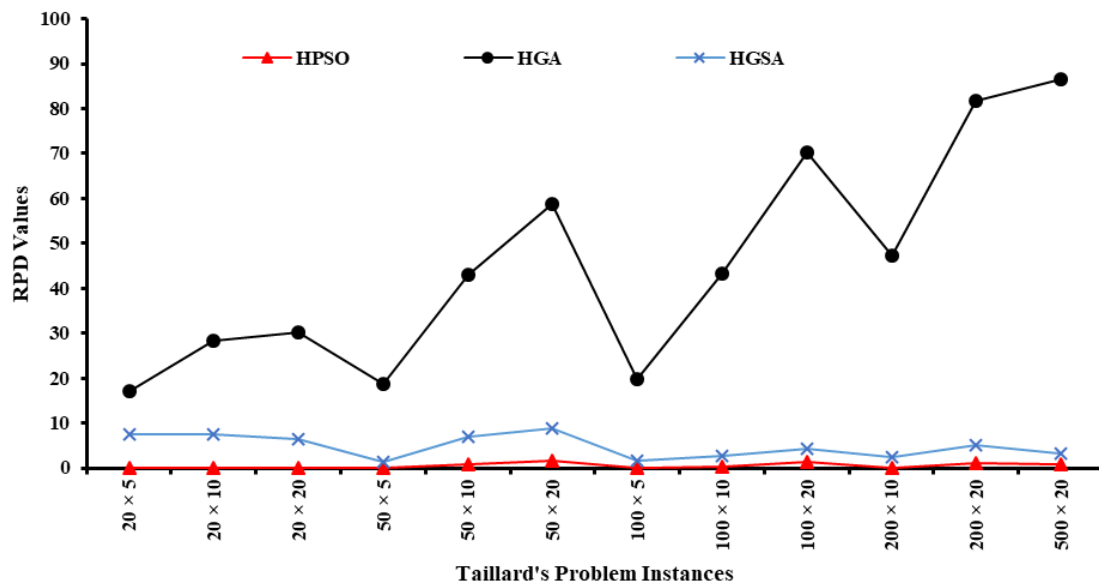


Figure 6. Comparison of HPSO with HGA and HGSA heuristics for 120 Taillard’s instances.

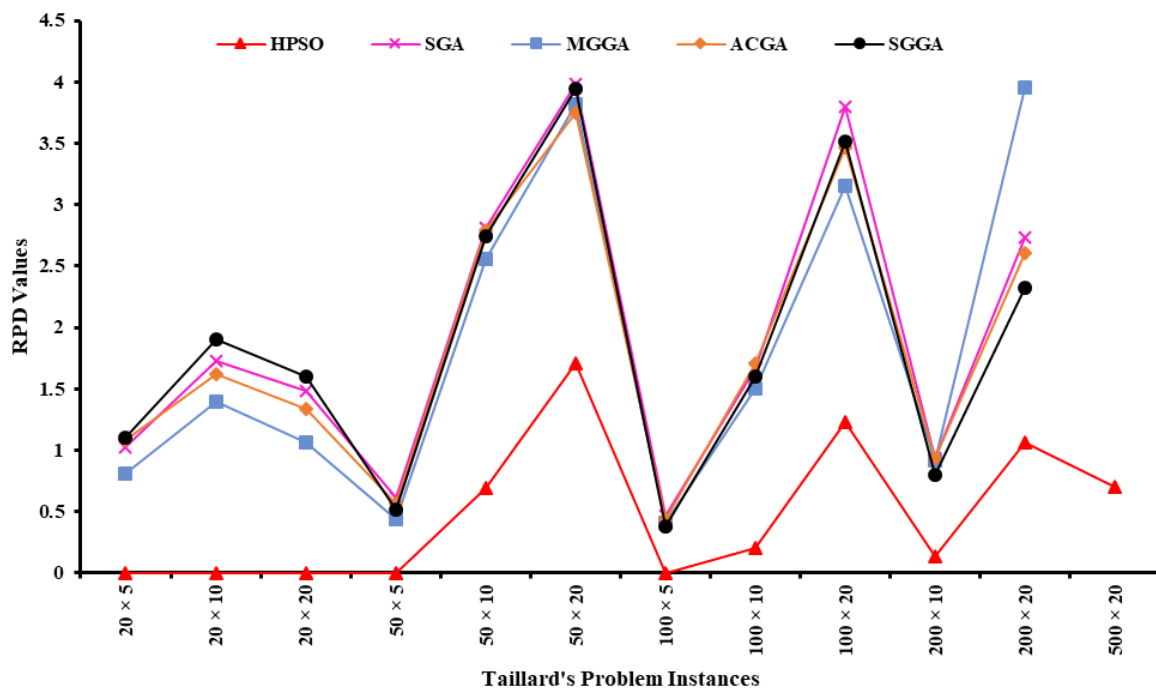


Figure 7. Comparison of HPSO with hybrid GA-based heuristics.

The results of the Q-Learning algorithm [15] have not been shown in Table 6 due to the limited results reported by the author, as only 30 out of the 120 problems limited to a maximum problem size of 50×20 were analyzed. However, a comparison was performed for ARPD values for the limited number of problems for both HPSO and Q-learning. The results show a superior performance of the HPSO compared to the Q-Learning algorithm.

Table 6. Comparison of ARPD values of different metaheuristics with the algorithm developed during this research.

Problem Instance	Average Relative Percentage Deviation															
	HPSO	HWA	CWA	BAT	CLN	NEHT	ACO	CPSO	HGA	HGSA	SGA	MGGA	ACGA	SGGA	PSOENT	HAPSO
20 × 5	0	0	0.04	0.54	2.25	3.35	1.19	1.05	17.03	7.57	1.02	0.81	1.08	1.1	0	0
20 × 10	0	0	0.67	2.61	4.01	5.02	1.7	2.42	28.27	7.42	1.73	1.4	1.62	1.9	0.07	0.09
20 × 20	0	0	0.68	2.54	3.32	3.73	1.6	1.99	30.1	6.37	1.48	1.06	1.34	1.6	0.08	0.07
50 × 5	0	0	0.08	0.06	0.71	0.84	0.43	0.9	18.66	1.31	0.61	0.44	0.57	0.52	0.02	0.05
50 × 10	0.69	0.54	0.79	4.00	4.23	5.12	0.89	4.85	42.9	7.05	2.81	2.56	2.79	2.74	2.11	2.01
50 × 20	1.71	0.44	2.37	6.65	5.73	6.26	2.71	6.4	58.72	8.86	3.98	3.82	3.75	3.94	3.83	3.2
100 × 5	0	0.09	0.05	0.06	0.28	0.46	0.22	0.74	19.9	1.49	0.47	0.41	0.44	0.38	0.09	0.14
100 × 10	0.2	0.46	0.41	0.86	1.45	2.13	1.22	2.94	43.26	2.76	1.67	1.5	1.71	1.6	1.26	1.17
100 × 20	1.23	1.52	1.87	3.84	4.74	5.23	2.22	7.11	70.2	4.21	3.8	3.15	3.47	3.51	4.37	4.13
200 × 10	0.14	0.49	0.28	0.68	1.1	1.43	0.64	2.17	47.33	2.38	0.94	0.92	0.94	0.8	1.02	1.06
200 × 20	1.06	2.07	1.82	2.91	4.07	4.41	1.3	6.89	81.7	5.16	2.73	3.95	2.61	2.32	4.27	4.27
500 × 20	0.7	0.91	1.19	1.66	1.91	2.24	1.68	-	86.49	3.3	-	-	-	-	2.73	3.43
Average	0.48	0.54	0.85	2.20	2.82	3.35	1.32	3.41	45.38	4.82	1.93	1.82	1.85	1.86	1.65	1.64

Note: The smallest ARPD values reported for each group of problems and overall average are presented as boldfaced.

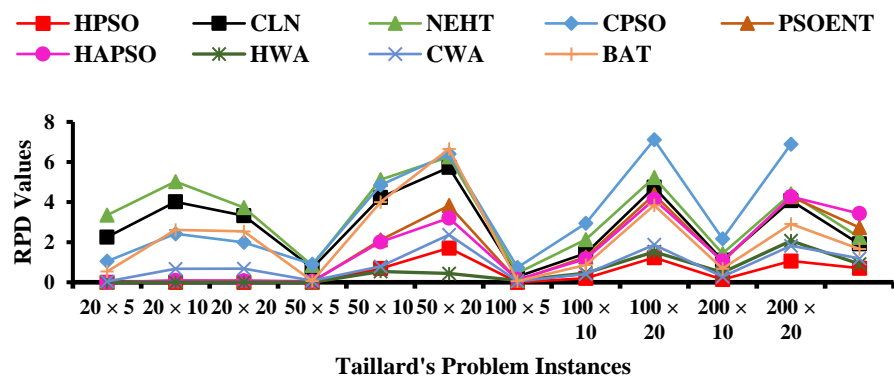


Figure 8. Comparison of HPSO with CLS, NEHT, and three other hybridized PSO techniques.

A row-wise comparison yields the performance variation of different approaches for individual problem groups. It is important to note that the technique developed during this research (HPSO) outperformed the other algorithms for each problem set. Although there is a performance variation across the techniques for different problem sizes, the algorithm was consistently better than all the other techniques. A smaller ARPD value in each problems group resulted in an overall smallest average ARPD value of 0.48, significantly better than the closest value of 0.85 for the HWA algorithm. It validates the claim that the HPSO approach, developed during this research, is comparatively more robust and remains consistent even while handling large sized problems. Furthermore, the average computation time of HPSO is reported in Table 7.

Table 7. Comparison of Average Computation time in seconds of PSO, PSO-VNS, and HPSO.

Serial Number	Problem Size	Matrix Size	PSO	PSO-VNS	HPSO
1	20 × 5	100	4.32	6.16	8.34
2	20 × 10	200	10.80	12.47	15.01
3	20 × 20	400	18.68	20.50	23.81
4	50 × 5	250	16.38	18.81	22.55
5	50 × 10	500	39.00	73.76	111.63
6	50 × 20	1000	87.60	135.39	190.19
7	100 × 5	500	59.40	71.92	89.20
8	100 × 10	1000	332.64	403.58	501.14
9	100 × 20	2000	591.60	718.40	892.53
10	200 × 10	2000	763.20	925.89	1149.64
11	200 × 20	4000	839.40	1028.85	1285.45
12	500 × 20	10,000	998.40	1225.25	1531.97

5. Conclusions

As a member of the class of NP-complete problems, PFSP has been regularly researched and reported in the literature. Several heuristic-based approaches in the literature can efficiently handle this problem. However, for the larger problem sizes, most of the researchers focused on hybridized metaheuristics due to their ability to produce quality results in polynomial time, even for large-sized problems. Following this trend, a PSO-based approach was developed during this research in a stepwise manner. First, a standard PSO was developed, then it was hybridized with VNS, and finally, with SA. The final version, HPSO, outperformed not just standard PSO and PSO-VNS, but it also performed exceedingly well against other algorithms, including HGA, HGSA, ACO, BAT, WOA, and CWA. Comparisons based on the ARPD values showed that the performance of HPSO remained comparatively consistent, as evidenced by its small overall ARPD value of 0.48.

Author Contributions: Conceptualization, I.H. and W.S.; data curation, I.H.; formal analysis, W.S.; investigation, M.U.A.; methodology, I.H. and A.T.; project administration, A.Z.; supervision, S.A.; validation, A.T.; writing—original draft, I.H.; writing—review & editing, A.T., M.M., S.A., M.U.A. and A.Z. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: All data that support the findings of this study are included within the article.

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

References

1. Arora, D.; Agarwal, G. Meta-Heuristic Approaches for Flowshop Scheduling Problems: A Review. *Int. J. Adv. Oper. Manag.* **2016**, *8*, 1–16. [[CrossRef](#)]
2. Khurshid, B.; Maqsood, S.; Omair, M.; Sarkar, B.; Ahmad, I.; Muhammad, K. An Improved Evolution Strategy Hybridization With Simulated Annealing for Permutation Flow Shop Scheduling Problems. *IEEE Access* **2021**, *9*, 4505–94522. [[CrossRef](#)]
3. Book, R.V. Book Review: Computers and Intractability: A Guide to the Theory of NP -Completeness. *Bull. Am. Math. Soc.* **1980**, *3*, 898–905. [[CrossRef](#)]
4. Graham, R.L.; Lawler, E.L.; Lenstra, J.K.; Kan, A.H.G.R. Optimization and Approximation in Deterministic Sequencing and Scheduling: A Survey. In *Discrete Optimization II*; Annals of Discrete Mathematics; Hammer, P.L., Johnson, E.L., Korte, B.H., Eds.; Elsevier: Amsterdam, The Netherlands, 1979; Volume 5, pp. 287–326.
5. Johnson, S.M. Optimal Two- and Three-Stage Production Schedules with Setup Times Included. *Nav. Res. Logist. Q.* **1954**, *1*, 61–68. [[CrossRef](#)]
6. Hundal, T.S.; Rajgopal, J. An Extension of Palmer's Heuristic for the Flow Shop Scheduling Problem. *Int. J. Prod. Res.* **1988**, *26*, 1119–1124. [[CrossRef](#)]
7. Campbell, H.G.; Dudek, R.A.; Smith, M.L. A Heuristic Algorithm for the n Job, m Machine Sequencing Problem. *Manag. Sci.* **1970**, *16*, B-630–B-637. [[CrossRef](#)]
8. Zhang, G.; Zhang, L.; Song, X.; Wang, Y.; Zhou, C. A Variable Neighborhood Search Based Genetic Algorithm for Flexible Job Shop Scheduling Problem. *Clust. Comput.* **2019**, *22*, 11561–11572. [[CrossRef](#)]
9. Ruiz, R.; Maroto, C. A Comprehensive Review and Evaluation of Permutation Flowshop Heuristics. *Eur. J. Oper. Res.* **2005**, *165*, 479–494. [[CrossRef](#)]
10. Mumtaz, J.; Zailin, G.; Mirza, J.; Rauf, M.; Sarfraz, S.; Shehab, E. Makespan Minimization for Flow Shop Scheduling Problems Using Modified Operators in Genetic Algorithm. *Adv. Transdiscipl. Eng.* **2018**, *8*, 435–440. [[CrossRef](#)]
11. Umam, M.S.; Mustafid, M.; Suryono, S. A Hybrid Genetic Algorithm and Tabu Search for Minimizing Makespan in Flow Shop Scheduling Problem. *J. King Saud. Univ. Comput. Inf. Sci.* **2022**, *34*, 7459–7467. [[CrossRef](#)]
12. Radha Ramanan, T.; Iqbal, M.; Umarali, K. A Particle Swarm Optimization Approach for Permutation Flow Shop Scheduling Problem. *Int. J. Simul. Multidiscip. Des. Optim.* **2014**, *5*, A20. [[CrossRef](#)]
13. Marichelvam, M.K.; Geetha, M.; Tosun, Ö. An Improved Particle Swarm Optimization Algorithm to Solve Hybrid Flowshop Scheduling Problems with the Effect of Human Factors—A Case Study. *Comput. Oper. Res.* **2020**, *114*, 104812. [[CrossRef](#)]
14. Shen, C.; Chen, Y.L. Blocking Flow Shop Scheduling Based on Hybrid Ant Colony Optimization. *Int. J. Simul. Model.* **2020**, *19*, 313–322. [[CrossRef](#)]
15. He, Z.; Wang, K.; Li, H.; Song, H.; Lin, Z.; Gao, K.; Sadollah, A. Improved Q-Learning Algorithm for Solving Permutation Flow Shop Scheduling Problems. *IET Collab. Intell. Manuf.* **2022**, *4*, 35–44. [[CrossRef](#)]
16. Abdel-Basset, M.; Manogaran, G.; El-Shahat, D.; Mirjalili, S. A Hybrid Whale Optimization Algorithm Based on Local Search Strategy for the Permutation Flow Shop Scheduling Problem. *Future Gener. Comput. Syst.* **2018**, *85*, 129–145. [[CrossRef](#)]

17. Li, J.; Guo, L.; Li, Y.; Liu, C.; Wang, L.; Hu, H. Enhancing Whale Optimization Algorithm with Chaotic Theory for Permutation Flow Shop Scheduling Problem. *Int. J. Comput. Intell. Syst.* **2021**, *14*, 651–675. [[CrossRef](#)]
18. Bellabai, J.R.; Leela, B.N.M.; Kennedy, S.M.R. Testing the Performance of Bat-Algorithm for Permutation Flow Shop Scheduling Problems with Makespan Minimization. *Braz. Arch. Biol. Technol.* **2022**, *65*. [[CrossRef](#)]
19. Liao, C.-J.; Tseng, C.-T.; Luarn, P. A Discrete Version of Particle Swarm Optimization for Flowshop Scheduling Problems. *Comput. Oper. Res.* **2007**, *34*, 3099–3111. [[CrossRef](#)]
20. Abdel-Basset, M.; Mohamed, R.; Abouhawwash, M.; Chakraborty, R.K.; Ryan, M.J. A Simple and Effective Approach for Tackling the Permutation Flow Shop Scheduling Problem. *Mathematics* **2021**, *9*, 270. [[CrossRef](#)]
21. Goldberg, D.E.; Holland, J.H. Genetic Algorithms and Machine Learning. *Mach. Learn.* **1988**, *3*, 95–99. [[CrossRef](#)]
22. Chen, C.-L.; Vempati, V.S.; Aljaber, N. An Application of Genetic Algorithms for Flow Shop Problems. *Eur. J. Oper. Res.* **1995**, *80*, 389–396. [[CrossRef](#)]
23. Chang, P.-C.; Chen, S.-H.; Fan, C.-Y.; Chan, C.-L. Genetic Algorithm Integrated with Artificial Chromosomes for Multi-Objective Flowshop Scheduling Problems. *Appl. Math. Comput.* **2008**, *205*, 550–561. [[CrossRef](#)]
24. Chen, S.-H.; Chang, P.-C.; Cheng, T.C.E.; Zhang, Q. A Self-Guided Genetic Algorithm for Permutation Flowshop Scheduling Problems. *Comput. Oper. Res.* **2012**, *39*, 1450–1457. [[CrossRef](#)]
25. Chen, W.; Hao, Y.F. Genetic Algorithm-Based Design and Simulation of Manufacturing Flow Shop Scheduling. *Int. J. Simul. Model.* **2018**, *17*, 702–711. [[CrossRef](#)]
26. Wei, H.; Li, S.; Jiang, H.; Hu, J.; Hu, J. Hybrid Genetic Simulated Annealing Algorithm for Improved Flow Shop Scheduling with Makespan Criterion. *Appl. Sci.* **2018**, *8*, 2621. [[CrossRef](#)]
27. Tseng, L.-Y.; Lin, Y.-T. A Hybrid Genetic Algorithm for No-Wait Flowshop Scheduling Problem. *Int. J. Prod. Econ.* **2010**, *128*, 144–152. [[CrossRef](#)]
28. Hassan, R.; Cohanin, B.; de Weck, O.; Venter, G. A Comparison of Particle Swarm Optimization and the Genetic Algorithm. In Proceedings of the 46th AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics and Materials Conference, Austin, TX, USA, 18–21 April 2015; American Institute of Aeronautics and Astronautics: Reston, Virginia, 2005.
29. Kennedy, J.; Eberhart, R. Particle Swarm Optimization. In Proceedings of the ICNN'95—International Conference on Neural Networks, Perth, Australia, 27 November–1 December 1995; pp. 1942–1948.
30. Tasgetiren, M.F.; Sevcli, M.; Liang, Y.-C.; Gencyilmaz, G. Particle Swarm Optimization Algorithm for Single Machine Total Weighted Tardiness Problem. In Proceedings of the 2004 Congress on Evolutionary Computation (IEEE Cat. No.04TH8753), Portland, OR, USA, 19–23 June 2004; pp. 1412–1419.
31. Tasgetiren, M.F.; Sevcli, M.; Liang, Y.-C.; Gencyilmaz, G. Particle Swarm Optimization Algorithm for Permutation Flowshop Sequencing Problem. In *Lecture Notes in Computer Science*; Springer: Berlin/Heidelberg, Germany, 2004; pp. 382–389.
32. Taillard, E. Benchmarks for Basic Scheduling Problems. *Eur. J. Oper. Res.* **1993**, *64*, 278–285. [[CrossRef](#)]
33. Moslehi, G.; Khorasanian, D. A Hybrid Variable Neighborhood Search Algorithm for Solving the Limited-Buffer Permutation Flow Shop Scheduling Problem with the Makespan Criterion. *Comput. Oper. Res.* **2014**, *52*, 260–268. [[CrossRef](#)]
34. Lian, Z.; Gu, X.; Jiao, B. A Similar Particle Swarm Optimization Algorithm for Permutation Flowshop Scheduling to Minimize Makespan. *Appl. Math. Comput.* **2006**, *175*, 773–785. [[CrossRef](#)]
35. Singh, M.R.; Mahapatra, S.S. A Swarm Optimization Approach for Flexible Flow Shop Scheduling with Multiprocessor Tasks. *Int. J. Adv. Manuf. Technol.* **2012**, *62*, 267–277. [[CrossRef](#)]
36. Lu, F.-Q.; Huang, M.; Ching, W.-K.; Siu, T.K. Credit Portfolio Management Using Two-Level Particle Swarm Optimization. *Inf. Sci.* **2013**, *237*, 162–175. [[CrossRef](#)]
37. Shieh, H.-L.; Kuo, C.-C.; Chiang, C.-M. Modified Particle Swarm Optimization Algorithm with Simulated Annealing Behavior and Its Numerical Verification. *Appl. Math. Comput.* **2011**, *218*, 4365–4383. [[CrossRef](#)]
38. Zhang, X.-F.; Koshimura, M.; Fujita, H.; Hasegawa, R. Hybrid Particle Swarm Optimization and Convergence Analysis for Scheduling Problems. In Proceedings of the 14th Annual Conference Companion on Genetic and Evolutionary Computation, Philadelphia, PA, USA, 7–11 July 2002; ACM: New York, NY, USA, 2002; pp. 307–314.
39. Taillard, E. Some Efficient Heuristic Methods for the Flow Shop Sequencing Problem. *Eur. J. Oper. Res.* **1990**, *47*, 65–74. [[CrossRef](#)]
40. Marinakis, Y.; Marinaki, M. Particle Swarm Optimization with Expanding Neighborhood Topology for the Permutation Flowshop Scheduling Problem. *Soft Comput.* **2013**, *17*, 1159–1173. [[CrossRef](#)]
41. Bewoor, L.; Chandra Prakash, V.; Sankal, S. Evolutionary Hybrid Particle Swarm Optimization Algorithm for Solving NP-Hard No-Wait Flow Shop Scheduling Problems. *Algorithms* **2017**, *10*, 121. [[CrossRef](#)]
42. Lu, F.; Bi, H.; Huang, M.; Duan, S. Simulated Annealing Genetic Algorithm Based Schedule Risk Management of IT Outsourcing Project. *Math. Probl. Eng.* **2017**, *2017*, 6916575. [[CrossRef](#)]
43. Li, Y.; Wang, C.; Gao, L.; Song, Y.; Li, X. An Improved Simulated Annealing Algorithm Based on Residual Network for Permutation Flow Shop Scheduling. *Complex. Intell. Syst.* **2021**, *7*, 1173–1183. [[CrossRef](#)]
44. Isiet, M.; Gadala, M. Sensitivity Analysis of Control Parameters in Particle Swarm Optimization. *J. Comput. Sci.* **2020**, *41*, 101086. [[CrossRef](#)]
45. Shi, Y.; Eberhart, R. A Modified Particle Swarm Optimizer. In Proceedings of the 1998 IEEE International Conference on Evolutionary Computation Proceedings, Anchorage, AK, USA, 4–9 May 1998; IEEE World Congress on Computational Intelligence (Cat. No.98TH8360). pp. 69–73.

46. Bansal, J.C.; Singh, P.K.; Saraswat, M.; Verma, A.; Jadon, S.S.; Abraham, A. Inertia Weight Strategies in Particle Swarm Optimization. In Proceedings of the 2011 Third World Congress on Nature and Biologically Inspired Computing, Salamanca, Spain, 19–21 October 2011; pp. 633–640.
47. Zhu, X.; Wang, H. A New Inertia Weight Control Strategy for Particle Swarm Optimization. In *AIP Conference Proceedings*; American Institute of Physics Inc.: College Park, MD, USA, 2018; Volume 1955.
48. Ozcan, E.; Mohan, C.K. Particle Swarm Optimization: Surfing the Waves. In Proceedings of the 1999 Congress on Evolutionary Computation-CEC99 (Cat. No. 99TH8406), Washington, DC, USA, 6–9 July 1999; Volume 3, pp. 1939–1944.
49. Zhang, L.; Wu, J. A PSO-Based Hybrid Metaheuristic for Permutation Flowshop Scheduling Problems. *Sci. World J.* **2014**, *2014*, 902950. [[CrossRef](#)]
50. Ying, K.-C.; Liao, C.-J. An Ant Colony System for Permutation Flow-Shop Sequencing. *Comput. Oper. Res.* **2004**, *31*, 791–801. [[CrossRef](#)]
51. Jarboui, B.; Ibrahim, S.; Siarry, P.; Rebai, A. A Combinatorial Particle Swarm Optimisation for Solving Permutation Flowshop Problems. *Comput. Ind. Eng.* **2008**, *54*, 526–538. [[CrossRef](#)]
52. Marinakis, Y.; Marinaki, M. An Adaptive Parameter Free Particle Swarm Optimization Algorithm for the Permutation Flowshop Scheduling Problem. In *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*; Springer: Berlin/Heidelberg, Germany, 2019; Volume 11943, pp. 168–179.

Disclaimer/Publisher’s Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.