Available online at www.qu.edu.iq/journalcm
JOURNAL OF AL-QADISIYAH FOR COMPUTER SCIENCE AND MATHEMATICS
ISSN:2521-3504(online) ISSN:2074-0204(print)

Enhanced PSO Based on Population Initialization and Exploration for the Permutation Flow Shop Scheduling Problem

Azhar Y. Abdulhussein, Mohanad AL-Behadili*

Department of Mathematics, College of Science, University of Basrah, Iraq.

*Email: aizhar.jasim.sci@uobasrah.edu.iq, *Email: mohanad.saad@uobasrah.edu.iq*

ARTICLE INFO

Article history:
Received: 25 /10/2021
Rrevised form: 15 /11/2021
Accepted : 21 /12/2021
Available online: 02 /01/2022

Keywords:

Permutation Flow Shop Scheduling Problem;
Particle Swarm Optimization;
N-NEH+ Heuristic;
Iterated Local Search.

ABSTRACT

In this paper, a hybrid PSO (NLPSO*) is adapted to improve the obtained local optimal solution for the Permutation Flow Shop Scheduling Problem (PFSP) with minimizing the makespan. In this method, an improved NEH heuristic called N-NEH+ is used to generate a good initial population. Then the PSO is triggered, followed by Iterated Local Search (ILS) to increase the coverage of exploration and exploitation search in the solution space. Both of the N-NEH+ and ILS are simple and efficient algorithms. A computational study is performed to show the efficiency of the proposed technique. Several of well-known PFSP instances of small, medium, and large sizes were used in this study. The experimental study shows that the NLPSO* algorithm is significantly efficient in reaching better local optimal solutions.

MSC.41A25; 41A35; 41A3

https://doi.org/10.29304/jqcm.2021.13.4.867

1. Introduction

Scheduling problem is belonging to bigger class of the Combinatorial Optimization Problems (COP). It is one of the most important and common operations research problems in real life, such as manufacturing, production, and other applications. The scheduling problem is a decision-making process used on an ordinary foundation in many manufacturing and services industries. In manufacturing processes, the job is the item to be manufactured, and resources are machines. The general goal of scheduling is to decrease production time and costs [1].

The PFSP is one of the well-known types of scheduling problems. It is very important in daily life applications and the theory of scheduling [1]. The PFSP consists of number of n jobs $(i = 1, \dots, n)$ flowing in the same order on m machines $(j = 1, \dots, m)$. Other assumptions in PFSP are that it assumes each machine handles each job in an equal order, and no machine can handle more than one job simultaneously. Different objective functions have been used

∗Corresponding author: : Azhar Abdulhussein.

Email addresses: : aizhar.jasim.sci@uobasrah.edu.iq.

Communicated by: Dr. Rana Jumaa Surayh aljanabi.

with the PFSP. The most common criterion is makespan or so-called (C_{max}). Other different objective functions defined for the PFSP include a minimum of maximum completion time [1], total weighted [2], etc. In this paper, the PFSP is considered to minimize the makespan. The PFSP is an NP-Hard problem [3], that means it has massive number of feasible solutions. Different methods are proposed and developed in the literature of PFSP. These methods have been classified into various categories: exact, heuristics, metaheuristics, artificial intelligence, etc. Exact methods have been applied for PFSP to find the global optimal solution. The branch and bound method is one of the well-known exact methods used for small instances of the PFSP [4]. Since the PFSP is an NP-Hard problem, the size of the problem is important in exact methods. For this, when the size of the problem increases, exact methods consume huge amount of time. This reason pushed researchers to find alternative methods to handle this type of the NP-Hard problem.

In 1954 [5], Johnson introduced the first heuristic method for the PFSP. The author solves the problem to minimize the makespan of jobs. He considers the case of only two machines to schedule n jobs. [6] presented a new algorithm for solving PFSP with minimizing the makespan called CDS, considered as an extension of Johnson's rule. In 1983, Nawaz, Enscore, and Ham introduced the first efficient constructive heuristic called NEH, that still standing today as one of the best heuristics that applied for the PFSP [7]. Several researchers have developed the NEH heuristic for the PFSP, such as [8] proposed two methods to improve the original NEH, based on two independent stages in the method. The results showed that the successive use of the two improvement methods resulted in an average improvement compared to the effective results of the original NEH heuristic. In [9], the authors improved NEH using a new index value that combines standard deviation and means. In addition, they used a new local search method to improve the method, and the results showed superiority over other experimental methods used to obtain the best solutions. [10] introduced an efficient N-NEH+ method, which extends the classical version of the NEH method. The authors used the N-list of N jobs to be candidates for partial sequencing. In this work, an initial population of the NLPSO* method is generated using N-NEH+.

Metaheuristics are advanced optimization methods used to treat large and complex COPs. For example, [11] used a fireworks algorithm to solve the job shop scheduling problem. Many metaheuristics have been proposed to solve the PFSP with makespan minimization. For example, Simulated Annealing [12], Genetic Algorithm [13], Tabu Search [14], Ant Colony Optimization [15], PSO [16,17], Locust Swarm Optimization [18] and so on. For more details about the methods to solve the PFSP with minimizing the makespan, see [19]. The PSO belongs to the population-based nature inspired and swarm intelligence types of meta-heuristics. This algorithm is inspired by the movement of some organisms such as birds, fish, and insects. The PSO has been used successfully in solving different COPs, including scheduling problems. This algorithm was introduced by Kennedy and Eberhart in 1995 [20]. Tasgetiren et al. [17] used the PSO to solve PFSP with the makespan criterion. The authors used the smallest value position rule (SPV) to convert the position vector into a sequence of particles. Then the variable local search of neighbor search method (VNS) is applied to improve the solution quality. In [21], the authors introduced a PSO-based hybrid metaheuristic for PFSP by integrating simulated annealing and VNS with PSO to improve the exploitation ability. Also, they used the path relinking strategy to improve exploration ability. In [22], the authors combined PSO with NEH to increase the efficiency of the solution. Also, they used the VNS to overcome early convergence of PSO.

In this paper, a hybrid NLPSO* is adapted and used for solving PFSP with minimizing the makespan. The NLPSO* algorithm combines PSO and improved N-NEH+ heuristics with ILS, where the N-NEH+ and ILS are fast and simple algorithms. The improved N-NEH+ heuristic is used to generate good quality initial population. Also, ILS is implemented to improve the exploitation and exploration in the solution space. The remainder of this work includes the following: The definition of the PFSP is given in section 2. The PSO, using the N-NEH+ to generate an initial population, and ILS algorithms are discussed in Section 3. Section 4 includes the parameters used and the experimental results. The last section, 5, discusses the conclusions and future works.

2. Definition of the Problem

In the PFSP, the sequence of processing n-jobs on m-machines is kept unchanged. In other words [1], there are n jobs $J_1, J_2, \dots, J_n$ must be scheduled on m machines $M_1, M_2, \dots, M_m$ in the same order. The goal is to find the optimal sequence among $n!$ sequence. Let p_{ij} is a non-negative integer indicating the time required to process job i by machine j, and it is called processing time, where $i = 1,2,..,n$ and $j = 1,2,\dots,m$. There are some assumptions to be considered in the PFSP:

- All operations must start at time zero.
- No process should be interrupted during its execution.
- Each job must be treated on one machine, as well as each machine must treat one job at a time.
- One machine cannot be set to process more than one job simultaneously.
- Processing of all jobs must be done in the same order.

In this work, the sequence that minimizes the maximum completion time is determined using the objective function of makespan. The makespan is the minimum of C_{max}, which indicates for time to complete the processing of the last job n on the last machine m and can be calculated as shown below:

- $C_{11} = p_{11}$.
- $C_{i1} = p_{(i)1} + C_{(i-1)1} \quad i = 2, \dots, n.$
- $C_{1j} = p_{1(j)} + C_{1(j-1)} \quad j = 2, \dots, m.$
- $C_{ij} = p_{ij} + max\{C_{(i-1)j}, C_{i(j-1)}\} \quad i = 2, \dots, n \ , \ j = 2,..,m.$
- $C_{max} = C_{nm}$.

3. The proposed methods

In this section, three methods of PSO, NEH with N-NEH+ and ILS, and their hybridization are discussed in detail.

3.1 The PSO algorithm

PSO is a stochastic search technique used to find local optimal solutions for COPs. This algorithm is inspired by the collective behavior of some organisms like birds, fish, and insects for searching for food. PSO is considered one of

the important metaheuristic algorithms, which has many applications in various areas of life, such as communications, design, power systems, control, and many more. In PSO, each of individuals inside the swarm is called a particle. This particle is a candidate solution. Each of these particles has its velocity and position. To get the optimal solution, each i^{th} particle adjusts its orientation in the search space according to the best position obtained so far (pbest) and the best position obtained by any particle in the whole swarm (gbest). After finding the pbest and the gbest, the particle updates both the velocity and the position in every dimension t $(1 \leqslant t \leqslant T)$ by:

$$v_{i,t}(k+1) = \omega v_{i,t}(k) + c_1 r_1 \left[p_{i,t}(k) - x_{i,t}(k)\right] + c_2 r_2 \left[p_{g,t}(k) - x_{i,t}(k)\right] \quad (1)$$

$$x_{i,t}(k+1) = x_{i,t}(k) + v_{i,t}(k+1) \quad (2)$$

Where r_1 and r_2 are random samples generated within [0,1]. k is the iteration count. ω determines the amount of the particle's previous velocity conserved and is called inertia weight. Acceleration coefficients are a term for both c_1 and c_2 and they are always positive. $v_{i,t}(k+1)$ and $v_{i,t}(k)$ are the next and current of velocity i^{th} particle respectively. The particle i at iteration k is denoted by $X_i(k)$, this particle can be written as a $[x_{i,1}(k), x_{i,2}(k), \dots, x_{i,T}(k)]$, where each element in this vector indicates that the particle i is located at iteration k with respect to dimension t. The velocity V of each i^{th} particle at iteration k can also be written as a vector $[v_{i,1}(k), v_{i,2}(k), \dots, v_{i,T}(k)]$. The personal best (pbest) position can be written as a vector $P_i = [p_{i,1}, p_{i,2}, \dots, p_{i,T}]$, and the global pest (gbest) position can also be written as a vector $G(k) = [g_1(k), g_2(k), \dots, g_T(k)]$ [17].

The PFSP with minimizing the makespan was solved using PSO for the first time by Tasgetiren et al. [17], where they used the heuristic SPV rule to rearrange particles in descending sequence. The steps can be as follow:

1. After initializing the parameters, the algorithm starts.
2. At $k = 0$.
3. Generate both the initial velocity and the initial position randomly, the position of the particle as $X_i(0) = \left[x_{i,1}(0), x_{i,2}(0), \dots, x_{i,T}(0)\right]$ and its velocity as $V_i(0) = [v_{i,1}(0), v_{i,2}(0), \dots, v_{i,T}(0)], i = 1,2,..,\sigma$.
4. Use the SPV rule to find the sequence $\Gamma_i(0) = \{\Gamma_{i,1}(0), \Gamma_{i,2}(0), \dots, \Gamma_{i,T}(0)\}$.
5. Set $P_i(0) = X_i(0)$, and calculate the fitness function (makespan) for each particle, then determine which has the lowest fitness function let be, $X_\varphi(0)$, $\varphi \in i = 1,2,..,\sigma$.
6. Compute the gbest $G(0)$, where $G(0) = X_\varphi(0)$.
7. For $k = k + 1$.
8. Update ω as $\omega\ (k+1) = \alpha.\ \omega\ (k)$, where α is the diminishing factor.
9. Update the velocity using equation 1, then update the position using equation 2.
10. Find the sequence $\Gamma_i(k+1) = \{\Gamma_{i,1}(k+1), \Gamma_{i,2}(k+1), \dots, \Gamma_{i,T}(k+1)\}$ by applying the SPV rule.

11. Find the fitness function (makespan) for each X_i.
12. If $fitnessX_i(k+1) < fitnessP_i(k)$, then update the pbest as $P_i(k+1) = X_i(k+1)$ and determine which particle has the lowest fitness function, let be, $X_\varphi(k+1)$.
13. If $fitnessX_\varphi(k+1) < fitnessG(k)$, then update the gbest as $G(k+1) = X_\varphi(k+1)$.
14. The algorithm stops if the specified maximum number of iterations or CPU is exceeded. Otherwise, repeat the same steps above, starting with step 7.

3.2 The NEH and NEH+ heuristics

The NEH algorithm [7] is considered one of the most famous constructive heuristic methods for finding the best sequence for a PFSP with minimizing the makespan. The main idea of the NEH heuristic is that the jobs with largest total process time should be given higher priority to be processed. NEH is the most efficient measure of flow shop scheduling problem, which builds a sequence of jobs by inserting recursively unscheduled jobs into the subsequent partial sequences. NEH has been used to solve many problems, and it has also been used as an initial solution in some methods. In this work, a good initial population to the proposed NLPSO* algorithm is generated using a modified NEH heuristic called N-NEH+. Subsequently, the SPV rule is used to construct a job sequence for this population [17]. The NEH and NEH+ algorithms can be described in Figures 1 and 2.

1	**Input**: n jobs, m machines, and the processing times of each job on each machine.
2	**Compute** the total processing times for each job on all machine.
3	**Sort** the n jobs in descending order according to the total processing times and **generate** initial sequence Γ_0.
4	**Generate** a partial sequence containing the first two jobs of Γ_0.
5	**Take** the first job from the partial sequence and insert it into all possible positions of that sequence, and **create** two partial sequences.
6	**Calculate** the $makespan$ for each of the two partial sequences.
7	**Keep** the sequence with minimizing the $makespan$ as a new partial sequence.
8	**Set** $i = 3$.
9	**Take** the job in the i^{th} position of Γ_0 and enter it in all possible positions in the previous partial sequence.
10	**Calculate** the $makespan$ for all the resulting sequences.
11	**Keep** the sequence with minimizing the $makespan$ as a new partial sequence.
12	**If** $n = i$, **stop**, otherwise set $i = i + 1$ and go to step 9.
13	**Output**: Best solution.

FIGURE 1. Procedure of the NEH Heuristic

1 **Input:** n jobs, m machines, and p_{ij}.

2 C**ompute** the total processing times for each job on all machine.

3 **Sort** the n jobs in descending order according to the total processing times and **generate** initial list Γ_0.

4 **Put** $i = 1$ and create partial sequence $R = 1$.

5 **Initialize** an N-list of candidate jobs $\Gamma_N = \{i+1, \dots, i+N\}$ and delete the relevant jobs from Γ_0.

6 **Evaluate** each of the jobs in N-list, and then insert the best obtained job in the partial sequence R, and remove it from Γ_N.

7 **If** ($\Gamma_0 \neq \phi$), take the first job from Γ_0 and append it to N-list and delete this job from Γ_0.

8 **Repeat** steps 6 and 7 until all jobs are scheduled.

9 **Output:** Best solution.

FIGURE 2. Procedure of the N-NEH+ Heuristic

3.3 Iterated Local Search

ILS is an essential and successful algorithm [23,25], which is fundamentally used for improve large-scale optimization problems and give local optimal solution. Also, it may have the ability to make significant changes at any stage. [25] proposed the first ILS algorithm for the PFSP. ILS is applied to the given problem through four procedures; generate an initial solution, perturbation, local search, and acceptance criterion. In this work, the ILS presented in [25] is used. The ILS procedure can be summarized in Figure 3, where a local search procedure based on the insertion neighborhood was used. A number of swap-moves or at least one interchange move were applied in the perturbation procedure. Finally, a local search for Large-Step Markov Chain [24] with fixed temperature was used as an acceptance criterion. It accepts the worst new solution with a probability $e^{\frac{\left(c_{max}(\Gamma') - c_{max}(\Gamma)\right)}{T}}$, if a new solution is less than or equal to the old, it is always accepted, where T is a temperature parameter. The purpose of the perturbation is to escape from the local optima or a specific search space by optimizing the candidate solutions more than the modifications made in the local search stage. The acceptance criterion is of great importance in determining which solutions will continue. Figure 3 shows the ILS procedure, where Γ' represents the new permutation of jobs, Γ^* represents the optimal permutation.

Input: $\Gamma_0 = \{\Gamma_1, \dots, \Gamma_n\}$ the initial sequence
$makespan \leftarrow great\ value$
$\Gamma^* \leftarrow \Gamma_0,\ \Gamma \leftarrow \Gamma_0$
while {the stopping criterion not satisfied}
do
Choose Γ' from the neighborhood of $N_k(\Gamma)$
if $Cmax(\Gamma') < Cmax(\Gamma)$, **then** $\Gamma \leftarrow \Gamma'$
if $Cmax(\Gamma) < Cmax(\Gamma^*)$
then $\Gamma^* \leftarrow \Gamma$
end if
else
if $random \leq e^{-\left(\frac{Cmax\ (\Gamma') - Cmax\ (\Gamma)}{T}\right)}$
then $\Gamma \leftarrow \Gamma'$
end if
end if
end while
Output: Γ^* and makespan

FIGURE 3. Procedure of the ILS

4. Experimental results

In this study, the well-known benchmark instances of the PFSP, including Carlier [26] and Taillard [27], are tested. The implementations of the proposed algorithms have been done using MapleSoft 2020. The computer specification is 8GB RAM and an Intel(R) Core (TM) i7-8565U/1.99 GHz processor. Table 1 shows the parameters that used in this study.

TABLE 1. The parameters used for the proposed algorithms

Expression	Given parameter
c_1, c_2	2, 2
ω_0	0.975
α	0.9
ρ	$2n$
The number of iterations	500
Replications	10

The implementation is repeated ten times for each instance to obtain more reliable results. Suppose the best-obtained solution through any algorithm is C_{best} and assume the lower bound of makespan is C_{lb}. This paper considers three types of relative deviations: WPD, RPD, and APD. WPD is the worst relative percentage deviation to C_{best}, RPD is the best relative percentage deviation to C_{best}, and APD is the average percentage deviation to C_{best}. WPD, RPD and APD are defined in the following equations:

$$WPD = \frac{Opt_{Sol} - Worst_{Sol}}{Opt_{Sol}} \quad (3)$$

$$RPD = \frac{Opt_{Sol} - Best_{Sol}}{Opt_{Sol}} \quad (4)$$

$$APD = \frac{Opt_{Sol} - Average_{Sol}}{Opt_{Sol}} \quad (5)$$

This section compares the proposed hybrid NLPSO* algorithm against some well-known and recent algorithms for the PFSP with minimizing the makespan. The comparison includes the following algorithms:

1. NPSO*.
2. NPSO.
3. ILPSO.
4. PSO [17].
5. VNSPSO [17].
6. EDA-CSO [28].
7. BA [28].

NPSO* is the hybridization of NEH+ and PSO. Similarly, the NPSO includes NEH and PSO. Also, ILPSO represents the ILS and PSO. While the VNSPSO refers to the application of PSO and VNS [17]. Moreover, the EDA-CSO is the improved cat swarm algorithm [28]. Finally, BA is the Bat algorithm [28]. In Tables 2, 3, and 4, the average of 11 instances of each size of Taillard's benchmark problems is given with ten replications. Table 2 shows the comparison of ILPSO and VNSPSO; the results showed that the ILPSO produced better solutions compared to VNSPSO, precisely in medium and large instances ($50 \times 20, 100 \times 10, 100 \times 5, 100 \times 20, 200 \times 10$, and 200×20). In Table 3, the NPSO* is compared against the NPSO; the results illustrated that the performance of NPSO* is better than NPSO in eight instances out of 11. Table 4 and Figure 4 show that the NLPSO* algorithm gives better results than the PSO algorithm. This confirms the strength and effectiveness of the proposed NLPSO* algorithm in finding local optimal solutions, especially for medium and large-scale PFSP problems.

TABLE 2. RPD of VNSPSO and ILPSO for Taillard's instances

	ILPSO	*VNSPSO*
20x5	0.62	0.49
20x10	0.98	0.46
20x20	0.67	0.34
50x5	0.48	0.22
50x10	0.71	0.64
50x20	0.09	0.48
100x5	0.06	0.17
100x10	0.17	0.33
100x20	0.26	0.39
200x10	0.08	0.22
200x20	0.07	0.36

TABLE 3. RPD of NPSO and NPSO* for Taillard's instances

	NPSO*	NPSO
20x5	0	0.12
20x10	0.26	0.28
20x20	0.04	0.17
50x5	0.07	0.19
50x10	0.97	0.6
50x20	1.3	0.29
100x5	0.02	0.11
100x10	0.19	0.41
100x20	0.09	0.17
200x10	0.01	0.14
200x20	0.03	0.36

TABLE 4. RPD of PSO and NLPSO* for Taillard's instances

	PSO	NLPSO*
20x5	1.25	0
20x10	1.19	0
20x20	1.15	0
50x5	0.70	0.02
50x10	1.16	0.94
50x20	1.35	0.99
100x5	0.34	0
100x10	1.04	0.89
100x20	0.99	0.04
200x10	0.71	0.41
200x20	0.81	0.53

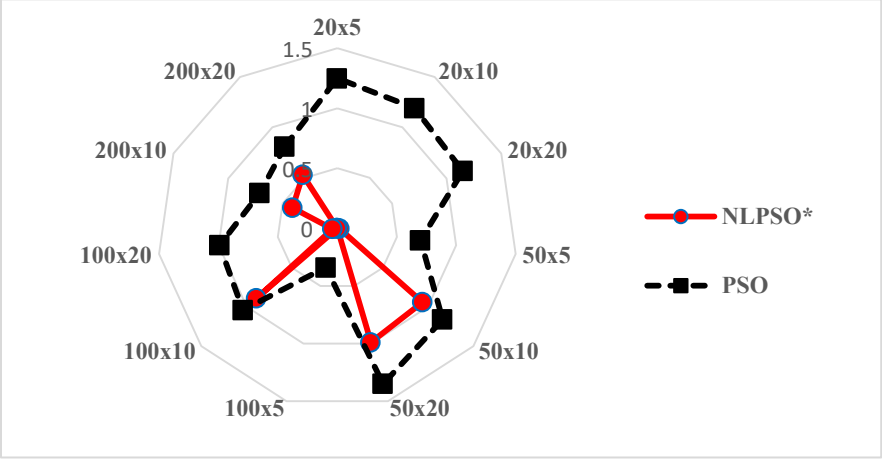

FIGURE 4. RPD of NLPSO* and PSO for Taillard's instances

For Carlier's instances, the performance of NLPSO*, PSO, EDA-CSO, and BA are tested, as given in Table 5. The results showed the proposed NLPSO* algorithm's superiority over the other algorithms. Figures 5 and 6 include the APD and WPD for the compared algorithms.

TABLE 5. Performance of NLPSO*, PSO, EDA-CSO and BA for Carlier's instances

		NLPSO*			PSO			EDA-CSO			BA		
	$n \times m$	BRE	ARE	WRE	BRE	ARE	WRE	BRE	ARE	WRE	BRE	ARE	WRE
Car1	11 × 5	0	0	0.07	0	0.52	2.72	0	0.01	0.20	0	0.32	1.68
Car2	13 × 4	0	1.09	4.72	0	5.22	10.8	0	1.37	5.00	0	3.68	6.29
Car3	12 × 5	0	1.35	2.98	0	3.77	9.04	0	1.85	3.17	0	2.36	3.87
Car4	14 × 4	0	0.22	4.64	0	3.86	6.55	0	0.37	4.98	0	2.64	5.25
Car5	10 × 7	0	0.24	1.42	0	1.24	2.12	0	0.18	1.09	0	1.01	1.84
Car6	8 × 9	0	0.65	1.96	0	2.06	5.7	0	0.52	2.12	0	1.44	3.39
Car7	7 × 7	0	0.23	2.57	0	1.61	4.51	0	0.15	2.03	0	0.91	2.58
Car8	8 × 8	0	0.09	1.01	0	3.85	8.88	0	0.29	1.33	0	1.08	2.51

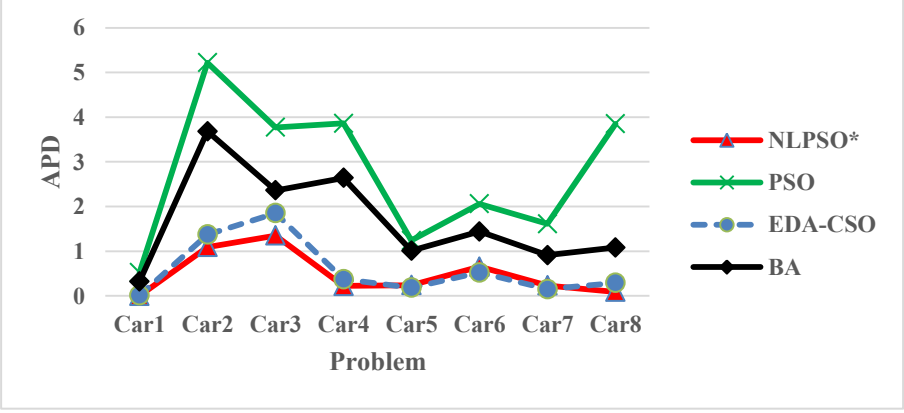

FIGURE 5. APD of NLPSO*. PSO. EDA-CSO and BA for Carlier's instances

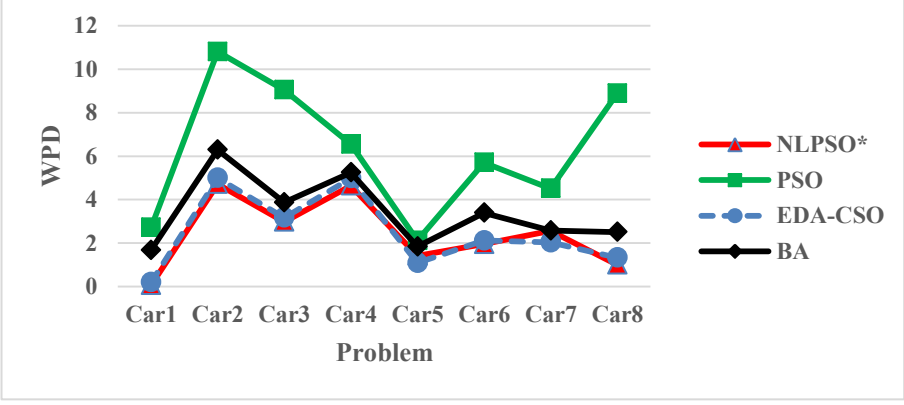

FIGURE 6. WPD of NLPSO*, PSO, EDA-CSO and BA for Carlier's instances

5. Conclusion

In this work, the solution of the PFSP with minimizing the makespan using a hybrid NLPSO* is presented and discussed. In this algorithm, the N-NEH+ is applied to generate better initial population, since the algorithm that starts from good quality initial solution leads to better results. Then the PSO algorithm is used and the obtained solutions are improved by ILS. The experimental study is conducted with well-known benchmark problems of PFSP instances, including Taillard and Carlier instances. The experimental results prove the outperforming of the NLPSO* algorithm on other algorithms, including the recent EDA-CSO algorithm. For future work, the NLPSO* algorithm can used to solve other COPs such as vehicle routing problem, knapsack problem, etc. Also, proposing other heuristics or local search method to improve the efficiency of various metaheuristics could be consider as future study.

References

[1] M. L. Pinedo, "*Scheduling*," Cham: Springer International Publishing, 2016. doi: 10.1007/978-3-319-26580-3.

[2] Al Zuwaini, M. K., Abdul Razaq TS, and S. K. Al Saidy. "Flow-Shop scheduling Problem to Minimize Total Weighted Late Work." *Journal of Al-Qadisiyah for computer science and mathematics* 1.1 (2009): 147-160.

[3] M. R. Garey, D. S. Johnson, and R. Sethi, "The complexity of flowshop and jobshop scheduling," *Mathematics of operations research*, vol. 1, no. 2, pp. 117-129, 1976, doi: org/10.1287/moor.1.2.117.

[4] J. Carlier and I. Rebaï, "Two branch and bound algorithms for the permutation flow shop problem," *European Journal of Operational Research*, vol. 90, no. 2, pp. 238–251, 1996, doi: 10.1016/0377-2217(95)00352-5.

[5] S. M. Johnson, "Optimal two-and three-stage production schedules with setup times included," *Naval research logistics quarterly*, vol. 1, no. 1, pp. 61-68, 1954, doi: org/10.1002/nav.3800010110.

[6] H. G. Campbell, R. A. Dudek, and M. L. Smith, "A Heuristic Algorithm for the n Job, m Machine Sequencing Problem," *Management Science*, vol. 16, no. 10, p. B-630-B-637, Jun. 1970, doi: 10.1287/mnsc.16.10.B630.

[7] M. Nawaz, E E. Enscore, I. Ham, "A heuristic algorithm for the m-machine, n-job flow-shop sequencing problem," *OMEGA, The International Journal of Management Science*, vol. 11, no. 1, pp. 91–95, 1983, doi: 10.1016/0305-0483(83)90088-9.

[8] C. Sauvey and N. Sauer, "Two NEH heuristic improvements for flowshop scheduling problem with makespan criterion," *Algorithms*, vol. 13, no. 5, pp. 1-14, 2020, doi: 10.3390/A13050112.

[9] G. A. D. P. K and Komarudin, "Development of NEH for Permutation Flowshop Scheduling Problem," in *Proceedings of the 3rd Asia Pacific Conference on Research in Industrial and Systems Engineering 2020*, Jun. 2020, pp. 278-283, doi: 10.1145/3400934.3400985.

[10] R. Puka, J. Duda, A. Stawowy, and I. Skalna, "N-NEH+ algorithm for solving permutation flow shop problems," *Computers and Operations Research*, vol. 132, Aug. 2021, doi: 10.1016/j.cor.2021.105296.

[11] Hasoon, Jamal N., and Rehab Hassan. "Solving Job Scheduling Problem Using Fireworks Algorithm." *Journal of Al-Qadisiyah for computer science and mathematics* 11.2 (2019): 1-8.

[12] I. Osman and C. Potts, "Simulated annealing for permutation flow-shop scheduling," *Omega*, vol. 17, no. 6, pp. 551-557 pages, Jan. 1989, doi: 10.1016/0305-0483(89)90059-5.

[13] C. R. Reeves, "A genetic algorithm for flowshop sequencing," *Computers & Operations Research*, vol. 22, no. 1, pp. 5-13 pages, Jan. 1995, doi: 10.1016/0305-0548(93)E0014-K.

[14] Widmer, Marino, and Alain Hertz. "A New Heuristic Method for the Flow Shop Sequencing Problem." *European Journal of Operational Research,* 1989. 41(2): 186–193, doi: 10.1016/0377-2217(89)90383-4.

[15] T. Stützle *et al.*, "An Ant Approach to the Flow Shop Problem," *In Proceedings of the 6th European Congress on Intelligent Techniques & Soft Computing (EUFIT'98*, no. February 1970, pp. 1560–1564, 1997.

[16] M. Al-Behadili, D. Ouelhadj, and D. Jones, "*Multi-objective particle swarm optimisation for robust dynamic scheduling in a permutation flow shop*,"vol. 557. 2017. doi: 10.1007/978-3-319-53480-0_49.

[17] M. F. Tasgetiren, M. Sevkli, Y.-C. Liang, and G. Gencyilmaz, "Particle Swarm Optimization Algorithm for Permutation Flowshop Sequencing Problem," 2004, pp. 382–389. doi: 10.1007/978-3-540-28646-2_38.

[18] M. al Behadili, H. Zaki, and K. al Yasiri, "Locust swarm optimisation for the permutation flow shop scheduling problem," *International Journal of Mathematics in Operational Research*, vol. 18, no. 4, p. 545, 2021, doi: 10.1504/IJMOR.2021.114207.

[19] A. N. H. Zaied, M. M. Ismail, and S. S. Mohamed, "Permutation flow shop scheduling problem with makespan criterion: literature review," vol. 99, no. 4, pp. 830-848, 2021.

[20] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proceedings of ICNN'95 - International Conference on Neural Networks*, 1995, vol. 4, pp. 1942–1948. doi: 10.1109/ICNN.1995.488968.

[21] L. Zhang and J. Wu, "A PSO-Based Hybrid Metaheuristic for Permutation Flowshop Scheduling Problems," *The Scientific World Journal*, vol. 2014, pp. 1-8, 2014, doi: 10.1155/2014/902950.

[22] T. R. Ramanan, M. Iqbal, and K. Umarali, "A particle swarm optimization approach for permutation flow shop scheduling problem," *International Journal for Simulation and Multidisciplinary Design Optimization*, vol. 5, p. A20, 2014, doi: 10.1051/smdo/2013006.

[23] T. Stützle and R. Ruiz, "Iterated Local Search," in *Handbook of Heuristics*, vol. 1-2 volume, Cham: Springer International Publishing, 2018, pp. 579-605 pages. doi: 10.1007/978-3-319-07124-4_8.

[24] O. Martin, S. W. Otto, and E. W. Felten, "*Large-Step Markov Chains for the Traveling Salesman Problem*," Citeseer, 1991.

[25] T. Stützle, "Applying iterated local search to the permutation flow shop problem," Citeseer, 1998.

[26] B. Qian, L. Wang, R. Hu, and W. Wang, "A hybrid differential evolution method for permutation flow-shop scheduling," *Int J Adv Manuf Technol*, vol. 38, pp. 757–777, 2008, doi: 10.1007/s00170-007-1115-8.

[27] E. Taillard, "Benchmarks for basic scheduling problems," *European Journal of Operational Research*, vol. 64, no. 2, pp. 278–285, 1993, doi: 10.1016/0377-2217(93)90182-M.

[28] X. Pei and Y. Tang, "Improved cat swarm optimization for permutation flow shop scheduling problem," in *Journal of Physics: Conference Series*, 2021, vol. 2010, no. 1 issue, p. 12018, doi: 10.1088/1742-6596/2010/1/012018.