Journal of Soft Computing and Artificial Intelligence
Research Article 2(2): pp:85-97 (2021)
Received: October 22, 2021 Accepted: November 15, 2021
JSCAI
ISSN: 2717-8226

Application of Social Spider Optimization for Permutation Flow Shop Scheduling Problem

Mohamed Kurdi

Computer Engineering Department, Aydın Adnan Menderes University, Turkey
Corresponding author: First A. Author (e-mail: mohamed.kurdi@adu.edu.tr).

ABSTRACT Permutation flow shop scheduling problem (PFSP) is an NP-complete problem with a wide range of applications in many real-world applications. Social spider optimization (SSO) is a swarm intelligence algorithm proposed for continuous optimization problems. Recently, SSO has received increased interest in the field of combinatorial optimization as well. For this reason, in this paper, SSO algorithm is proposed to solve the PFSP with make span minimization. The proposed algorithm has been tested on 141 well-known benchmark instances and compared against six other conventional and best-so-far metaheuristics. The obtained results show that SSO outperforms some of the compared works although they are hybrid methods.

KEYWORDS: Metaheuristic, Optimization, Flow Shop Scheduling, Social Spider, Swarm Intelligence.

1. INTRODUCTION

Flow shop is a renowned manufacturing layout in which a set of jobs should be processed, in the same order, on a set of machines. The flow shop scheduling problem considers the sequence of the jobs over the machines with respect to a certain performance measure, such as makespan, maximum lateness, or total weighted completion time. If each machine should process the jobs in the same order, the problem is called as permutation flow shop scheduling problem (PFSP) [1]. PFSP is an NP-complete problem that has several real-life application fields, such as computing designs, production, information processing, communications, and transportation [2].

Due to its complexity and practical relevance, PFSP has been addressed by a considerable number of metaheuristic algorithms. These algorithms include discrete jaya algorithm [3], genetic-shuffled frog-leaping [4], Iterative beam search [5], evolutionary algorithm [6], hybrid backtracking search [2], shuffled complex evolution [7], genetic algorithm [8-10] bacterial foraging optimization [11], bat algorithm (BA) [12], rhinoceros search [13], biogeography-based optimization [14], differential evolution [15] [16], harmony search [17], cuckoo search [18], scatter search [19], iterated greedy [20], monkey search [21], and ant colony optimization (ACO) [22].

Social spider optimization (SSO) is a new swarm intelligence algorithm that has been proposed for continuous optimization problems [23]. However, recently, there has been an increased interest in applying it for solving combinatorial problems. Works such as [24-28] have shown it as a promising area of research for combinatorial problems.

In this paper, SSO is proposed for the PFSP with makespan minimization. The aim is to examine the effectiveness of SSO on PFSP as it has been widely used as a benchmark problem to validate the effectiveness of many optimization algorithms. To the best of our knowledge, there is no published work to address the PFSP by using this algorithm. The remainder of this paper is structured as follows. The problem definition is given in the next section. Section 3 describes the proposed algorithm. The computational results are reported in Section 4. The conclusion and future works are presented in Section 5.

2. PROBLEM DEFINITION

Suppose that *n* jobs $\{J_i\}_{i=1}^{n}$ need to be sequentially processed on *m* machines $\{M_k\}_{k=1}^{m}$. Each job J_i is composed of *m* operations $(O_{i1}, O_{i2}, \ldots, O_{im})$. All jobs should have the same processing order on each machine. O_{ik} represents the operation of job J_i on machine M_k which needs using M_k solely for a specified continued time called P_{ik} (pre-emption is not allowed). Let $\pi = \{\pi_1, \pi_2, \ldots, \pi_n\}$ be a permutation of the jobs in which π_i denotes the index of the job placed at the *i*th position of *π*. Then, the completion time of each job $C(\pi_i, k)$, $i = 1, \ldots, n$ can be calculated by the following set of recursive formulas [29].

$$C(\pi_1, 1) = p_{\pi_1 1}, \tag{1}$$

$$C(\pi_i, 1) = C(\pi_{i-1}, 1) + p_{\pi_i 1}, \quad i = 2, \ldots, n, \tag{2}$$

$$C(\pi_1, k) = C(\pi_1, k-1) + p_{\pi_1 k}, \quad k = 2, \ldots, \mathrm{m}, \tag{3}$$

$$C(\pi_i, k) = \max\{C(\pi_{i-1}, k), C(\pi_i, k-1)\} + p_{\pi_i k}, \quad i = 2, \ldots, n, \quad k = 2, \ldots, \mathrm{m}, \tag{4}$$

Then, the makespan is given by

$$C_{\max}(\pi) = C(\pi_n, m). \tag{5}$$

Therefore, the problem turns into finding a permutation π^* in the set of all permutations Π such that

$$C_{\max}(\pi^*) \le C(\pi_n, m) \quad \forall \pi \in \Pi. \tag{6}$$

3. SOCIAL SPIDER OPTIMIZATION

3.1 BACKGROUND

The social spider optimization (SSO) suggested by Cuevas et al. [30] is a recent swarm intelligence algorithm inspired by the collaborative behavior of social spider colony. This behavior can be summarized in the following way. A social spider colony is composed of two essential components: spiders and communal web. The spiders are grouped into two different categories: males and females. Based on its gender, each spider collaborates in different tasks such as building and maintaining the communal web, prey capturing, and mating. Interactions among spiders are either direct or indirect. Direct interactions involve physical contact, such as mating. Indirect interactions take place by using the communal web as a medium of communication that transfers important information, such as the size of the trapped preys and characteristics of the neighboring members. This information, which is encoded as small vibrations, is a crucial portion for the mutual coordination among the spiders. The intensity of these vibrations is dependent on the locations and weights of the spiders that generate them [30].

This collaborative behavior can be utilized for solving optimization problems by simulation as follows. SSO imagines the communal web as the search space. The location of a spider in the communal web symbolizes a solution of the problem in the search space. Each spider is given a value for its weight that depends on the fitness of the solution that is represented by it. Unlike most of the existent swarm algorithms, SSO models two different search agents (spiders): males for performing extensive exploitation and females for performing efficient exploration. This allows not only to imitate the collaborative behavior of the colony in a better realistic way, but also to utilize computational operators that can delay the premature convergence and somehow strike an exploration–exploitation balance [30]. The search process is controlled by three operators: the female cooperative operator that changes the locations of females, the male cooperative operator that changes the locations of males, and the mating operator that produces new spiders that are located on new locations in the search space [30]. Figure 1 describes the general framework of the proposed SSO. As it can be noticed, a supplementary step has been added to the classical SSO, in which the locations of the spiders that exist in the continuous space are converted to their equivalent locations in the combinatorial space. What follows is the mathematical modelling of the proposed SSO [30].

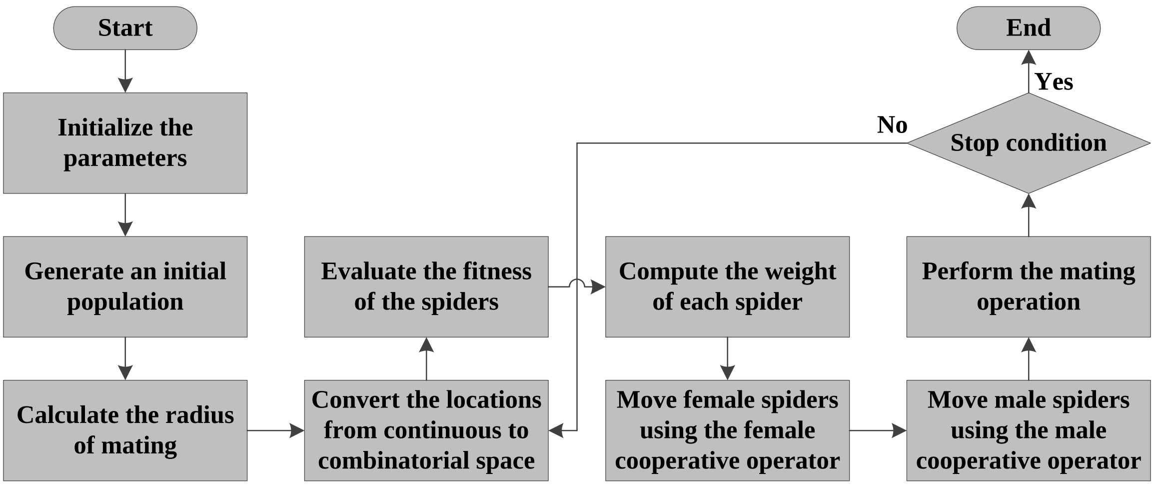

Figure 1. The proposed SSO.

3.2 GENDER ASSIGNATION

The first step in SSO is determining the number of female and male spiders. Since social spider colonies are highly female-biased ones, the number of females N_f is selected at random in the range 65–90% of the population. Let N be the total number of spiders, then N_f is calculated using the following formula.

$$N_f = floor[(0.9 - \text{rand} \cdot 0.25) \cdot N] \tag{7}$$

where rand is a random number in the range (0, 1), and the floor function is used to convert a real number into an integer number. The number of male spiders N_m is calculated as the complement between N and N_f using the following formula.

$$N_m = N - N_f \tag{8}$$

For that reason, the entire population of spiders S is split into female spiders group $F = \{f_1, f_2, \dots f_{N_f}\}$ and male spiders group $M = \{m_1, m_2, \dots m_{N_m}\}$, where $S = F \cup M (S = \{s_1, s_2, \dots s_N\})$, such that $S = \{s_1 = f_1, s_2 = f_2, \dots, s_{N_f} = f_{N_f}, s_{N_f+1} = m_1, s_{N_f+2} = m_2, \dots, s_N = m_{N_m}\}$.

3.3 COLONY INITIALIZATION

The locations of all spiders are initialized at random. Each spider location, f_i or m_i, is an n-dimensional vector that represents the optimization variables, where n is the number of jobs in PFSP. The n components of each vector are uniformly distributed between the predefined lower initial parameter bound p_j^{low} and upper parameter bound p_j^{high}. These values are calculated using the following formulas.

$$f_{i,j}^0 = p_j^{low} + rand(0,1) \cdot \left(p_j^{high} - p_j^{low}\right) \quad i = 1,2, \dots, N_f;\ j = 1,2, \dots, n \tag{9}$$

$$m_{k,j}^0 = p_j^{low} + rand(0,1) \cdot \left(p_j^{high} - p_j^{low}\right) \quad k = 1,2, \dots, N_m;\ j = 1,2, \dots, n \tag{10}$$

where j refers to the index of a variable, i refers to the index of a female individual, k refers to the index of a male individual, the value 0 indicates that the individuals belong to the initial population, and rand(0,1) is a function used to generate a random number in the range (0,1). Hence, $f_{i,j}$ is the jth job of the ith female spider and $m_{k,j}$ is the jth job of the kth male spider.

3.4 CONTINUOUS TO COMBINATORIAL TRANSFORMATION

Since SSO works only on real numbers encoding, the random keys discretization method, which was initially proposed in [31], is utilized to transfer the locations of spiders from the continuous space to their corresponding locations in the combinatorial space.

3.5 WEIGHT CALCULATION AND ASSIGNATION

In the biological metaphor, the size of a spider is the characteristic that estimates its ability to do its assigned duties well. In SSO, each individual i of the population S is assigned a weight w_i that depends on the quality of the solution it symbolizes (irrespective of gender). The weight of each spider is calculated by using the following formula.

$$w_i = \frac{C_{max}(s_i) - C_{max}^{worst}}{C_{max}^{best} - C_{max}^{worst}} \tag{11}$$

where $C_{max}(s_i)$ is the makespan value obtained by decoding the permutation of jobs that corresponds to the location of the spider s_i, and C_{max}^{best} and C_{max}^{worst} are the makespan values of the best and worst individuals in the population, respectively.

3.6 MODELING OF THE VIBRATIONS

The vibrations observed by a spider depend on the distance and weight of the spiders that generate them. In SSO, the vibrations observed by spider i as a result of the information sent by another spider j are modeled according to the following formula.

$$Vib_{ij} = w_j \cdot e^{-d_{i,j}^2} \tag{12}$$

where $d_{i,j}$ is the Euclidean distance between the two spiders. SSO considers that each spider i is supposed to be able to detect vibrations from three other spiders. These spiders are the closest one that has a higher weight $Vibc_i$, the best spider in the colony $Vibb_i$, and the nearest female spider $Vibf_i$.

3.7 FEMALE COOPERATIVE OPERATOR

Female spiders are commonly attracted to the other (male or female) spiders in accordance with their vibrations transmitted over the communal web. Strong vibrations are generated by either big spiders or other neighboring spiders lying nearby the spider that is perceiving them. The decision of attraction or repulsion is made according to an internal state which is affected by several factors such as reproduction cycle, curiosity, and other random phenomena. This behavior is modeled by the female cooperative operator which is defined as follows.

$$f_i^{k+1} = \begin{cases} f_i^k + \alpha \cdot Vibc_i \cdot (s_c - f_i^k) + \beta . Vibb_i \cdot (s_b - f_i^k) \\ \qquad + \delta \cdot \left(\text{rand} - \frac{1}{2}\right) \quad if\ r_m < PF \\ f_i^k - \alpha \cdot Vibc_i \cdot (s_c - f_i^k) - \beta . Vibb_i \cdot (s_b - f_i^k) \\ \qquad + \delta \cdot \left(\text{rand} - \frac{1}{2}\right) \quad if\ r_m > PF \end{cases} \tag{13}$$

where α, β, δ, r_m, and rand are random numbers in the range (0, 1), k represents the iteration counter, PF represents the threshold value used for determining whether an attraction or repulsion movement is produced, s_c is the closest spider to spider i that has a higher weight, and s_b is the best spider in the population.

3.8 MALE COOPERATIVE OPERATOR

Male spiders consider themselves as a group of alpha males which dominate the colony resources. Hence, the males are divided into two sets: dominant and non-dominant males. Dominant males usually have superior fitness attributes to non-dominant males. In addition, dominant males are enticed to the nearest female spiders in the communal web. On the hand, non-dominant males tend to localize on the center of the set of males as a strategy to benefit from the resources left over from the dominant males. For implementing such phenomena, the set of males $M = \{m_1, m_2, \ldots m_{N_m}\}$ is sorted according to their weight values in increasing order. The male spider that is located in the middle, whose weight value is w_{N_f+m}, is treated as the median male spider. The male spiders whose weight values are bigger the median value are treated as members of the set of dominant males *D*, and the rest of males are treated as members of the non-dominant set *ND*. In accordance to this, the variation of locations for the male spiders is modeled by the following formula.

$$m_i^{k+1} = \begin{cases} m_i^k + \alpha \cdot Vibf_i \cdot (s_f - m_i^k) + \delta \cdot \left(\text{rand} - \frac{1}{2}\right) \\ \qquad if\ w_{N_f+i} > w_{N_f+m} \\ m_i^k + \alpha \cdot \left(\frac{\sum_{h=1}^{N_m} m_h^k \cdot w_{N_f+h}}{\sum_{h=1}^{N_m} w_{N_f+h}} - m_i^k\right) \\ \qquad if\ w_{N_f+i} \leq w_{N_f+m} \end{cases} \tag{14}$$

where s_f is the nearest female spider to the male i, and $\sum_{h=1}^{N_m} m_h^k \cdot w_{N_f+h} / \sum_{h=1}^{N_m} w_{N_f+h}$ is the weighted mean of the set of male spiders *M*.

3.9 MATING OPERATOR

Mating is performed by dominant males and females. A dominant male spider m_g ($m_g \in D$) can mate with a set of female spiders E^g if they exist within a specific radius *r* (range of mating). This radius, which depends on the size of the search space, is calculated by the following formula.

$$r = \frac{\sum_{j=1}^{n}(p_j^{high} - p_j^{low})}{2 \cdot n} \tag{15}$$

If $E^g = \emptyset$, the mating process is revoked. Otherwise, mating occurs and a new brood s_{new} is generated by taking in consideration all elements of the set T^g which is the union $m_g \cup E^g$. During the mating process, the weight of each element of the set T^g controls the probability of its impact on the new brood. The members with higher weight values have higher probabilities to impact the new spider than those with lighter weight values. The impact probability Ps_i of each member is calculated by the roulette wheel method, which is defined as follows.

$$Ps_i = \frac{w_i}{\sum_{j \in T^g} w_j} \tag{16}$$

where $i \in T^g$. When a new spider is born, it is immediately compared to the spider holding the worst weight of the whole colony. If the new spider is superior to the worst spider, it replaces it. Otherwise, the worst spider is kept and the new spider is neglected.

4. COMPUTATIONAL RESULTS

SSO has been implemented in C++. The tests have been run on a personal computer with 3.4 GHz CPU and 8 GB RAM. Two well-known benchmark sets were used for the evaluation. The first set is Reeves's set [32]. This set is composed of 21 instances that are divided into 7 equal subsets of different sizes. These sizes range from 20 jobs and 5 machines up to 75 jobs and 20 machines. The second set is Taillard's set [33]. This set is composed of 120 instances that are divided into 12 equal subsets of different sizes. These sizes range from 20 jobs and 5 machines up to 500 jobs and 20 machines. In order to tune the parameters, preliminary experiments have been done on 19 instances selected randomly from the two benchmark sets (one of each subset). Consequently, the parameters have been set as follows: $N=100$, $PF=0.7$, α, β, δ, and r_m are as assigned random values between zero and one, and the maximum number of iterations is 10000.

SSO was compared with conventional and best-so-far metaheuristics. The metaheuristics that were compared using Reeves's set are chaotic local search based bacterial foraging algorithm (CLS-BFO) [11], shuffled complex evolution algorithm with opposition-based learning (SCE-OBL) [7], and Genetic algorithm integrated with artificial chromosomes (ACGA) [34]. The metaheuristics that were compared using Taillard's set are memetic algorithm with novel semi-constructive evolution operators (MASC) [8] which is one of the best-so-far approaches for the problem, hybrid whale optimization algorithm based on local search strategy (HWA) [35], and self-guided differential evolution with neighborhood search (NS-SGDE) [16]. SSO was run 10 independent times and the best solution (BS) among them was considered for the comparison. Table 1 presents the results on Reeves's set. It lists instance name, instance size (number of jobs * number of machines), best known solution (BKS), BS of each algorithm. From Table 1, it can be seen that SSO is able to obtain better solutions than the others on most of the instances. However, to make a precise comparison, the relative error of BS for each instance (*PE*), and the average of *PE* for the whole set of instances (*APE*) were calculated for each algorithm as follows.

$$PE = 100 \times \left(\frac{BS-BKS}{BKS}\right) \quad (17)$$

$$APE = \left(\sum_{i=1}^{21}\left(\frac{BS_i-BKS_i}{BKS_i}\right) \times 100\right)/21 \quad (18)$$

Table 2 presents the results. It lists *APE* values for SSO and the other algorithms (OA), and the percentage improvement (*PI*) achieved by SSO in *APE* values with respect to each of the other algorithms, which was calculated as follows.

$$PI = 100 \times (OA_{APE} - SSO_{APE})/OA_{APE} \quad (19)$$

From Table 2, it can be seen that SSO has the lowest *APE* value and produces relative improvements to all the other algorithms. This shows that SSO is an effective approach since the compared works are hybrid methods. To further verify the effectiveness of SSO, the one-way ANOVA test was applied on the *PE* values. Figure 2 shows the results. From Figure 2, it can be seen that SSO outperforms the compared algorithms, and the resulting *p*-value is 0,009 which implies that the algorithms are significantly statistically different with each other.

Table 1. The computational results on Reeves's set.

Instance	*n*m*	BKS	SSO	SCE-OBL	CLS-BFO	ACGA
Rec1	20x5	1245	1247	1249	1249	1249
Rec3	20x5	1109	1109	1111	1111	1109
Rec5	20x5	1242	1245	1245	1245	1245
Rec7	20x10	1566	1566	1584	1584	1566
Rec9	20x10	1537	1537	1545	1545	1537
Rec11	20x10	1431	1431	1431	1449	1431
Rec13	20x15	1930	1935	1963	1968	1935
Rec15	20x15	1950	1968	1993	1993	1950
Rec17	20x15	1902	1923	1944	1954	1911
Rec19	30x10	2093	2117	2156	2139	2099
Rec21	30x10	2017	2017	2064	2059	2046
Rec23	30x10	2011	2030	2067	2073	2021
Rec25	30x15	2513	2566	2584	2638	2545
Rec27	30x15	2373	2397	2445	2443	2396
Rec29	30x15	2287	2333	2364	2408	2304
Rec31	50x10	3045	3104	3179	3180	3105
Rec33	50x10	3114	3118	3154	3187	3140
Rec35	50x10	3277	3277	3281	3292	3277
Rec37	75x20	4890	5096	5327	5422	5193
Rec39	75x20	5043	5185	5391	5465	5276
Rec41	75x20	4910	5135	5334	5436	5208

Table 2. *APE* of SSO and the other works on Reeves's set.

Algorithm	***APE***		***PI***
	OA (%)	**SSO (%)**	**SSO (%)**
CLS-BFO	2,675	1,123	58
SCE-OBL	3,255	1,123	65
ACGA	1,247	1,123	10

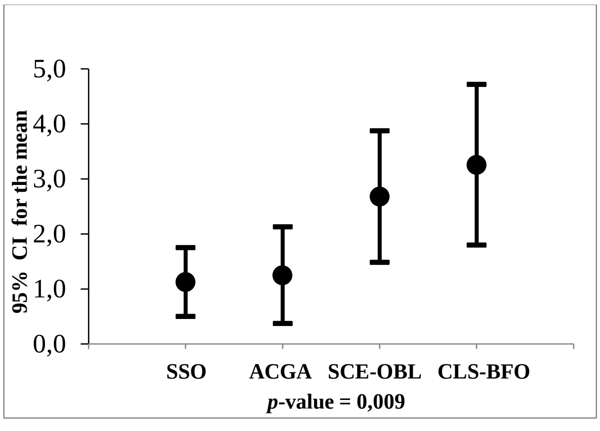

Figure 2. Means and 95% confidence intervals on Reeves's set.

Table 3. The computational results on ST1.

Instance	*n*m*	BKS	SSO	NS-SGDE	HWA	MASC
Ta001	20x5	1278	1282	1278	1278	1278
Ta002	20x5	1359	1359	1359	1359	1359
Ta003	20x5	1081	1088	1081	1081	1081
Ta004	20x5	1293	1300	1293	1293	1293
Ta005	20x5	1235	1237	1235	1235	1235
Ta006	20x5	1195	1195	1195	1195	1195
Ta007	20x5	1239	1243	1239	1239	1239
Ta008	20x5	1206	1206	1206	1206	1206
Ta009	20x5	1230	1231	1230	1230	1230
Ta010	20x5	1108	1108	1108	1108	1108
Ta011	20x10	1582	1598	1582	1582	1582
Ta012	20x10	1659	1682	1659	1659	1659
Ta013	20x10	1496	1513	1496	1496	1496
Ta014	20x10	1377	1395	1377	1377	1377
Ta015	20x10	1419	1440	1419	1419	1419
Ta016	20x10	1397	1404	1397	1397	1397
Ta017	20x10	1484	1493	1484	1484	1484
Ta018	20x10	1538	1555	1538	1538	1538
Ta019	20x10	1593	1606	1593	1593	1593
Ta020	20x10	1591	1611	1591	1591	1591
Ta021	20x20	2297	2329	2297	2297	2297
Ta022	20x20	2099	2125	2099	2099	2099
Ta023	20x20	2326	2350	2326	2326	2326
Ta024	20x20	2223	2244	2223	2223	2223
Ta025	20x20	2291	2309	2291	2291	2291
Ta026	20x20	2226	2244	2228	2226	2226
Ta027	20x20	2273	2296	2273	2273	2273
Ta028	20x20	2200	2229	2200	2200	2200
Ta029	20x20	2237	2252	2237	2237	2237
Ta030	20x20	2178	2195	2178	2178	2178
Ta031	50x5	2724	2724	2724	2724	2724
Ta032	50x5	2834	2839	2834	2834	2834
Ta033	50x5	2621	2621	2621	2621	2621
Ta034	50x5	2751	2753	2751	2751	2751
Ta035	50x5	2863	2863	2863	2863	2863
Ta036	50x5	2829	2832	2829	2829	2829
Ta037	50x5	2725	2725	2725	2725	2725
Ta038	50x5	2683	2703	2683	2683	2683
Ta039	50x5	2552	2561	2552	2552	2552
Ta040	50x5	2782	2782	2782	2782	2782
Ta041	50x10	2991	3053	3021	3021	3024
Ta042	50x10	2867	2938	2896	2891	2882
Ta043	50x10	2839	2890	2888	2869	2852
Ta044	50x10	3063	3071	3075	3063	3063
Ta045	50x10	2976	3024	3027	3001	2982
Ta046	50x10	3006	3050	3029	3006	3006
Ta047	50x10	3093	3133	3124	3126	3099
Ta048	50x10	3037	3046	3055	3046	3038
Ta049	50x10	2897	2927	2928	2897	2902
Ta050	50x10	3065	3131	3092	3078	3077
Ta051	50x20	3850	3974	3916	3876	3889
Ta052	50x20	3704	3808	3744	3715	3720
Ta053	50x20	3640	3772	3702	3653	3667
Ta054	50x20	3720	3849	3793	3755	3754
Ta055	50x20	3610	3746	3677	3649	3644
Ta056	50x20	3681	3795	3743	3703	3708
Ta057	50x20	3704	3835	3784	3723	3754
Ta058	50x20	3691	3829	3757	3704	3711
Ta059	50x20	3743	3870	3795	3763	3772
Ta060	50x20	3756	3875	-	3767	3778

Table 4. The computational results on ST2.

Instance	*n**m*	BKS	SSO	NS-SGDE	HWA	MASC
Ta061	100x5	5493	5493	5493	5493	5493
Ta062	100x5	5268	5284	5283	5268	5268
Ta063	100x5	5171	5193	5182	5175	5175
Ta064	100x5	5014	5023	5018	5018	5014
Ta065	100x5	5250	5250	5253	5250	5250
Ta066	100x5	5135	5137	5135	5135	5135
Ta067	100x5	5246	5259	5246	5246	5246
Ta068	100x5	5094	5106	5094	5094	5094
Ta069	100x5	5448	5467	5448	5448	5448
Ta070	100x5	5322	5338	5322	5324	5322
Ta071	100x10	5770	5787	5784	5776	5770
Ta072	100x10	5349	5379	5362	5362	5349
Ta073	100x10	5676	5691	5691	5691	5677
Ta074	100x10	5781	5849	5826	5825	5781
Ta075	100x10	5467	5513	5503	5491	5467
Ta076	100x10	5303	5328	5308	5308	5304
Ta077	100x10	5595	5662	5610	5608	5596
Ta078	100x10	5617	5695	5630	5630	5625
Ta079	100x10	5871	5916	5882	5891	5875
Ta080	100x10	5845	5903	5881	5848	5845
Ta081	100x20	6202	6377	6360	6280	6257
Ta082	100x20	6183	6360	6278	6278	6223
Ta083	100x20	6271	6450	6405	6368	6325
Ta084	100x20	6269	6393	6394	6350	6303
Ta085	100x20	6314	6477	6452	6377	6380
Ta086	100x20	6364	6544	6461	6430	6431
Ta087	100x20	6268	6439	6385	6354	6306
Ta088	100x20	6401	6603	6496	6515	6472
Ta089	100x20	6275	6451	6428	6396	6330
Ta090	100x20	6434	6625	6538	6527	6456
Ta091	200x10	10862	10947	10887	10885	10872
Ta092	200x10	10480	10542	10555	10512	10487
Ta093	200x10	10922	11005	10980	10965	10922
Ta094	200x10	10889	10939	10917	10889	10889
Ta095	200x10	10524	10537	10537	10524	10526
Ta096	200x10	10326	10377	10357	10375	10330
Ta097	200x10	10854	10908	10929	10868	10868
Ta098	200x10	10730	10798	10798	10751	10731
Ta099	200x10	10438	10478	10465	10465	10454
Ta100	200x10	10675	10758	10727	10727	10680
Ta101	200x20	11195	11418	11468	11335	11280
Ta102	200x20	11203	11488	11487	11517	11272
Ta103	200x20	11281	11559	11549	11481	11378
Ta104	200x20	11275	11465	11553	11405	11376
Ta105	200x20	11259	11444	11438	11374	11310
Ta106	200x20	11176	11421	11445	11335	11265
Ta107	200x20	11360	11593	11596	11438	11430
Ta108	200x20	11334	11597	11592	11530	11398
Ta109	200x20	11192	11457	11485	11439	11266
Ta110	200x20	11288	11567	11607	11499	11355
Ta111	500x20	26059	26493	26420	26388	26187
Ta112	500x20	26520	26953	26942	26714	26779
Ta113	500x20	26371	26787	26729	26648	26496
Ta114	500x20	26456	26817	26751	26656	26618
Ta115	500x20	26334	26698	26643	26579	26500
Ta116	500x20	26477	26874	26832	26666	26647
Ta117	500x20	26389	26691	26609	26594	26529
Ta118	500x20	26560	26913	26925	26711	26772
Ta119	500x20	26005	26425	26326	26228	26223
Ta120	500x20	26457	26905	26766	26695	26617

For the clarity of presentation, the benchmark set of Taillard was divided into two subsets. The first subset is named ST1 and contains the instances where the number of operations <=100. The second subset is named ST2 and contains the instances where the number of operations >100. Table 3 and Table 4 show BS obtained by the compared works on these two subsets.

From Table 3 and Table 4, it can be seen that the performance of SSO deteriorates on this set. However, in order to provide a more thorough comparison, the one-way ANOVA test was applied on the *PE* values over the whole set of instances. Figure 3 shows the results. From Figure 3, it can be noticed that MASC performs better than all the compared works. This is because MASC combines the complementary strengths of population-based (global search), constructive methods, and single-point (local search) methods. The four works can be sorted according to the resulting *APE* values from the best to the worst as follows: MAC 0,278, HWA 0,460, NS-SGDE 0,750, SSO 1,268. It can be also noticed that the resulting *p*-value is 3,83E-22 which indicates that there is a statistically significant difference between the compared algorithms.

The deterioration of SSO performance on this set can be due to two reasons. First, Taillard's set is more difficult to solve than Reeves's set. Second, SSO didn't use any problem-specific operators or external single-point mechanisms. Therefore, it cannot intensify the search in the promising areas of the search space, i.e. the neighborhood of good solutions.

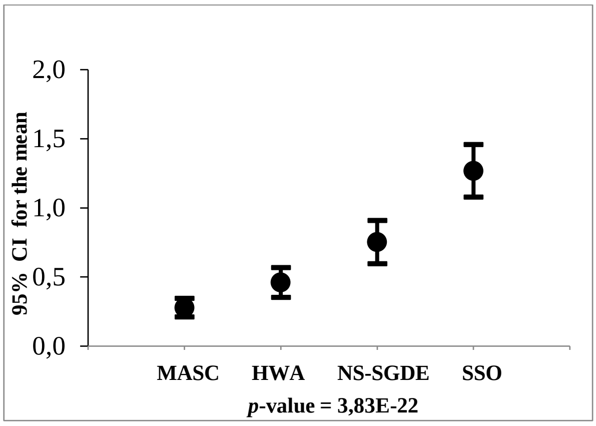

Figure 3. Means and 95% confidence intervals on Taillard's set.

5. CONCLUSION AND FUTURE WORKS

In this work, SSO algorithm is proposed to solve the permutation flow shop problem with makespan minimization. The aim is to investigate the effectiveness of SSO in solving this combinatorial problem. The well-known benchmark sets of Reeves and Taillard were used for the evaluation. Six other conventional and best-so-far algorithms were used for the comparison. The computational results show that SSO outperforms three of them although they are hybrid methods. However, the results also show that it fails to compete with the best-so-far algorithms. This is due to the fact that the best-so-far approaches are memetic algorithms that combine the advantages of population-based and single-point algorithms. Therefore, it is expected that SSO will perform very well if it is hybridized with a single-point algorithm, and this hybridization could be one of the future works of this research.

REFERENCES

[1] V. Fernandez-Viagas, R. Ruiz, J. M. Framinan, "A new vision of approximate methods for the permutation flowshop to minimise makespan: State-of-the-art and computational evaluation", *Eur. J. Oper. Res*., vol. 257, no. 3, pp. 707-721, Mar. 2017.

[2] Q. Lin, L. Gao, X. Li, C. Zhang, "A hybrid backtracking search algorithm for permutation flow-shop scheduling problem", *Comput. Ind. Eng*., vol. 85, pp. 437-446, July. 2015.

[3] N. A. Alawad, B. H. Abed-alguni, "Discrete Jaya with refraction learning and three mutation methods for the permutation flow shop scheduling problem", *J Supercomput*, pp. 1-22, July. 2021.

[4] P. Wu, Q. Yang, W. Chen, B. Mao, H. Yu, "An Improved Genetic-Shuffled Frog-Leaping Algorithm for Permutation Flowshop Scheduling", *Complexity*, 2020.

[5] L. Libralesso, P. A. Focke, A Secardin, V. Jost, "Iterative beam search algorithms for the permutation flowshop", *arXiv preprint arXiv*, Sep. 2020.

[6] C. Y. Hsu, P. C. Chang, M. H. Chen, "A linkage mining in block-based evolutionary algorithm for permutation flowshop scheduling problem", Comput. Ind. Eng., vol. 83, pp. 159-171, May. 2015.

[7] F. Zhao, J. Zhang, J. Wang, C. Zhang, "A shuffled complex evolution algorithm with opposition-based learning for a permutation flow shop scheduling problem", *Int. J. Computer Integr. Manuf.*, vol. 28, no. 11, pp. 1220-1235, Oct. 2015.

[8] M. Kurdi, "A memetic algorithm with novel semi-constructive evolution operators for permutation flowshop scheduling problem", *Appl. Soft Comput.*, vol. 94, Sept. 2020.

[9] M. Bessedik, F. B. S. Tayeb, H. Cheurfi, A. Bliz, "An immunity-based hybrid genetic algorithms for permutation flowshop scheduling problems", *Int. J. Adv. Manuf. Syst*., vol. 85. no 9-12, pp. 2459-2469, Aug. 2016.

[10] F. B. S. Tayeb, M. Bessedik, M. Benbouzid, M. Benbouzid, H. Cheurfi, A. Blizak, "Research on Permutation Flow-shop Scheduling Problem based on Improved Genetic Immune Algorithm with vaccinated offspring", *Procedia Comput. Sci.*, vol. 112, pp. 427-436, 2017.

[11] F. Zhao, Y. Liu, Z. Shao, X. Jiang, C. Zhang, J. Wang, "A chaotic local search based bacterial foraging algorithm and its application to a permutation flow-shop scheduling problem", *Int. J. Computer Integr. Manuf.*, vol. 29, no. 9, pp. 962-981, 2016.

[12] Ö. Tosun, M. K. Marichelvam, "Hybrid bat algorithm for flow shop scheduling problems", *Int. J. Math. Oper.*, vol. 9, no. 1, pp. 125-138, 2016.

[13] S. Deb, Z. Tian, S. Fong, R. Tang, R. Wong, N. Dey, "Solving permutation flow-shop scheduling problem by rhinoceros search algorithm", *Soft Comput.*, vol. 22, no. 18, pp. 6025-6034, 2018.

[14] J. Lin, "A hybrid discrete biogeography-based optimization for the permutation flow shop scheduling problem" *Int. J. Prod. Res.*, vol. 54, no. 16, pp. 4805-4814, 2016.

[15] V. Santucci, M. Baioletti, A. Milani, "Solving permutation flowshop scheduling problems with a discrete differential evolution algorithm", *AI Commun.*, vol. 29, no. 2, pp. 269-286, 2016.

[16] W. Shao, D. Pi, "A self-guided differential evolution with neighborhood search for permutation flow shop scheduling", *Expert Syst. Appl*., vol. 51, pp. 161-176, June. 2016.

[17] F. Zhao, Y. Liu, Y. Zhang, W. Ma, C. Zhang, "A hybrid harmony search algorithm with efficient job sequence scheme and variable neighborhood search for the permutation flow shop scheduling problems", *Eng. Appl. Artif. Intell.*, vol. 65, pp. 178-199, Oct. 2017.

[18] H. Wang, W. Wang, H. Sun, Z. Cui, S. Rahnamayan, S. Zeng, "A new cuckoo search algorithm with hybrid strategies for flow shop scheduling problems", *Soft Comput.*, vol. 21, no. 15, pp. 4297-4307, 2017.

[19] K .Govindan, R. Balasundaram, N. Baskar, P. Asokan, “A hybrid approach for minimizing makespan in permutation flowshop scheduling”, *J. Syst. Sci. Syst. Eng.*, vol. 26, no. 1, pp. 50-76, 2017.

[20] J. Dubois-Lacoste, F. Pagnozzi, T. Stützle, “An iterated greedy algorithm with optimization of partial solutions for the makespan permutation flowshop problem”, *Comput. Oper. Res.*, vol. 81, pp.160-166, May. 2017.

[21] M. K. Marichelvam, Ö. Tosun, M. Geetha, “Hybrid monkey search algorithm for flow shop scheduling problem under makespan and total flow time”, *Appl. Soft Comput*., vol. 55, no. 82-92, 2017.

[22] Y. Zhang, Y. Yu, S. Zhang, Y. Luo, L. Zhang, “Ant colony optimization for Cuckoo Search algorithm for permutation flow shop scheduling problem”, *Syst. Sci. Control. Eng.*, vol. 7, no. 1, pp. 20-27, 2019.

[23] S. M. Lim, K. Y. Leong. “A brief survey on intelligent swarm-based algorithms for solving optimization problems”, in *Nature-inspired Methods for Stochastic, Robust and Dynamic Optimization*, 2018, ch. 4, pp 47–61.

[24] S. Kavitha, P. Venkumar, N . Rajini, P. Pitchipoo, “An Efficient Social Spider Optimization for Flexible Job Shop Scheduling Problem”, *J. Adv. Manuf. Syst.*, vol. 17, no. 2, pp. 181-196, 2018.

[25] Y. Wang, L. Zhu, J. Wang, J. Qiu, “An improved social spider algorithm for the Flexible Job-Shop Scheduling Problem”, in *ICEDIF*, Harbin, China, 2015, pp. 157-162.

[26] M. Kurdi, “A Social Spider Optimization Algorithm for Hybrid Flow Shop Scheduling with Multiprocessor Task”, in *12th NCMCONFERENCES*, Ankara, Turkey, 2018. Available at SSRN 3301792.

[27] G. Zhang, K. Xing, “Memetic social spider optimization algorithm for scheduling two-stage assembly flowshop in a distributed environment”, *Comput. Ind. Eng.*, vol. 125, pp. 423-433, Nov. 2018.

[28] G. Zhou, Y. Zhou, R. Zhao, “Hybrid social spider optimization algorithm with differential mutation operator for the job-shop scheduling problem”, *J. Ind. Manag*., vol. 17, no. 2, pp. 533-548, Mar. 2021.

[29] G. I. Zobolas, C. D. Tarantilis and G. Ioannou, “Minimizing makespan in permutation flow shop scheduling problems using a hybrid metaheuristic algorithm”, *Comput. Oper. Res.*, vol. 36, no. 4, pp. 1249-1267, Apr. 2009.

[30] E. Cuevas, M. Cienfuegos, D. Zaldívar, M. Pérez-Cisneros, ”A swarm optimization algorithm inspired in the behavior of the social-spider”, *Expert Syst. Appl.*, vol 40, no. 16, pp. 6374-6384, Nov. 2013.

[31] J. C. Bean, “Genetic algorithms and random keys for sequencing and optimization”, *ORSA J. Comput.*, vol. 6. no. 2, pp.154-160, 1994.

[32] C. R. Reeves, ”A genetic algorithm for flowshop sequencing”, *Comput. Oper. Res*., vol. 22, no. 1, pp. 5-13, Jan. 1995.

[33] E. Taillard, "Benchmarks for basic scheduling problems”, *Eur. J. Oper. Res*., vol. 64, no. 2, pp. 278-285, 1993.

[34] P. C. Chang, S. H. Chen, C. Y. Fan, C. L. Chan, “Genetic algorithm integrated with artificial chromosomes for multi-objective flowshop scheduling problems”, *Appl. Math. Comput*., vol. 205, no. 2, pp. 550-561, Nov. 2008.

[35] M. Abdel-Basset, G. Manogaran, D. El-Shahat, S. Mirjalili, “A hybrid whale optimization algorithm based on local search strategy for the permutation flow shop scheduling problem”, *Future Gener. Comput. Syst*. vol. 85, pp. 129-145, Aug. 2018.