International Journal of Electrical and Computer Engineering (IJECE)
Vol. 11, No. 3, June 2021, pp. 2266~2274
ISSN: 2088-8708, DOI: 10.11591/ijece.v11i3.pp2266-2274

Adaptation and parameters studies of CS algorithm for flow shop scheduling problem

Driss Belbachir, Fatima Boumediene, Ahmed Hassam, Latéfa Ghomri
Department of Electrical and Electronics Engineering, Faculty of Technology, Abou-bekr Belkaid University, Algeria

Article Info

Article history:

Received Nov 23, 2019
Revised Dec 10, 2020
Accepted Dec 21, 2020

Keywords:

Combinatorial optimization
Cuckoo search algorithm
Flow-shop problem
Meta-heuristics
Scheduling problems

ABSTRACT

Scheduling concerns the allocation of limited resources overtime to perform tasks to fulfill certain criterion and optimize one or several objective functions. One of the most popular models in scheduling theory is that of the flow-shop scheduling. During the last 40 years, the permutation flow-shop sequencing problem with the objective of makespan minimization has held the attraction of many researchers. This problem characterized as Fm/prmu/Cmax in the notation of Graham, involves the determination of the order of processing of n jobs on m machines. In addition, there was evidence that m-machine permutation flow-shop scheduling problem (PFSP) is strongly NP-hard for m ≥3. Due to this NP-hardness, many heuristic approaches have been proposed, this work falls within the framework of the scientific research, whose purpose is to study Cuckoo search algorithm. Also, the objective of this study is to adapt the cuckoo algorithm to a generalized permutation flow-shop problem for minimizing the total completion time, so the problem is denoted as follow: Fm | | Cmax. Simulation results are judged by the total completion time and algorithm run time for each instance processed.

Corresponding Author:

Driss Belbachir
Departments of Electrical and Electronics Engineering
Abou-bekr Belkaid University
Faculty of Technology, Tlemcen, BP 230, 13000 Chetouane Tlemcen, Algeria
Email: driss.belbachir@gmail.com

1. INTRODUCTION

Obtaining economic and reliable schedules constitutes the core of excellence in customer service and of efficiency in manufacturing systems. During the last decades, manufacturing scheduling has been identified to be one among the foremost important decisions in planning and control of commercial plant operations, both in science and in practice. Moreover, scheduling is seen as a decision-making process that's utilized in manufacturing industries also as in service industries [1].

An automated or automatic system is a system performing operations for which the human intervenes is only in the programming of the system and in its setting. Furthermore, the goals of an automated system are to perform tasks that are complex or dangerous for humans, perform difficult or repetitive tasks, or improve efficiency and accuracy. Designed to maintain the efficiency of a "flow shop", automated production systems consisting of a set of numerically controlled machine tools and storage stations connected together by an automated handling system, all commanded and controlled by a central computer [2].

Following a study made by Stecke [3], the problems considered in a AIMS (automated industrial manufacturing systems) are classified into four hierarchical levels, namely: design, planning, scheduling and control problems. The work of our article is part of the resolution of scheduling problems for AIMS using a meta-heuristic (cuckoo search). A significant amount of research in deterministic scheduling has been

dedicated to finding efficient algorithms for scheduling problems during a polynomial time. However, many scheduling problems don't have a polynomial time algorithm; these problems are called NP-hard problems [4].

Most scheduling problems are classified NP-hard in complexity theory [5]. In addition, the majority of industrial problems are so complex that the number of solutions exponentially increases with the size of the problem. Nevertheless, the class of NP-hard problems is one that attracts researcher's attention in the optimization field to propose new approaches. But until this moment no algorithm is effective facing such problems.

Flow shop scheduling problems were proved to be NP-hard when the number of machines m ≥3 [6]. Hence, exact methods cannot be utilised to find an optimal solution for the problems. Researchers have developed many heuristics and meta-heuristics for the flow shop scheduling problems to find near optimal solutions [7]. In the case where it is desired to solve the problem in an exact manner, techniques such as linear programming, dynamic programming or the branch and bound method are often used as shown in Figure 1. It should be remembered that the execution time and the memory space of these methods increase exponentially depending on the size of the problems to be treated.

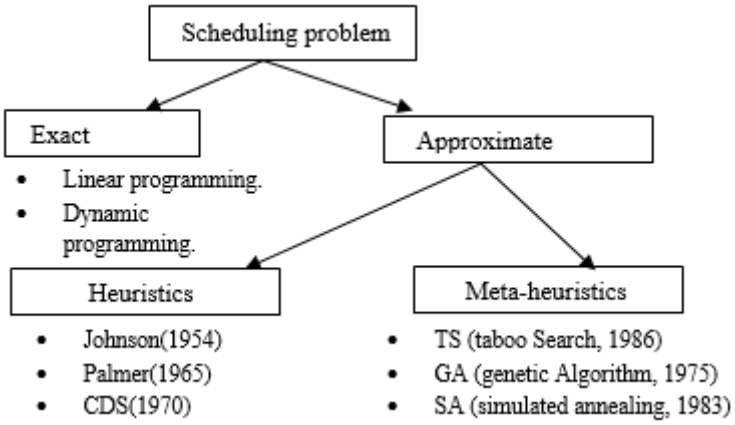

Figure 1. A general classification of methods for solving optimization problems

The need to find quickly a solution of good quality favors the appearance of approximate or stochastic algorithms such as heuristics, see for example: Johnson [8], Palmer [9], Campbell, Dudek, and Smith [10]. Meta-heuristics are general heuristic procedures that can be used for many problems, in our case, to the PFSP. These methods normally start from a random sequence or a sequence constructed by heuristics and iterate until a stopping criterion is met. There is large part of research work done for the PFSP and meta-heuristics, we will show out a few noteworthy papers mainly dealing with simulated annealing (SA) [11], taboo search (TS) [12], genetic algorithms (GA) [13].

In this article, we are interested in a meta-heuristic that is inspired by the natural behavior of a bird species called Cuckoo. This choice was motivated by several reasons; firstly, it's a new algorithm (meta-heuristic) developed very recently by Yang and Deb [14]. Secondly, there is not a very detailed study on the parameters of this meta-heuristic. Three, it has been successfully applied in several types of problems. So we propose an adaptation of CS algorithm for flow shop scheduling problem and a sensitivity analysis according to the parameters of the algorithm to minimize the total completion time Cmax.

After section 1 which was devoted to an introduction to automated industrial manufacturing systems, classification of our problem and methods of resolution. The rest of this paper is organized as follows: Section 2 presents a brief literature review that brings together the most important works of cuckoo research that was done in different fields. A description of the algorithm to study and its adaptation is summarized in section 3. The last section is devoted to the results found and their interpretations.

2. LITTERATURE REVIEW

As we mentioned previously, Cuckoo search is a recently developed meta-heuristic algorithm based on the parasitic breeding behavior of certain species of cuckoo and the presence of Lévy flights in such reproduction strategy [15]. Nowadays cuckoo search has been used in almost every area of: scheduling [16], function optimization [17], engineering optimization [17], and planning [17]. The recent applications of CS for optimization problems have shown its promising effectiveness [18]. For example, the work of Wang *et al.* [16]

where a new cuckoo search algorithm with local search (NCS) is proposed for solving the permutation flow shop scheduling problem.Hence, for population initialization; they used the NEH heuristic (Nwaz *et al.*) to generate high quality initial solutions. A novel variant of the CS algorithm referred as the Inter-Species Cuckoo Search. We can also mention the work done by Tsipianitis *et al.* [19] where this work was done for testing the dynamic tuning of certain important parameters of the CS algorithm, namely the probability of fraction (pa) (an alien egg to be found by the host bird) and the step size of Lèvy flights, in combination with the implementation of three functions: one static and two dynamic approaches. Finally, a hybrid optimization algorithm is developed by combining the most efficient features of CS with those of another swarm-based optimizer, namely bird swarm algorithm (BSA) to accelerate the convergence towards global optimum.

ISCS [15] is developed to minimize the makespan and mean flow time in both hybrid flow-shop scheduling (HFS) and permutation flow-shop sequencing problems (PFSP). Furthermore, a hybrid meta-heuristic based on cuckoo search algorithm and differential evolution for numerical optimization is proposed by Xi *et al.* [20], the algorithm has been tested and compared with other meta-heuristics, Computational experiments depth on a wide range of sets of problems show that the proposed algorithm outperforms many other meta-heuristics. Zhang *et al.* [21], made a study referenced state of the art swarm intelligence based intelligent algorithms, especially ant colonies and the cuckoo search algorithm, with the modification of each algorithm and hybridization strategy of the mentioned algorithms. They presented the modified ACO (ant colony optimization), modified CS, and HACCS (hybrid ant colony and cuckoo search) algorithms to solve the heating route problem. The proposed algorithms were applied to the ZF (zhuozhou-fangshan) heating engineering project. The results show that modified ACO can find the route (solution) with the smallest objective function value, while the modified CS can find the route overlapped to the manual selected route better. Furthermore, the modified CS ran more quickly but stuck into the premature convergence. Following the convergence study mentioned above, the best route chosen by the hybrid algorithm HACCS was the same as the modified ACO algorithm, but with greater efficiency and better stability.

3. RESEARCH METHOD

In the past several years, different kinds of nature-inspired optimization algorithms have been created, and they become very popular. We can montion, particles swarm optimization PSO [22] was inspired by fish and bird swarm intelligence, ant colony optimization (ACO) [21] and The API algorithm were inspired from the foraging behaviour of a population of ants [23]. Cuckoo search is a recent meta-heuristic. It has enriched the number of population-based meta-heuristics solutions, which is inspired by the paradigm of birds grouping.

In Yang and Deb invented a new algorithm meta-heuristic inspired by nature. Specifically, the algorithm was inspired by the obligate brood parasitic behavior of some cuckoo species from which comes the name: cuckoo search (CS) in combination with Levy's flight behavior of some birds and fruit flies in nature [14]. A second version was created by Yang and Deb in 2010 named cuckoo optimization algorithm (COA) [17]. In addition to the CS, the COA algorithm, is an algorithm with similar inspiration in nature, has recently attracted much attentions in solving optimization problems.

The pioneers of the CS algorithm were inspired by the parasitic reproduction behavior of some species of cuckoo that lay their eggs within the nests of other species by entrusting the responsibility of incubating, feeding and rearing their chicks to the host birds. These can detect cuckoo's eggs in their nests; during this case the host bird will either eject the cuckoo's eggs out of its nest or abandon its own nest and build another one in another location.

The CS proposed by Yang and Deb in 2009 is based on the following three basic rules:

- Each cuckoo lays one egg at a time. He drops it in a nest that he chooses randomly.
- A nest of good quality will be carried over to the next generation.
- The number of host nests is fixed, and an egg laid by a cuckoo can be discovered by the host bird according to a probability $P_a \in [0, 1]$. In this case, the host bird can either throw the egg away or abandon the nest, and build a completely new nest in a new location randomly chosen.

Based on these three rules, the basic steps of the cuckoo search (CS) can be summarized as the pseudo code shown in Figure 2 [14].

A cuckoo i generates a new solution $xi(t+1)$ via Lévy flights, according to (1);

$$xi(t+1) = xi(t) + \alpha \oplus Lévy(s, \lambda) \quad (1)$$

where α >0 is the maximal length of the step which should be bound on the scale of the space of search for the problem to be handled. In most cases, $\alpha = 1$ is used [14].

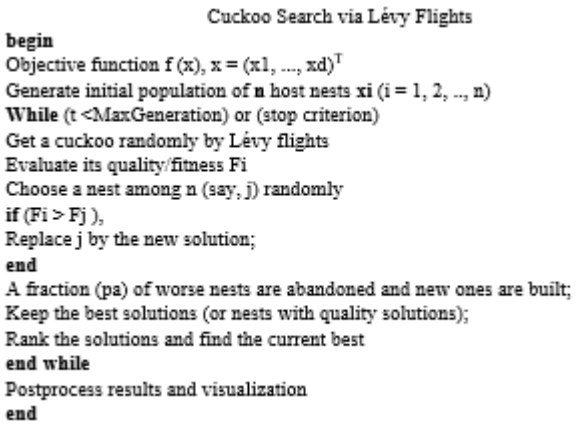

Figure 2. The pseudo code of the cuckoo search

Equation (1) is stochastic (a random walk). Generaly, a random walk is a Markovian chain wiche the next step depends on the current step, whose is the first part of the equation, followed by the transition probability which is the second part. The product represents entrywise multiplications. This entrywise product is similar to those used in PSO, but here the random walk via the Lévy flight is more effective in the exploration of the search space because its step size is much longer in long-term [18]. The Lévy flight represents essentially a random walk whereas the random step size (α) is defined from a Lévy distribution.

$$Lévy \sim u = t,^{-\lambda} \ (1 < \lambda \leq 3) \quad (2)$$

It should be noted that the Lévy has an infinite variance with an infinite mean [14].

The main characteristic of the algorithm CS is its simplicity. In fact, by comparing with other population or agent-based meta- heuristics algorithms, there is a similarity with PSO and GA, but CS it uses some sort of elitism and/or selection similar to that emplyed in harmony search. Furthermore, the randomization is more efficient as the step length is heavy-tailed, and any large step is possible.

3.1. Adaptation algorithm

CS is a population-based algorithm there with few parameters to be defined, and thus it is potentially more generic to adapt to a wider class of optimization problems. The original version of the CS algorithm was proposed to solve continuous problems of optimization [24]. However, the problems of optimization are not all continuous type; they can also be discrete [21] or binary type [18]. Generally, the modification can be categorized into two classes. First class is the adjustment of the parameters, and the second is hybridization with other intelligence algorithms [25]. The work in this section will focus on the adaptation of the CS to flow shop scheduling problem, it is the most well-known combinatorial optimization problem in the real world. The CS algorithm is adapted and applied on the FSSP with its own procedure without using other improvements to show its performance to this type of problem and to have results to be studied. The result of adaptation is a new basic version of CS named "basic discrete cuckoo search (BDCS)". The procedure of adaptation requires a definition of the following elements:

3.1.1. Nest

In the case of our combinatorial optimization problem (PFSP), a nest is considered as an individual of the population with its own solution. Moreover, a nest has the following properties:

- The number of nests is equal to the size of the population.
- The total number of nests is fixed from the begining.
- An abandoned or destroyed nest involves replacing an individual of the population with a new one.
- An egg in a nest represents a solution, in our case represents a sequence of jobs.

3.1.2. Egg

As we have said before, a cuckoo can lay a single egg in a single nest, which gives the eggs the following characteristics:

- An “egg in a nest” is an adopted solution by the individual of the population.
- An egg detected and rejected by a cuckoo implies a bad solution.
- A cuckoo egg is a new candidate solution for a place in the population.

3.1.3. Objective function

The objective function (of test, fitness) is a function which, with each solution in the space of search associates a numerical value to represent its quality or fitness. So the quality or fitness of a solution is proportional to the value of the objective function. In our problem, a nest (sequence) of better quality gives us an optimal Cmax or a solution close to optimal Cmax.

3.1.4. Search space

The search space in a very combinatorial optimization cases will be seen as a "graph" where vertices represent solutions and edges connect neighboring pairs of solutions [26].

a. Movement

In the combinatorial problem, the coordinates of a solution in space research are modified through the properties of the problem being addressed. Generally, the change of position within the combinatorial space is done by a change within the order of the elements of the solution, by a combination, a permutation, or a set of operators named perturbation or movement.

b. Neighborhood

The concept of neighborhood requires that the new solution of a given solution be generated by the smallest possible perturbance. This disruption should make the minimum changes to the present solution.

c. Lévy flights

Lévy Flights has as objective, an intensive research around a solution, followed by long-term steps. According to Yang and Deb, looking for a new better solution is more efficient via Levy flights. So, to enhance the quality of the search we'll associate the length of the step with the value generated by Lévy flights.

d. Step

The step is that the distance between two solutions. It's based on the topology of space and therefore the concept of neighborhood. During this work we have classified the steps consistent with their length, the character and also the number of successive perturbations.

In BDCS, Levy flights have control over all displacements in space of solutions to local and global scale. However, we will show how to present a solution in space and how to move from the present solution to another using some operators, such as swap, insert, and inverse [16]. In this paper, the swapping strategy search is used to generate a neighbor of the current solution. The first operator is 'swap'; two jobs at different positions in the current solution are selected and switched. By performing the operation, a new solution is getting. Figure 3 describes the operation of the exchange operator.

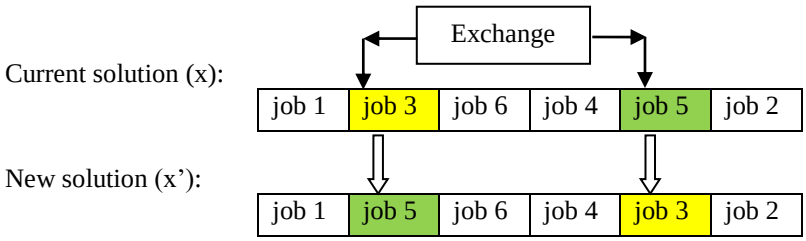

Figure 3. The swap operator

The second operator is 'insert'; the chosen job is moved from its current position in solution (x) and inserted to another position. After this operation, we can get a new solution (x'). Figure 4 shows the insert operator. The last operator is 'inverse'; the tasks between two different positions chosen in solution (x) are reversed. After this process, we can obtain a new solution (x'). Figure 5 illustrates the inverse operator.

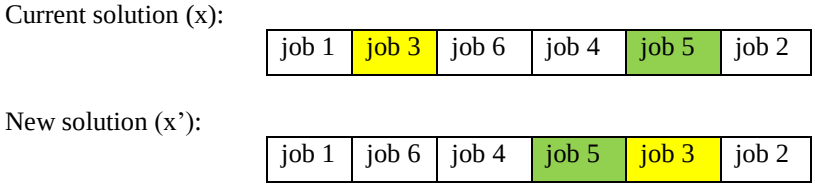

Figure 4. The insert operator

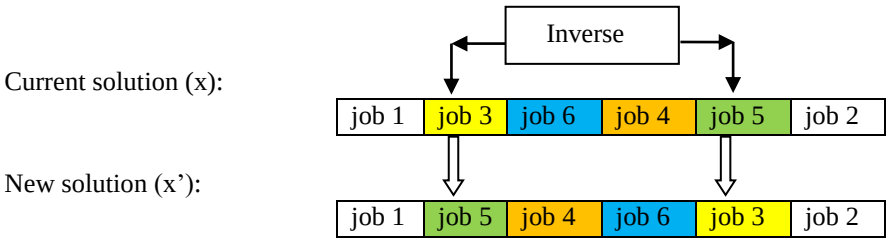

Figure 5. The inverse operator

4. RESULTS AND ANALYSIS

The main objective of scheduling is to find a job order or sequence of jobs that can be realized in a reasonable amount of time that optimizes the proposed objective function. Besides, this type of problem is usually frequented in a variety of manufacturing or service industries, it is one of the most well known scheduling problems is the permutation flow-shop scheduling problem. When solving the PFSP, BDCS adopt its own procedure without using other improvements in order to study the parameters that influence on its speed of convergence to a good solution.

To certify the performance of the proposed BDCS for the flow shop scheduling problem, computational simulations are carried out with some well-studied problems taken from the OR-Library. In this paper, six problems from three classes of PFSP test problems designed by Taillard are selected for our studies [27]. The first two problems are small instances (Taillard 1-Taillard 2) inserted in Tables 1-2. The second problems are medium instances (Taillard 3-Taillard 4) employed in Tables 3-4 and the last two problems selacted are a large instances (Taillard 5-Taillard 6) found in Tables 5-6. These problems have been widely utilised as benchmarks for testing the performance of algorithms by many researchers such as Wang H. *et al.* for FSSP [16], Van Hoorn *et el.* for job shop [28], and Benziani *et al.* for open shop [29].

The tables summarize the experimental results, the first column shows the problem size and we give in the column 'lower/upper bound' the best lower bound (obtained by branch and link methods) and the upper bound (the solutions most known in the literature) for each instance. The 'nest/itt' column shows the number of the nest and the number of iterations. The 'pa' column indicates the percentage of destruction of bad nests, and pa varies from 0.3 to 0.9 with a step of 0.2.

We collected the results then included them in six tables which we divided on the basis of the problem size. The problem with five (5) machines is a small problem in Tables 1-2, the problem with ten (10) machines is a medium problem in Tables 3-4, and the problem with twenty (20) machines is a large problem in Tables 5-6. Anyway, we will discuss the results for each problem (small, medium, and large problem). The experiments vary according to three (3) parameters which are: the probability of fraction or destruction (Pa), the number of nests (Nest) and the number of iterations (itt). The algorithm was encoded in MATLAB 12.0 and run on an Intel Core i7-2700k CPU 3.50 GHz with 8.0 GB Memory in the Windows 7 professional 64. The value of Cmax and the simulation time 'Time' given in the table are the average of ten simulations for each instance. In each table we put the variables so that we can read the table from left to right or from top to bottom and in the end we presented the final results as follows.

Initially,we start with small problem in Tables 1-2, we chose to read the table according to the variable Pa by fixing the number of nests and the number of iterations, we can notice a convergence towards the uppersolution each time we increase the percentage of Pa. For example in Table 2 the instance 5m/100j; nest=100; itt=100; Pa=0.3 we have Cmax= 5624, and for Pa=0.9 we have Cmax= 5590. But for the same instance and the same parameters, we have an increase in the simulation time; for Pa=0.3 we needed 51.22 seconds; and for Pa=0.9 the Time increase to 66.93 seconds. By fixing the numbe rof nests and by varying the number of iterations and the Pa, we can see that the Cmax improves each time when the value of destroying the bad nests (Pa) increases as shown in Tables 1 and 2, which gives us a good population for the next generation.

If we want to study the changes of Cmax according to the number of iterations, the number of nests is determined at 100 nests and the percentage of change of the bad nests is fixed at 0.5, and the problem size is 10 m/20j as shown in Table 3. We can clearly notice that the Cmax for 100/400 Cmax=1726 is better than Cmax for 100/100 Cmax=1759. On the other hand, we note that the simulation time increases five times, for 100/400 Time=281.08 s, than the time needed to find a solution for 100/100 is 52.73 s. Moreover, keeping the same parameters for the Medium problem 2 at Table 4 and increasing the percentage of Pa to 0.9 we have an improvement in Cmax, for 100/400 Cmax=11547 is better than Cmax for 100/100 Cmax= 11632.

Table 1. Small problem 1

Problem size	Lower/upper bound	Pa	0.3		0.5		0.7		0.9	
		nest/itt	C_{max}	Time	C_{max}	Time	C_{max}	Time	C_{max}	Time
5m/20j	1082/1108	100/100	1199	102.16	1199	94.6	1194	96.71	1193	100.3
		100/200	1199	101.94	1200	94.26	1198	96.76	1191	99.8
		100/400	1196	208.86	1188	209.45	1185	200.78	1183	200.92
		200/100	1205	186.02	1202	85.48	1202	89.58	1198	93.56
		200/200	1198	209.28	1199	200.07	1193	209.82	1189	211.67
		200/400	1190	344.22	1188	421.74	1185	391.09	1190	386.54
		400/100	1203	199.2	1196	198.51	1191	201.31	1189	201.11
		400/200	1195	387.66	1191	385.71	1186	388.78	1181	408.8
		400/400	1187	825.22	1185	828.95	1183	837.17	**1180**	857.93

Table 2. Small problem 2

Problem size	Lower/upper bound	Pa	0.3		0.5		0.7		0.9	
		nest/itt	Cmax	Time	Cmax	Time	Cmax	Time	Cmax	Time
5m/100j	5272/5328	100/100	5624	51.22	5600	57.09	5592	63.26	5590	66.93
		100/200	5584	94.81	5578	110.79	5571	124.43	5567	144.36
		100/400	5563	447.31	5557	267.34	5556	283.52	5552	290.78
		200/100	5606	124.82	5589	135.16	5583	409.60	5573	151.73
		200/200	5581	238.54	5563	266.17	5565	284.97	5555	302.90
		200/400	5556	496.78	5555	539.91	5550	598.65	5533	612.99
		400/100	5607	273.33	5570	456.81	5568	347.39	5555	314.78
		400/200	5580	465.21	5563	548.81	5552	599.95	5550	708.80
		400/400	5548	1139.8	5540	1938.9	5537	1149.1	**552**1	1221.7

Table 3. Medium problem 1

Problem size	Lower/upper bound	Pa	0.3		0.5		0.7		0.9	
		nest/itt	C_{max}	Time	C_{max}	Time	C_{max}	Time	C_{max}	Time
10m/20j	1356/1591	100/100	1760	55.73	1759	52.73	1753	59.63	1723	34.94
		100/200	1758	111.91	1745	120.16	1740	120.39	1739	118.28
		100/400	1728	267.13	1726	281.08	1724	286.01	1724	280.51
		200/100	1752	104.85	1751	106.68	1745	129.22	1743	141.07
		200/200	1745	224.22	1739	223.72	1735	221.97	1732	222.31
		200/400	1739	413.00	1736	432.67	1731	429.11	1723	438.55
		400/100	1748	192.18	1742	198.61	1737	197.99	1734	193.81
		400/200	1741	396.41	1735	402.77	1732	393.67	1722	397.27
		400/400	1730	1002.55	1724	901.96	1721	987.37	**1717**	1117.71

Table 4. Medium problem 2

Problem size	Lower/upper bound	Pa	0.3		0.5		0.7		0.9	
		nest/itt	Cmax	Time	Cmax	Time	Cmax	Time	Cmax	Time
10m/200j	10616/10676	100/100	11652	126.84	11647	127.05	11637	147.88	11632	165.08
		100/200	11589	288.59	1185	325.45	11580	375.01	11577	424.42
		100/400	11573	694.99	1155	591.00	11554	723.58	11547	743.61
		200/100	11640	308.68	11638	313.77	11615	383.42	11589	416.37
		200/200	11601	490.31	11584	533.98	11590	602.47	11574	622.58
		200/400	11583	932.72	11562	1015.76	11549	1091.37	11535	1217.48
		400/100	11615	500.89	11590	552.91	11588	598.93	11582	611.85
		400/200	11581	894.65	11566	1035.02	11563	1071.58	11552	1209.98
		400/400	11544	1834.50	11553	2067.02	11529	2114.70	**11528**	2421.35

When we go to big problems, the problem size influences on the simulation time, If we take for example: 200/200; Pa=0.9 for the problem 20 m/20j as shown in Table 5, the Time=314.61 s and for 20 m/500j as shown in Table 6 the Time jumps to 1 308.61 s. It can be clearly seen that there is an exponential increase in the simulation time. Furthermore, there is another very important parameter in the study which influence on the simulation time and the improvement of Cmax, it is the number of the nest. To study the influences of this parameter (nest) on the obtained results, we chose the problem 20m/500j as shown in Table 6 with the parameters Pa=0.9, the number of nests is varied between 100, 200, 400, and the number of iterations is set to 400. We note that the Cmax=29119 for 200/400 with a Time=2625.14 s is better than Cmax=29159 for 100/400 with a Time=1,354.97 s, and the Cmax=29111 for 400/400 is better than the first two with an explosion in simulation time, Time=4477.28 s.

Table 5. Large problem 1

Problem size	Lower/upper bound	Pa	0.3		0.5		0.7		0.9	
		nest/itt	Cmax	Time	Cmax	Time	Cmax	Time	Cmax	Time
		100/100	2424	18.71	2420	34.10	2415	36.22	2414	58.64
		100/200	2405	143.04	2405	144.54	2400	143.99	2398	138.07
		100/400	2392	259.71	2380	240.07	2380	291.49	2379	296.45
		200/100	2418	134.46	2411	144.40	2401	150.11	2401	138.11
20m/20j	1900/2178	200/200	2400	283.28	2396	276.98	2386	251.88	2382	314.61
		200/400	2391	571.30	2375	550.15	2370	705.96	2367	636.08
		400/100	2397	257.50	2394	250.81	2390	769.73	2376	362.73
		400/200	2396	563.07	2382	560.20	2380	663.61	2375	720.24
		400/400	2376	812.48	2370	839.69	2363	965.35	2356	1173.13

Table 6. Large problem 2

Problem size	Lower/upper bound	Pa	0.3		0.5		0.7		0.9	
		nest/itt	Cmax	Time	Cmax	Time	Cmax	Time	Cmax	Time
		100/100	29341	278.81	29321	311.00	29298	338.67	29259	354.30
		100/200	29309	492.90	29278	559.37	29220	559.75	29208	643.78
		100/400	29213	952.81	29202	1082.46	29169	1150.32	29159	1354.97
		200/100	29337	491.72	29333	560.84	29272	629.82	29223	708.53
20m/500j	25922/26189	200/200	29271	865.20	29266	999.54	29256	1094.72	29160	1308.61
		200/400	29197	1592.44	29197	2010.00	29152	2322.36	29119	2625.14
		400/100	29279	830.72	29269	975.65	29240	1031.80	29215	1250.50
		400/200	29242	1720.54	29180	2098.38	29186	2259.06	29170	2517.44
		400/400	29175	3598.88	29169	3674.28	29138	4406.18	29111	4477.28

5. CONCLUSION

In this study, we were interested in adapting a meta-heuristic to solve the flow shop type scheduling problem in an automated industrial system in deferred time, without taking into account any constraints. The adapted meta-heuristic is a meta-heuristic with a population of solutions. It is inspired by the natural behavior of a bird species and formulated by the cuckoo search algorithm: the CS algorithm. Only ten (10) years ago the cuckoo algorithm was created, but it can prove its effectiveness on many difficult optimization problems, such as flow shop scheduling problems with a number of machines greater than or equal to three machines.

We tested several parameters that make up the algorithm, we started with the fraction parameter (pa), we notice that, by increasing the percentage of destruction, the objective function (In our case, it is the Cmax) will be improved. After that, we took the number of iterations to study, its influence on the convergence of the algorithm; we found that, if we increase the number of iterations there is an improvement of the objective function; on the other hand, we had an explosion in the simulation time.

The last parameter is the number of the nest, which has an undeniable effect on the quality of the final solution and on the speed of convergence of the algorithm. Besides that, we have treated large problems. These latter have a direct correlation relation with the simulation time, that is to say, if the problem size has become large, the simulation time increases and vice versa. To conclude, there is a direct correlation between the variables pa, itt, nest and the time needed to find the solution. In addition, there is an inverse relationship between the previously mentioned variables and the objective function.

To validate our results, we used one of the most known references in the literature (Taillard instances), the results are close to the best solutions found (upper bound) but we still have improvements to make. Finally, it can be said that BDCS is suitable to be adapted to solve the FSSP and provide good results. This can be explained mainly by a good balance between exploration of space to different research areas and exploitation of promising spaces, that because lévy flight was employed.

There are several improvements that we can focus on in the future; For example, improving the initial population by integration of a heuristic. It is possible to exploit new regions, that by increasing the number of nests and the number of iterations to see the changes to the objective function and the simulation time. We can also hybridize with meta-heuristics known in the literature such as taboo research, ant colonies or simulated annealing, to improve the objective function.

This work can be considered as a reference for researchers who want to use the cuckoo algorithm in their works; they can take the appropriate parameters to their needs. For example, if they want a good solution, they must take a high value of pa and a large number of nests, on the other hand if they want to optimize the simulation time, a small value of iterations and the number of nests, it must be taken.

REFERENCES

[1] Jose M. Framinan, Rainer Leisten, Rubén Ruiz García, "Manufacturing Scheduling Systems: An Integrated View on Models, Methods and Tools," *Springer London Heidelberg New York Dordrecht, first edition*, 2014.

[2] Khedim, A., et al., "Ordonnancement temps réel d'un FMS par l'approche de l'algorithme de colonies d'abeilles," *Colloque International Sur Le Monitoring Des Systèmes Industriels*, CIMSI, 2014.

[3] K. E. Stecke, "Design, planning, scheduling and control problems in flexible manufacturing systems," *Annals of Operations Research*, vol. 3, no. 1, pp. 3-12, 1985.

[4] Pinedo, M., "Scheduling: Theory, Algorithms and Systems," *Prentice Hall, New Jersey, second edition* 2002.

[5] Graham R. L., Lawler E. L., Lenstra J. K. and Rinnooy Kan A. H. G., "Optimisation and approximation in deterministic sequencing and scheduling: a survey," *Annals of Discrete Mathematics*, vol. 5, pp. 287-326, 1979.

[6] Garey, M. R. D., Johnson, D. S., Sethi, R., "The complexity of flow shop and job shop scheduling," *Mathematics of Operations Research*, vol. 1, no. 2, pp. 117-129, 1976.

[7] Alexander J. Benavides, Marcus Ritt, "Two simple and effective heuristics for minimizing the makespan in non-permutation flow shops," *Computers & Operations Research*, vol. 66, pp. 160-169, 2016.

[8] Johnson, S. M., "Optimal Two and Three Stage Production Schedules with Set-Up Times, Included," *Naval Research Logistics Quarterly*, vol. 1, no. 1, pp. 61-68, 1954.

[9] Palmer, D. S., "Sequencing jobs through a multi-stage process in the minimum total time-a quick method of obtaining a near optimum," *Journal of the Operational Research Society*, vol. 16, pp. 101-107, 1965.

[10] Campbell, H. G., Dudek, R. A., Smith, M. L., "A Heuristic Algorithm for the n Job, m Machine Sequencing Problem," *Management Science*, vol. 16, no. 10, pp. 630-637, 1970.

[11] S. Kirkpatrick, C. D. Gelatt, M. P. Vecchi, "Science Optimization by Simulated Annealing," *Science*, vol. 220, no. 4598, pp. 671-680, 1983.

[12] F. Glover, "Tabu Search-Part I," *ORSA Journal on Computing*, vol. 1, no. 3, 1989.

[13] Holland, J., "Adaptation in natural and artificial systems," *The University of Michigan Press, Ann Arbor*.1975.

[14] X.S. Yang, and S. Deb, "Cuckoo search via Lévy flights," *2009 World Congress on Nature & Biologically Inspired Computing (NaBIC)*, Coimbatore, 2009, pp. 210-214.

[15] Preetam Dasgupta, and Swagatam Das, "A Discrete Inter-Species Cuckoo Search for Flowshop Scheduling Problems," *Computers & Operations Research*, vol. 60, pp. 111-120, 2015.

[16] Wang H., Wang W. J., Sun H., Cui Z. H., Rahnamayan S., Zeng S. Y., "A new cuckoo search algorithm with hybrid strategies for flow shop scheduling problems," *Soft Computing*, vol. 21, pp. 4297-4307, 2016.

[17] I. Fister Jr., X. S. Yang, D. Fister, I. Fister, "Cuckoo search: A brief literature review, in: Cuckoo Search and Firefly Algorithm: Theory and Applications," *Studies in Computational Intelligence*, vol. 516, pp. 49-62, 2014

[18] Gherboudj, A., Layeb, A. and Chikhi, S., "Solving 0-1 knapsack problems by a discretebinary version of cuckoo search algorithm," *International Journal of Bio-Inspired Computation*, vol. 4, no. 4, pp. 229-236, 2012.

[19] Tsipianitis, A., and Tsompanakis, Y., "Improved Cuckoo Search algorithmic variants for constrained nonlinear optimization," in *Advances in Engineering Software*, vol. 149, p. 102865, 2020.

[20] Jun Xi, and Liming Zheng, "A hybrid algorithm based on cuckoo search and differential evolution for numerical optimization," *Journal Computing, Performance and Communication Systems*, vol. 4, no. 1, pp. 1-8, 2020.

[21] Zhang, Y., Zhao, H., Cao, Y., Liu, Q., Shen, Z., Wang, J., Hu, M., "A Hybrid Ant Colony and Cuckoo Search Algorithm for Route Optimization of Heating Engineering," *Energies*, vol. 11, no. 10, p. 2675, 2018.

[22] Kennedy, J. and Eberhart, R. C., "A discrete binary version of the particle swarm algorithm," *1997 IEEE International Conference on Systems, Man, and Cybernetics. Computational Cybernetics and Simulation*, Orlando, FL, USA, vol. 5, 1997, pp. 4104-4108.

[23] Boumediene F. Z., Houbad Y., Bessenouci H. N., Hassam A., Ghomri L., "A Makespan Minimization API Algorithm for Flow Shop Scheduling," in *Electrotehnica, Electronica, Automatica (EEA)*, vol. 67, no. 2, pp. 123-129, 2019.

[24] Gandomi, A. H., Yang, X.-S., & Alavi, A. H., "Cuckoo search algorithm: a metaheuristic approach to solve structural optimization problems," *Engineering with Computers*, vol. 29, no. 1, pp. 17-35, 2013.

[25] Chiroma, H., et al., "Bio-inspired computation: Recent development on the modifications of the cuckoo search algorithm," *Applied Soft Computing*, vol. 61, pp. 149-173, 2017.

[26] Ouaarab, Aziz, "Résolution de Problèmes d'Optimisation Combinatoire par des Métaheuristiques Inspirées de la Nature," *Recherche du Coucou via les Vols de Lévy*, 2015.

[27] E. Taillard, "Benchmarks for basic scheduling problems," *European Journal of Operational Research*, vol. 64, no. 2, pp. 278-285, 1993.

[28] Van Hoorn, Jelke J., "The Current state of bounds on benchmark instances of the job-shop scheduling problem," *Journal of Scheduling*, vol. 21, no .1, pp. 127-128, 2018.

[29] Benziani, Yacine, Kacem, Imed, E. T. Laroche, Pierre, "Genetic Algorithm for Open Shop Scheduling Problem," *2018 5th International Conference on Control, Decision and Information Technologies (CoDIT)*, Thessaloniki, 2018, pp. 935-939.