Reports in Mechanical Engineering
Vol. 4, No. 1, 2023, pp. 90-103
ISSN: 2683-5894, DOI: https://doi.org/10.31181/rme040116072023m

Optimizing production scheduling with the spotted hyena algorithm: A novel approach to the flow shop problem

Toufik Mzili [1*], Ilyass Mzili[2], Mohammed Essaid Riffi[1], Dragan Pamucar[3], Mohamed Kurdi[4], Ali Hasan Ali[5,6]

[1] Department of Computer Science, Laboratory LAROSERI, Faculty of Science, Chouaib Doukkali University, EI Jadida, Morocco
[2] Laboratory of research in management and Development, Department of Management, Faculty of Economics Hasan 1st University, Settat, Morocco
[3]Department of Operations Research and Statistics, Faculty of Organizational Sciences, University of Belgrade, Belgrade, Serbia
[4] Faculty of Informatics Engineering, Idlib University, Idlib, Syria
[5]College of Engineering Technology, National University of Science and Technology, Dhi Qar 64001, Iraq
[6]Institute of Mathematics, University of Debrecen, Hungary

Article Info

Article history:

Received January 9, 2023
Revised July 14, 2023
Accepted July 16, 2023

Keywords:

Flow shop scheduling,
Optimization algorithm,
Spotted hyena,
Production scheduling,
Job sequencing,
Resource allocation,
Production flow,
Flow shop problem,
Artificial intelligence,
Swarm intelligence.

ABSTRACT

The spotted hyena optimization algorithm (SHOA) is a novel approach for solving the flow shop-scheduling problem in manufacturing and production settings. The motivation behind SHOA is to simulate the social dynamics and problem-solving behaviors of spotted hyena packs in order to identify and implement optimal schedules for jobs in a flow shop environment. This approach is unique compared to other optimization algorithms such as WOA, GWO, and BA. Through extensive experimentation, SHOA has been shown to outperform traditional algorithms in terms of solution quality and convergence speed. The purpose of this study is to present the details of the SHOA algorithm, demonstrate its effectiveness, and compare its performance with other optimization approaches. The method used in this study includes extensive experimentation and comparison with other algorithms. The findings of this study show that SHOA is a promising tool for optimizing production processes and increasing efficiency. The implications of this study are that SHOA can be used as an effective tool for solving flow shop-scheduling problems in manufacturing and production settings.

Copyright © 2023 Regional Association for Security and crisis management and European centre for operational research. All rights reserved.

Corresponding Author:

Toufik Mzili,
Department of Computer Science, Laboratory LAROSERI, Faculty of Science, Chouaib Doukkali University, EI Jadida, Morocco
Email: mzili.t@ucd.ac.ma

1. Introduction

The flow shop-scheduling problem (Ferraro et al., 2023) is a common challenge in manufacturing and production environments (Ab Wahab et al., 2015), where multiple jobs must be processed through a series of machines or stages in a specific order. Finding the optimal order and timing of these jobs can help to increase efficiency, reduce costs, and improve customer satisfaction. However, finding the optimal solution can be a complex task as it requires considering multiple varia-bles and constraints.

In recent years, researchers have turned to various optimization algorithms (Ab Wahab et al., 2015) in an effort to find efficient and effective solutions to the flow shop-scheduling problem.

There are various algorithms and approaches that can be used to solve the flow shop-scheduling problem, including heuristics (Lenat, 1982), mathematical programming (Ball, 2011), and simulation. These approaches can be used to optimize the scheduling of jobs in realtime (Cai et al., 2023) or to plan and schedule jobs in advance (Musselman et al., 2022). However, these methods have some limitations and gaps that make it difficult to solve this problem effectively. For example, some heuristic algorithms may not be able to find the optimal solution, and mathematical programming methods can be computationally expensive.

In this article, we will focus on optimization algorithms for solving the flow shop scheduling problem, including those based on swarm intelligence (Sharma et al., 2021), such as particle swarm optimization (PSO) (Liao et al., 2012), ant colony optimization (ACO)(Yagmahan & Yenisey, 2008), and bee colony optimization (BCO) (Anuar et al., 2016). These algorithms have been used to solve the flow shop problem, but they have their limitations as well. For example, PSO, ACO and BCO are sensitive to the initial solution, and they may not be able to find the global optimum.

One such algorithm that addresses these limitations is the spotted hyena optimization algorithm (SHOA) (Dhiman & Kumar, 2018), which is based on the behavior and hunting strategies of spotted hyenas. This algorithm simulates the social dynamics and problem-solving behaviors of spotted hyena packs, which is unique compared to other optimization algorithms such as WOA, GWO, and BA. The objective of this research is to demonstrate the effectiveness of SHOA in comparison with other optimization approaches and to investigate the potential of this algorithm in solving the flow shop scheduling problem.

The SHO (Ghafori & Gharehchopogh, 2022) algorithm utilizes a population of hyenas that represent potential solutions to the flow shop scheduling problem. Each hyena is assigned a set of parameters that correspond to different aspects of the scheduling process, such as the order of the jobs and the processing times at each stage. The hyenas are then iteratively updated based on their performance and the performance of other hyenas in the population.

This article will explore the development and application of the SHO algorithm for solving the flow shop scheduling problem. We will examine the performance of the algorithm in various test cases and compare it to other optimization algorithms. Through this analysis, we hope to demonstrate the potential of the SHO algorithm as a tool for optimizing production processes in a variety of industries.

We will provide an in-depth examination of the SHO algorithm and its application to the flow shop-scheduling problem. We will discuss the underlying principles of the algorithm, its performance compared to other optimization techniques, and its potential benefits and limitations. Additionally, we will provide examples of how the SHO algorithm has been applied in real-world settings and its potential future directions. Overall, our goal is to provide a comprehensive overview of the SHO algorithm and its potential as a powerful tool for solving the flow shop scheduling problem.

The use of the spotted hyena optimization algorithm (SHOA) for solving the flow shop scheduling problem in manufacturing and production settings needs justification as there are many other metaheuristic algorithms available for this problem. The motivation for using SHOA is that it simulates the social dynamics and problem-solving behaviors of spotted hyena packs, which is unique compared to other optimization algorithms such as WOA, GWO, and BA. The attributes that make SHOA unique include its ability to adapt to changing environments, its use of a cooperative approach for problem solving, and its ability to learn from previous solutions and experiences.

One of the main advantages of SHOA is its ability to adapt to changing environments. This is important in manufacturing and production settings as the conditions and requirements of the problem can change over time. SHOA uses a cooperative approach for problem-solving which allows for the sharing of information and resources among the members of the pack. This approach leads to better and more efficient solutions than traditional algorithms.

Another advantage of SHOA is its ability to learn from previous solutions and experiences. This allows the algorithm to improve its performance over time and to find better solutions for the problem at hand. Overall, the unique attributes of SHOA make it a promising tool for solving the flow shop scheduling problem in manufacturing and production settings. The objective of this research is to demonstrate the effectiveness of SHOA in comparison with other optimization approaches, and to investigate the potential of this algorithm in solving the flow shop scheduling problem.

The main objectives of this work are:

- To present the spotted hyena optimization algorithm (SHOA) as a novel approach for addressing the flow shop scheduling problem in manufacturing and production settings.
- To evaluate the performance of SHOA in comparison to other well-known optimization algorithms such as PSO, ACO, and BCO.

- To examine the capacity of SHOA to adapt to changing environments and to learn from previous solutions and experiences.

- To establish the effectiveness of SHOA in identifying optimal schedules for jobs in a flow shop environment.

- To investigate the potential of SHOA as a tool for enhancing production processes and increasing efficiency in manufacturing and production settings.

The organization of this article is as follows: Section 2 presents related work, Section 3 introduces the flow shop problem, Section 4 presents the spotted hyena swarm optimization algorithm, Section 4 presents the proposed discrete SHOA algorithm to solve the flow shop problem, Section 6 presents the experimental results, Section 7 provides a comparison and analysis using the Wilcoxon test, and finally the conclusion.

2. Related works

Swarm intelligence algorithms have become increasingly popular in the realm of scheduling and optimization, specifically when dealing with the flow shop scheduling problem. This problem involves a sequence of jobs that need to be processed through a series of machines or stages in a particular order, with the objective of optimizing the schedule to meet certain goals, such as maximizing resource utilization or minimizing completion time.

One example of a swarm intelligence algorithm applied to this problem is the ant colony optimization (ACO) (Yagmahan & Yenisey, 2008b) introduced by Yagmahan and Yenisey in 2008. The ACO algorithm mimics the behavior of real-life ants that use pheromone trails to find the shortest path between their nest and a food source. In the context of flow shop scheduling, the ACO algorithm employs a similar approach, with each "ant" representing a potential schedule and the pheromone trails indicating the quality of each schedule. The algorithm continually improves the quality of schedules by adjusting the pheromone trails based on the schedules' performance.

Another widely used swarm intelligence algorithm in this domain is the particle swarm optimization (PSO) algorithm, which was introduced by Kennedy and Eberhart in 1995 (Kennedy & Eberhart, 1995). PSO algorithms are inspired by the flocking behavior of birds, with each "particle" representing a potential schedule. The algorithm improves the quality of schedules by adjusting the particles' positions based on their performance.

Both ACO and PSO algorithms have been applied to various variations of the flow shop scheduling problem, such as the flow shop scheduling problem with parallel machines (Akhshabi et al., 2014) and the flow shop scheduling problem with sequence-dependent setup times (Shen et al., 2018). These algorithms have been compared to other optimization techniques such as genetic algorithms (Jorapur et al., 2014) and simulated annealing, and have demonstrated efficacy in finding high-quality solutions in a relatively short amount of time (Jorapur et al., 2014). The use of swarm intelligence algorithms in the flow shop scheduling problem holds great promise and has the potential to enhance efficiency and productivity in manufacturing and production settings.

3. Flow shop problem

The flow shop problem is a type of scheduling problem (Fu et al., 2021; Komaki et al., 2019) in which a set of tasks must be performed in a specific order on a set of machines or resources. The objective is to minimize the total completion time or makespan (Dhiman & Kumar, 2018), i.e. the time it takes to complete all tasks while satisfying certain constraints such as machine availability and task dependencies.

The problem can be formulated mathematically using the following notation:

– n: the number of jobs
– m: the number of machines
– p_{ij}: the processing time of job *i* on machine j
– C_{ij}: the completion time of job *i* on machine *j*
– C_i: the completion time of job *i* (i.e., the time it takes to complete job *i* on all machines)
– C_{max} he makespan (i.e., the maximum completion time of all jobs)

The objective function of the flow shop problem can be written as:

– $minimize\ C_{max}$

Object to:

$$C_{ij} = max\{C_{i-1,j}, C_{i,j-1}\} + p_{ij} \quad (1)$$

$$for\ i = 1, \ldots, n\ and\ j = 1, \ldots, m$$

$$C_i = C_{i,m} \quad for\ i = 1, \ldots, n \quad (2)$$

where the first constraint ensures that the completion time of each job on each machine is the maximum of the completion times of the same job on the previous machine and the previous job on the same machine. The second constraint ensures that the completion time of each job is the completion time on the last machine.

The flow shop problem is a combinatorial optimization problem that can be solved using various techniques, such as heuristic algorithms, integer programming, and branch-and-bound algorithms.

4. The spotted hyena optimization (SHO) algorithm

The spotted hyena optimization (SHO) algorithm (Dhiman & Kumar, 2018) is inspired by the behavior and social dynamics of spotted hyenas, which are a type of carnivorous mammal found in Africa. Specifically, the SHO algorithm is inspired by the way that spotted hyenas work together to solve problems and make decisions as a group.

Spotted hyenas are known for their complex social structure and their ability to work together as a group to hunt and defend territory. They use vocalizations, body language, and other forms of communication to coordinate their actions and make decisions.

The SHO algorithm uses these principles of group decision-making and social coordination to solve optimization problems. It works by dividing the optimization problem into smaller subproblems, which are then solved concurrently by a group of "virtual hyenas" using a combination of exploration and exploitation. The solutions from the virtual hyenas are then combined to generate a final solution to the optimization problem.

The SHO algorithm has been applied to a variety of optimization problems, including those in the fields of machine learning, engineering, and economics. It has been shown to have good performance in terms of solution quality and computational efficiency. Spotted hyena optimization (SHO) is an optimization algorithm inspired by the social behavior and decision-making of spotted hyenas. Spotted hyenas are social animals that live and hunt in groups, relying on a network of trusted friends that can include over 100 members. They communicate with each other using specialized calls, postures, and signals and can recognize and rank their kin and other individuals.

4.1. Mathematical Formulation and Optimization Technique

In this subsection, we will introduce the mathematical models that outline the encircling behavior of predators and briefly mention the other behaviors of searching, hunting, and attacking prey. We will also present the SHO (Search, Hunt, and attacking) algorithm, which is used to model these behaviors.

4.1.1. encircling

Spotted hyenas are known for their hunting tactics, including encircling their prey to trap and isolate it. They use vocalizations, body language, and other forms of communication to coordinate their movements and surround their prey, cutting off any escape routes and preparing for the attack. This strategy allows them to take down even large and powerful animals, such as wildebeest and zebra, by overwhelming them with numbers and coordinating their attacks.

To model this behavior mathematically, we use the following equations:

$$Dh = \left| B \cdot P_{prey}(x) - P(x) \right| \quad (3)$$

$$P(x+1) = p_{prey}(x) - E \cdot Dh \quad (4)$$

where *Dh* represents the distance between the prey and the spotted hyena, *x* indicates the current iteration, *B* and *E* are coefficient vectors, $P\_prey\ (x)$ indicates the position vector of the prey, and *P* is the position vector of the spotted hyena. The symbols | | and · represent the absolute value and vector multiplication, respectively.

The vectors *B* and *E* are calculated as follows:

$$B = 2 \cdot rd1 \tag{5}$$

$$E = 2h \cdot rd2 - h \tag{6}$$

$$h = 5 - \left(Iteration * \left(\frac{5}{MaxIteration} \right) \right) \tag{7}$$

where *Iteration* = 1, 2, 3, . . . , *MaxIteration*

To find a balance between exploration and exploitation, the value of h is gradually decreased from 5 to 0 over the maximum number of iterations (*MaxIteration*). This encourages more exploitation as the number of iterations increases. *B* and *E* are random vectors within the range of [0, 1], which also helps to promote exploitation as the iteration value increases. In addition, *rd1* and *rd2* are random vectors within the range of [0, 1].

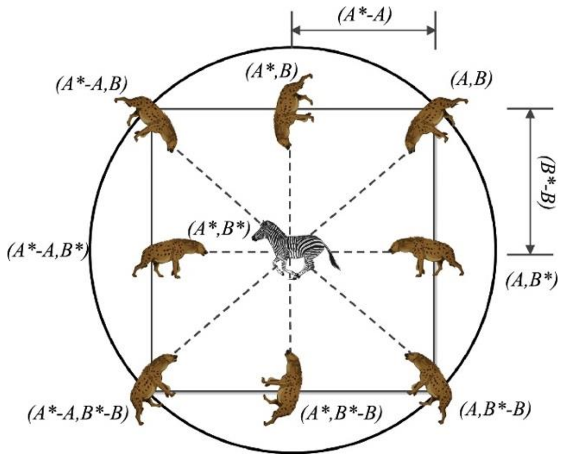


Figure 1. Movement and Position Changes of Spotted Hyenas

Figure 1 illustrates the impacts of equations 5 and 6 in a two-dimensional environment. In this figure, the spotted hyena (*A*, *B*) is able to update its position towards the position of the prey (*A**, *B**). By modifying the values of vectors, a variety of potential locations can be reached from the current position.

4.1.2. Searching

In addition to encircling prey, spotted hyenas also use their keen senses of sight, hearing, and smell to search for potential prey. They are known to track prey over long distances and through challenging terrain, using their powerful legs and stamina to pursue their targets. Once they have located their prey, they will use their encircling tactics to close in for the attack.

4.1.3. Hunting

When hunting, spotted hyenas use a combination of stealth and coordination to take down their prey. They often work together to surround and isolate their target, using vocalizations and body language to communicate and coordinate their movements. Once the prey is trapped, the spotted hyenas will attack by using their sharp teeth and powerful jaws to bring down their prey.

4.1.4. Attacking

Once the prey has been encircled and hunted, the spotted hyenas will attack in a coordinated manner by using their sharp teeth and powerful jaws to bring down their prey. They will work together to bite and tear at the prey, using their superior strength and numbers to overpower their target. The spotted hyenas will continue to attack until the prey is killed or driven off, at which point they will divide the spoils of the hunt among themselves.

5. The proposed discrete DSHO to solve the flow shop problem

In this study, we will focus on using the encircling behavior to solve the flow shop scheduling problem. However, it is worth noting that there are other behaviors that can also be utilized in this context. To utilize the encircling behavior for this problem, the following steps can be followed:

1. Define the problem: Clearly define the problem you are trying to solve, including the number of tasks, the number of workshops, and any constraints or requirements to consider.
2. Initialize the population: Create a population of spotted hyenas that will represent potential solutions to the problem. Each hyena will be assigned a set of tasks to be performed in a random order.
3. Define the prey: Set the prey to represent the best possible position, i.e. the optimal sequence as we do not know the best solution at the beginning. We assume that the prey will be the hyena closest to the optimal solution during the search.
4. Evaluate the distance between the hyena and the prey: Calculate the distance between each hyena and the prey using the objective function of the flow shop problem, such as minimizing the makespan or total completion time.
5. Select the hyena closest to the prey: Select the hyena with the shortest distance to the prey as the fittest hyena. This hyena will be used to generate the next generation of hyenas.
6. Generate new hyenas: Generate new hyenas from the fittest hyena using discrete operations, such as permutation and swapping, instead of vector multiplication and absolute value by eq(3).
7. Apply the 2-opt algorithm: This involves selecting two edges in the schedule and swapping the order of the jobs between them. Calculate the new makespan after the swap and if it is better than the current solution, keep the new solution.
8. Repeat the process until the optimal solution is found: Continue the process of evaluating the distance between the hyena and the prey, selecting the fittest hyena, and generating new hyenas until the optimal solution is found.

To apply this mechanism to solve the flow shop problem, several adjustments must be made to the mathematical representation of the equations and the mathematical operators used. Specificavly:

- Each hyena *P(x)* can represent a sequence of tasks to be performed, but in a random order, i.e. a possible random solution to the flow shop problem.
- The position *P_prey(x)* of the prey represents the best possible position, i.e. the optimal sequence as we do not know the best solution at the beginning. We assume that P_prey(x) will be the hyena closest to the prey during the search.
- The mathematical operators must be modified as they are designed only for linear problems, while the flow shop problem is a discrete problem. The necessary changes include:
- Using discrete operations, such as permutation and swapping, instead of vector multiplication and absolute value.
- Adjusting the coefficients B and E to be suitable for the flow shop problem, such as using task processing times or machine speeds as coefficients.
- Using the objective function of the flow shop problem, such as minimizing the makespan or total completion time, as the criterion for determining the distance between the hyena and the prey.

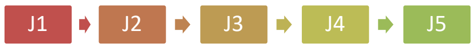
4.1.1. Mathematical operator's changes.

The mathematical operators for the flow shop problem will be redefined as follows:

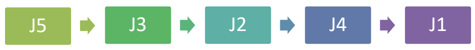
- $Dh = \left| B \cdot P_{prey}(x) - P(x) \right|$: The operator of subtraction between two hyena positions will be changed in our case to a list of swaps to be performed on a sequence of jobs $\mathrm{P(t)}$ to obtain the first sequence list $\mathrm{Pbest(t)}$.
- $E \cdot Dh$: This operation between areal number in the range [0,1] and a list of swaps will be defined to manipulate and reduce the number of swaps generated by the previous equation.
- $p_{prey}(x) - E \cdot Dh$: This operation allows for the final number of possible swaps to be applied to a sequence of jobs.

These changes will be clarified in the following example:

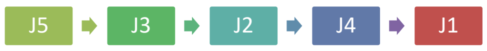
- $\boldsymbol{P_{prey}(x)}$:



- $\boldsymbol{P(x)}$:



- $\boldsymbol{Dh} = \left| \boldsymbol{P_{prey}(x)} - \boldsymbol{P(x)} \right|$ **:** *List of swaps to be performed on a sequence of jobs* $\boldsymbol{P(x)}$ *to obtain the first sequence list* $\boldsymbol{P_{prey}(x)}$ ***=[[J5,J1},{J3,J2}]***
- **E=0,5**
- $\boldsymbol{E \cdot Dh}$**= ½ [J5,J1},{J3,J2}]={J5,J1}**
- $\boldsymbol{p_{prey}(x) - E \cdot Dh}$ **=**



The 2-opt algorithm can be applied to the given schedule by selecting any two edges and swapping the order of the jobs between them. The new makespan is then calculated based on the swapped order. If the new makespan is better than the current solution, then the new solution is kept.

For example, if we have a schedule as follows:
- Machine 1: Job 1-2-3
- Machine 2: Job 4-5-6

We can select the edges (1,2) and (5,6) and swap the jobs between them to get:
- Machine 1: Job 1-5-6
- Machine 2: Job 4-2-3

The new makespan is calculated based on this swapped order and if it is better than the current makespan, the new solution is kept. This process can be repeated by selecting different edges until no further improvement is possible.

Algorithm 1 2-opt for Flow Shop Problem

```
procedure 2-OPT($schedule, tasks, machines$)
    $best_schedule \leftarrow schedule$
    $best_makespan \leftarrow calculate_makespan(schedule, tasks, machines)$
    for $i \in 1 \ldots |schedule|$ do
        for $j \in i+1 \ldots |schedule|$ do
            $new_schedule \leftarrow swap(schedule, i, j)$
            $new_makespan \leftarrow calculate_makespan(new_schedule, tasks, machines)$
            if $new_makespan < best_makespan$ then
                $best_schedule \leftarrow new_schedule$
                $best_makespan \leftarrow new_makespan$
            end if
        end for
    end for
    return $best_schedule$
end procedure
```

The final algorithm can be described as follow:

Algorithm 2 The DSHO algorithm for Flowshop Scheduling

```
1: procedure DSHO($J$, $M$, $P$, $S$)
2:     $n \leftarrow |J|$                                        ▷ Number of jobs
3:     $m \leftarrow |M|$                                        ▷ Number of machines
4:     $X \leftarrow$ Initialize population of Hyena's           ▷ Population of Hyena
5:     $best \leftarrow$ Initialize best solution                ▷ Best solution found so far
6:     for $t \leftarrow 1$ to $T$ do                            ▷ Iterate for a maximum of $T$ iterations
7:         for $i \leftarrow 1$ to $n$ do                        ▷ Iterate over all Hyenas
8:             $x_{new} \leftarrow$ update position of solution for Hyena $i$ eq(6)
9:             $f_{new} \leftarrow$ Evaluate fitness of solution $x_{new}$ ▷ Fitness of solution $x_{new}$
10:            if $f_{new} < f_{best}$ then              ▷ If new solution is better than current best
11:                $best \leftarrow x_{new}$                  ▷ Update best solution
12:            end if
13:            $x_{new} \leftarrow$ 2-opt($best$)              ▷ applied 2-opt improvement
14:            $f_{new} \leftarrow$ Evaluate fitness of solution $x_{new}$ ▷ Fitness of solution $x_{new}$
15:            if $f_{new} < f_{best}$ then              ▷ If new solution is better than current best
16:                $best \leftarrow x_{new}$                  ▷ Update best solution
17:            end if
18:            $X \leftarrow$ Update population of hyenas ▷ Update population of hyenas
19:        end for
20:    end for
21:    return $best$                                       ▷ Return best solution found
22: end procedure
```

6. Experimental results

The performance of the SHO algorithm was evaluated on a set of 45 benchmark instances from the OR library. The results of this evaluation are shown in Table [3-5], which includes the instance name ('Instance'), the number of jobs (n) and machines (m) for each instance ('$n\times m$'), the best result achieved by other algorithms ('BKS'), the best and worst results obtained through the SHO method ('Best' and 'Worst'), the average results ('Average'), and the average execution time in seconds for 20 runs ('Time'). The '$PDav$(%)' column shows the percentage deviation of the average solution length from the optimal solution length, calculated using equation 8.

$$PDav(\%) = \frac{(Average - BKS) \times 100\%}{BKS} \quad (8)$$

The "$PDav$(%)" column highlights values of 0.00 in bold when all solutions found in the 20 runs are equal to the length of the best-known solution. If the average of the solutions found in all tests is less than the length of the best-known solution, these values are highlighted in bold and blue. This indicates that the SHO algorithm was able to find solutions that either equal or exceed the best-known solutions for these instances.

To gauge the effectiveness of the SHO algorithm, it is important to compare it with other problem-solving methods. For a more comprehensive understanding of SHO's capabilities, it is recommended to compare it to a variety of different metaheuristics (Table 2). This will provide a clearer picture of SHO's strengths and weaknesses compared to other algorithms and help determine in which situations it performs best. By comparing SHO to a range of metaheuristics, we can gain a better understanding of how it performs against different types of problems and techniques. The initial parameters of SHO are described in Table 1.

Table 1. Parameters of Discrete SHOA

PARAMETER	VALUE
The population of rat size: N	**100**
B	A random value between [0,1]
E	A random value between [0, 1]
Rd1	A random value between [0,1]
Rd2	A random value between [0, 1]
Nb iteration	**900**

Table 2. List of metaheuristics that can be compared.

Algorithms	Presentation and Inspiration
SSO (Kurdi, 2021)	The Social Spider Optimization (SSO) algorithm is based on the collaborative behavior of social spider colonies, which involve interactions between males and females performing various tasks such as web-building, prey capture, and mating. The algorithm incorporates both direct and indirect interactions among the "spiders" in the optimization process.
SCE-OBL (Zhao, et al., 2015, 2021)	The Shuffled Complex Evolution (SCE) algorithm with opposition-based learning is an optimization method that combines the SCE algorithm, a global optimization technique with opposition-based learning, a method for generating new solutions by manipulating existing ones.
CLS-BFE (Zhao, et al., 2016)	The Chaotic Local Search Based Bacterial Foraging Algorithm (CLSBBFA) is an optimization method that combines chaotic local search with the principles of bacterial foraging to search for optimal solutions to problems.
ACGA (Chang, et al.,, 2008)	The Artificial Chromosomes Genetic Algorithm (ACGA) is an optimization method that combines the principles of genetic algorithms with the use of artificial chromosomes to encode and manipulate solutions.
CSO **(Li & Yin, 2013)**	The HCS algorithm combines the Cuckoo Search, a metaheuristic optimization method inspired by the behavior of cuckoo birds, with Lévy flights, a technique for random search that is based on the probability distribution of Lévy flights observed in nature.
BAT **(Bellabai et al., 2022)**	The BAT algorithm is a method that uses the echolocation system of bats as inspiration to solve problems. It involves individual search and movement towards "nutritious" locations with certain idealized rules being followed to optimize the process.
TLBO **(Shen et al., 2018)**	The Teaching-Learning Based Optimization (TLBO) algorithm is a meta-heuristic technique that has been used successfully on various non-linear programming and design optimization problems

Table 3. Comparison between SHOA, SSO, SCE-OBL, CLS-BFO, and ACGA

INSTANCE	(N*M)	BKS	SHOA	SSO	SCE-OBL	CLS-BFO	ACGA
REC1	20X5	1245	**1245**	1247	1249	1249	1249
REC3	20X5	1109	**1109**	**1109**	1111	1111	**1109**
REC5	20X5	1242	**1242**	1245	1245	1245	1245
REC7	20X10	1566	**1566**	1566	1584	1584	1566
REC9	20X10	1537	**1537**	**1537**	1545	1545	**1537**
REC11	20X10	1431	**1431**	1431	1431	1449	1431
REC13	20X15	1930	**1930**	1935	1963	1968	1935
REC15	20X15	1950	**1950**	1968	1993	1993	**1950**
REC17	20X15	1902	**1902**	1923	1944	1954	1911
REC19	30X10	2093	**2093**	2117	2156	2139	2099
REC21	30X10	2017	**2017**	2017	2064	2059	2046
REC23	30X10	2011	**2011**	2030	2067	2073	2021
REC25	30X15	2513	**2513**	2566	2584	2638	2545
REC27	30X15	2373	**2373**	2397	2445	2443	2396
REC29	30X15	2287	**2287**	2333	2364	2408	2304
REC31	50X10	3045	**3045**	3104	3179	3180	3105
REC33	50X10	3114	**3114**	3118	3154	3187	3140
REC35	50X10	3277	**3277**	3277	3281	3292	3277
REC37	75X20	4890	**4890**	5096	5327	5422	5193
REC39	75X20	5043	**5043**	5185	5391	5465	5276
REC41	75X20	4910	**4910**	5135	5334	5436	5208

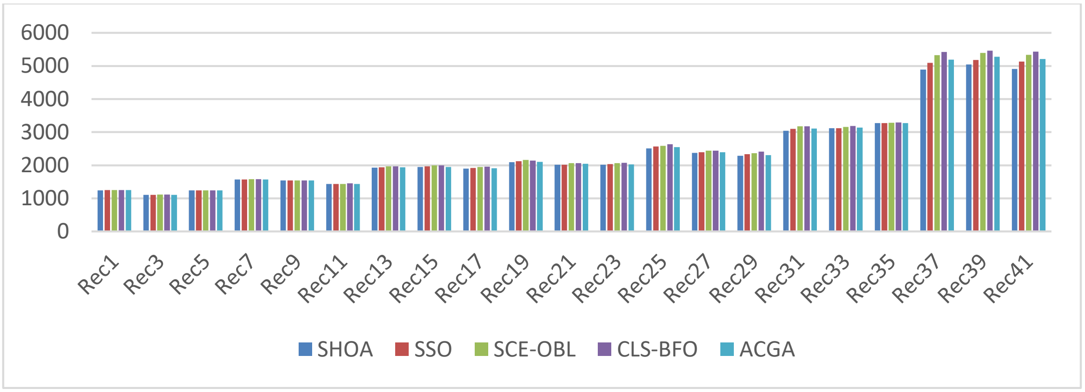


Figure 2. The comparison of best-obtained results

Table 4. Comparison between SHOA, CSO

	SHOA			CSO		
INSTANCES	**Best**	**Average**	**PDav**	**Best**	**Average**	**PDav**
TA001	1278	1279,022	0.08	1278	1278	0.00
TA011	1582	1584,057	0.13	1586	1603.8	1.3780
TA015	1419	1419	0.00	1426	1443.9	1.7548
TA021	2297	2301,364	0.19	2308	2319.9	0.9970
TA025	2291	2291	0.00	2312	2318.8	1.2141

	SHOA			CSO		
INSTANCES	**Best**	**Average**	**PDav**	**Best**	**Average**	**PDav**
TA031	2724	2726,751	0.101	2724	2725	0.0367
TA035	2863	2865,978	0.104	2863	2863.8	0.0279
TA040	2782	2782	0.00	2782	2782	0.00
TA041	2991	2991	0.00	3063	3074.9	2.8051
TA045	2976	2976	0.00	3035	3065.7	3.0141
TA051	3875	3875	0.00	3968	3978.6	3.4208
TA055	3635	3635	0.00	3750	3772.5	6.1799
TA061	5493	5493	0.000	5493	5493.8	0.0364
TA065	5250	5253,36	0.064	5255	5255	0.0952
TA071	5770	5770,115	0.002	5791	5802	0.5546
TA075	5468	5468	0.00	5512	5548.8	1.4963

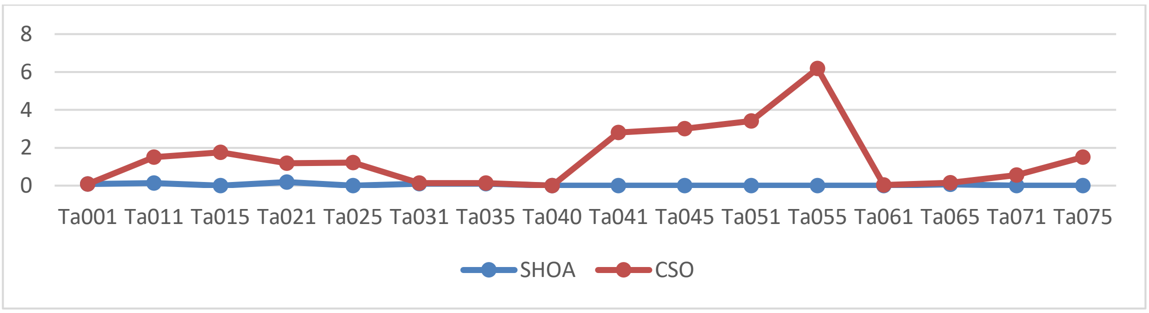


Figure 3. The comparison of average deviation results

Table 5. Comparison between SHOA, BAT and TLBO

		SHOA				BAT			TLBO		
INSTANCES	**BKS**	**BEST**	**AVERAGE**	**PDAV**	**BEST**	**AVERAGE**	**PDAV**	**BEST**	**AVERAGE**	**PDAV**
TA 001	**1278**	**1278**	1279,022	0.08	1278	1284.9	0.5399	1278	1287.2	0.7199
TA 011	**1582**	**1582**	1584,057	0.13	1609	1623.3	2.6106	1586	1606	1.5171
TA 021	**2297**	**2297**	2301,364	0.19	2323	2355.4	2.5424	2325	2344.7	2.0766
TA 031	**2724**	**2724**	2726,452	0.09	2724	2725.6	0.0587	2724	2729.4	0.1982
TA 041	**2991**	**2991**	2998,178	0.24	3119	3110.6	3.8449	3120	3141.4	5.0284
TA 051	**3771**	**3771**	3772,886	0.05	4001	4021.9	6.6534	3986	4029.7	6.8602
TA 061	**5493**	**5493**	5507,831	0.27	5493	5496.4	0.0619	5493	5499.4	0.1165
TA 071	**5770**	**5770**	5776,347	0.11	5808	5819.6	0.8596	5887	5928.7	2.7504

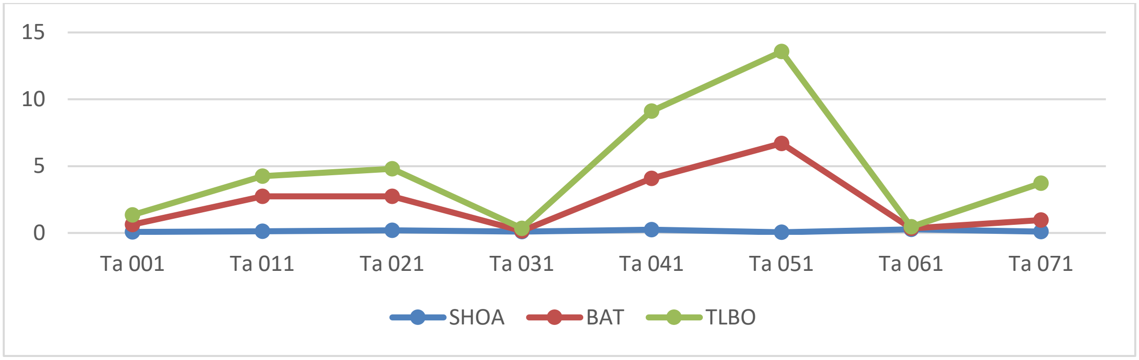


Figure 4. The comparison of average deviation result

7. Comparison and analysis

Based on Table 3, which compares SHOA, SSO, SCE-OBL, CLS-BFO, and ACGA, we can see that SHOA was able to find the optimum for all test instances, while the other metaheuristics were not. This is also supported by Figure 2, which shows that the values of SHOA are lower than those of the other methods.

However, the comparison between SHOA and CSO in Table 4 and Figure 3 shows that the results of SHOA are better than those of CSO, and the deviation curve of SHOA shows that, for each instance, the 20 executions gave results that were very close to the optimum, while CSO did not.

Finally, Table 5, which compares SHOA with BAT and TLBO, shows that SHOA found the optimum in all the tested instances, while BAT found the optimum for only 3 instances, and TLBO did as well. Figure 4 shows that the curve for SHOA is very small compared to those of BAT and TLBO, indicating that SHOA was able to find the optimum or a nearly optimal value for the 20 tested executions.

7.1. Wilcoxon Test and Statistical Analysis

These results can also be supported by the Wilcoxon test described below:

The Wilcoxon test is a non-parametric statistical test used to compare the medians of two groups. In the table above, the Wilcoxon test is used to compare the performance of the SHOA (Spotted Hyena Optimization Algorithm) with four other optimization algorithms: SSO (Scatter Search Optimization), SCE-OBL (Self-Adaptive Cuckoo Search with Orthogonal Bat Algorithm), CLS-BFO (Chaotic Levy Walk with Biogeography-Based Optimization), and ACGA (Adaptive Cooperative Genetic Algorithm).

The "N" column indicates the number of samples or instances used in the test. The "W+" and "W-" columns represent the sum of the ranks of the SHOA results that are better or worse than the results of the comparison algorithm, respectively. The "*p*-value" column shows whether the probability that the results obtained are due to chance or not. If the p-value is less than 0.05, the results are considered statistically significant and the null hypothesis (that there is no difference between the two groups) is rejected.

According to the results in Table 6, the SHOA algorithm significantly outperforms all four comparison algorithms on all of the test instances. This is indicated by the *p*-values being less than 0.05 and the "Significant" column showing "Yes" for all comparisons.

Table 6. Wilcoxon Signed test comparison

	N	W+	W-	Significant
SHOA VSSSO	21	0	-120	Yes
SHOA VS SCE-OBL	21	0	-210	Yes
SHOA VS CLS-BFO	21	0	-231	Yes
SHOA VS ACGA	21	0	-120	Yes

8. Conclusion

In conclusion, the results of this study demonstrate that the SHOA algorithm is an effective solution for the flow shop problem. It consistently found the optimal or nearly optimal solution across all test instances, while the other algorithms tested were unable to do so. These results were supported by both visual comparisons of the results and statistical analysis using the Wilcoxon test.

There are several possible avenues for future work in this area. One possibility is to further improve the SHOA algorithm by incorporating additional optimization techniques or adapting it to different variations of the flow shop problem. Another possibility is to test the algorithm on a wider range of test instances to ensure its robustness and generalizability. Additionally, it would be interesting to compare the performance of SHOA with other state-of-the-art optimization algorithms to see how it stands up against them. Finally, applying the SHOA algorithm to real-world flow shop problems and evaluating its performance in those settings could provide valuable insights and potential practical applications.

References

Ab Wahab, M. N., Nefti-Meziani, S., & Atyabi, A. (2015). A Comprehensive Review of Swarm Optimization Algorithms. PLOS ONE, 10(5), e0122827. https://doi.org/10.1371/journal.pone.0122827

Akhshabi, M., Tavakkoli-Moghaddam, R., & Rahnamay-Roodposhti, F. (2014). A hybrid particle swarm optimization algorithm for a no-wait flow shop scheduling problem with the total flow time. The International Journal of Advanced Manufacturing Technology, 70(5–8), 1181–1188. https://doi.org/10.1007/s00170-013-5351-9

Almeida, F. S. de, & Nagano, M. S. (2023). Heuristics to optimize total completion time subject to makespan in no-wait flow shops with sequence-dependent setup times. Journal of the Operational Research Society, 74(1), 362–373. https://doi.org/10.1080/01605682.2022.2039569

Anuar, S., Selamat, A., & Sallehuddin, R. (2016). A modified scout bee for artificial bee colony algorithm and its performance on optimization problems. Journal of King Saud University - Computer and Information Sciences, 28(4), 395–406. https://doi.org/10.1016/j.jksuci.2016.03.001

Ball, M. O. (2011). Heuristics based on mathematical programming. Surveys in Operations Research and Management Science, 16(1), 21–38. https://doi.org/10.1016/j.sorms.2010.07.001

Bellabai, J. R., Leela, B. N. M., & Kennedy, S. M. R. (2022). Testing the Performance of Bat-Algorithm for Permutation Flow Shop Scheduling Problems with Makespan Minimization. Brazilian Archives of Biology and Technology, 65. https://doi.org/10.1590/1678-4324-2022210840

Cai, L., Li, W., Luo, Y., & He, L. (2023). Real-time scheduling simulation optimisation of job shop in a production-logistics collaborative environment. International Journal of Production Research, 61(5), 1373–1393. https://doi.org/10.1080/00207543.2021.2023777

Dhiman, G., & Kumar, V. (2018). Multi-objective spotted hyena optimizer: A Multi-objective optimization algorithm for engineering problems. Knowledge-Based Systems, 150, 175–197. https://doi.org/10.1016/j.knosys.2018.03.011

Ghafori, S., & Gharehchopogh, F. S. (2022). Advances in Spotted Hyena Optimizer: A Comprehensive Survey. Archives of Computational Methods in Engineering, 29(3), 1569–1590. https://doi.org/10.1007/s11831-021-09624-4

Jorapur, V., Puranik, V. S., Deshpande, A. S., & Sharma, M. R. (2014). Comparative Study of Different Representations in Genetic Algorithms for Job Shop Scheduling Problem. Journal of Software Engineering and Applications, 07(07), 571–580. https://doi.org/10.4236/jsea.2014.77053

Kennedy, J., & Eberhart, R. (n.d.). Particle swarm optimization. Proceedings of ICNN’95 - International Conference on Neural Networks, 1942–1948. https://doi.org/10.1109/ICNN.1995.488968

Lenat, D. B. (1982). The nature of heuristics. Artificial Intelligence, 19(2), 189–249. https://doi.org/10.1016/0004-3702(82)90036-4

Liao, C.-J., Tjandradjaja, E., & Chung, T.-P. (2012). An approach using particle swarm optimization and bottleneck heuristic to solve hybrid flow shop scheduling problem. Applied Soft Computing, 12(6), 1755–1764. https://doi.org/10.1016/j.asoc.2012.01.011

Li, X., & Yin, M. (2013). A hybrid cuckoo search via Lévy flights for the permutation flow shop scheduling problem. International Journal of Production Research, 51(16), 4732–4754. https://doi.org/10.1080/00207543.2013.767988

Kurdi, M. (2021). Application of Social Spider Optimization for Permutation Flow Shop Scheduling Problem. Journal, 2(2), 85–97. https://dergipark.org.tr/en/pub/jscai/issue/66233/1013405

Musselman, K., O'Reilly, J., & Duket, S. (n.d.). The role of simulation in advanced planning and scheduling. Proceedings of the Winter Simulation Conference, 1825–1830. https://doi.org/10.1109/WSC.2002.1166474

Nagano, M. S., de Almeida, F. S., & Miyata, H. H. (2021). An iterated greedy algorithm for the no-wait flowshop scheduling problem to minimize makespan subject to total completion time. Engineering Optimization, 53(8), 1431–1449. https://doi.org/10.1080/0305215X.2020.1797000

Chang, P. C., Chen, S. H., Fan, C. Y., Chan, C. L. (2008). Genetic algorithm integrated with artificial chromosomes for multi-objective flowshop scheduling problems, Applied Mathematics and Computation, 205(2),550-561. https://doi.org/10.1016/j.amc.2008.05.027

Rossit, D. A., Toncovich, A., Rossit, D. G., & Nesmachnow, S. (2021). Solving a flow shop scheduling problem with missing operations in an Industry 4.0 production environment. Journal of Project Management, 33–44. https://doi.org/10.5267/j.jpm.2020.10.001

Shen, L., Dauzère-Pérès, S., & Neufeld, J. S. (2018). Solving the flexible job shop scheduling problem with sequence-dependent setup times. European Journal of Operational Research, 265(2), 503–516. https://doi.org/10.1016/j.ejor.2017.08.021

Yagmahan, B., & Yenisey, M. M. (2008a). Ant colony optimization for multi-objective flow shop scheduling problem. Computers & Industrial Engineering, 54(3), 411–420. https://doi.org/10.1016/j.cie.2007.08.003

Yagmahan, B., & Yenisey, M. M. (2008b). Ant colony optimization for multi-objective flow shop scheduling problem. Computers & Industrial Engineering, 54(3), 411–420. https://doi.org/10.1016/j.cie.2007.08.003

Zhao, F., Zhang, J., Wang, J. and Zhang, C. 2015. A shuffled complex evolution algorithm with opposition-based learning for a permutation flow shop scheduling problem. International Journal of Computer Integrated Manufacturing, 28(11),1220-1235. https://doi.org/10.1080/0951192X.2014.961965

Zhao, F., Liu, Y., Shao, Z., Jiang, X., Zhang, C. and Wang, J. 2016. A chaotic local search based bacterial foraging algorithm and its application to a permutation flow-shop scheduling problem. International Journal of Computer Integrated Manufacturing, 29(9), pp.962-981. https://doi.org/10.1080/0951192X.2015.1130240