FACTA UNIVERSITATIS
Series: **Mechanical Engineering** Vol. 22, Nº 1, 2024, pp. 77 - 100
https://doi.org/10.22190/FUME230615028M

Original scientific paper

HYBRID GENETIC AND PENGUIN SEARCH OPTIMIZATION ALGORITHM (GA-PSEOA) FOR EFFICIENT FLOW SHOP SCHEDULING SOLUTIONS

Toufik Mzili[1*], Ilyass Mzili[2], Mohammed Essaid Riffi[1], Dragan Pamucar[3,4], Vladimir Simic[5], Laith Abualigah[6,7,8,9], Bandar Almohsen[10]

[1]Department of Computer Science, Faculty of Science, Chouaib Doukkali University, EI Jadida, Morocco
[2]Department of Management, Laboratory of research in management and Development, Faculty of Economics and Management, Hassan First University, Settat, Morocco
[3]Department of Operations Research and Statistics, Faculty of Organizational Sciences, University of Belgrade, Belgrade, Serbia
[4]College of Engineering, Yuan Ze University, Taiwan
[5]Faculty of Transport and Traffic Engineering, University of Belgrade, Belgrade, Serbia
[6]Hourani Center for Applied Scientific Research, Al-Ahliyya Amman University, Jordan
[7]MEU Research Unit, Middle East University, Amman, Jordan.
[8]Applied science research center, Applied science private university, Amman, Jordan.
[9]Department of Electrical and Computer Engineering, Lebanese American University, Byblos, Lebanon
[10]Mathematics Department, College of Science, King Saud University, Saudi Arabia

ORCID iDs: Toufik Mzili https://orcid.org/0000-0002-5733-3119
Ilyass Mzili https://orcid.org/0000-0002-9096-8271
Mohammed Essaid Riffi https://orcid.org/0000-0003-1576-2174
Dragan Pamucar https://orcid.org/0000-0001-8522-1942
Vladimir Simic https://orcid.org/0000-0001-5709-3744
Laith Abualigah https://orcid.org/0000-0002-2203-4549
Bandar Almohsen https://orcid.org/0000-0002-2160-4159

Abstract. *This paper presents a novel hybrid approach, fusing genetic algorithms (GA) and penguin search optimization (PSeOA), to address the flow shop scheduling problem (FSSP). GA utilizes selection, crossover, and mutation inspired by natural selection, while PSeOA emulates penguin foraging behavior for efficient exploration. The approach integrates GA's genetic diversity and solution space exploration with PSeOA's rapid convergence, further improved with FSSP-specific modifications. Extensive experiments validate its efficacy, outperforming pure GA, PSeOA, and other metaheuristics.*

Key words: *Hybrid Metaheuristics, PSeOA, Scheduling Problem, Combinatorial Optimization, Artificial intelligence, Swarm intelligence.*

Received: June 15, 2023 / Accepted August 12, 2023
Corresponding author: Toufik Mzili
Department of Computer Science, Faculty of Science, Chouaib Doukkali University, Avenue Jabran Khalil Jabran, B.P 299-24000, El Jadida,Morocco
E-mail : mzili.t@ucd.ac.ma

1. INTRODUCTION

Combinatorial optimization problems [1] are a class of complex and diverse problems that encompass many application domains, such as logistics, production planning, communication network design, and many others. Among these problems is the multistage flow shop problem (FSSP), a classical scheduling problem that aims at optimizing the sequence of tasks to be performed on a set of machines to minimize a given criterion, such as the total task completion time (makespan) or the sum of the completion times of all tasks (total flow time).

Metaheuristics are high-level stochastic optimization approaches that aim at guiding the search process in the solution space by drawing on principles from nature, physics, or social systems. They have been widely used to solve large and complex combinatorial optimization problems because of their flexibility, adaptability, and ability to escape local optima. Among the most commonly used metaheuristics are genetic algorithms (GA) [2], particle swarm optimization (PSO) [3], rat swarm optimization [4], and ant colony optimization (ACO) [5].

Metaheuristics and hybrid metaheuristics offer several advantages for solving combinatorial optimization problems in various domains, including the flow shop problem. Some of the main advantages are:

Flexibility: Metaheuristics are generic optimization methods that can be applied to a wide variety of combinatorial optimization problems without requiring specific knowledge of the problem structure. This makes them particularly useful for solving problems from different domains.

Adaptability: Metaheuristics can be easily adapted to take into account the specificities and constraints of a particular problem. Dynamic adaptation mechanisms can be integrated to adjust the parameters of the algorithm during execution, thus improving the overall performance of the method.

Balanced exploration and exploitation: Metaheuristics are designed to balance exploration of the solution space (searching for new regions of the space) and exploitation of promising solutions (improving existing solutions). This balance allows metaheuristics to converge to high-quality solutions while avoiding local optima.

Robustness: Metaheuristics are often robust to perturbations and variations in the problem parameters. They can generally provide solutions of acceptable quality even in the presence of noise or uncertainty.

Efficiency: Hybrid metaheuristics, which combine the strengths of different metaheuristics, can offer better performance than individual metaheuristics. Hybridization exploits the advantages of each method, such as the genetic diversity of genetic algorithms, the fast convergence of particle swarm optimization algorithms, and the adaptability of ant colony optimization algorithms. This can lead to solutions and faster convergence.

In the field of optimization, numerous methods have been developed to solve various problems. These techniques are used in many fields, including engineering, economics and machine learning. These methods can be classified according to their principles and approaches. The classification presented highlights the distinct characteristics of each method, as well as their respective advantages and limitations. Fig.1 provides a visual summary of this classification, offering an overview that enables the different optimization methodologies to be explored in greater detail.

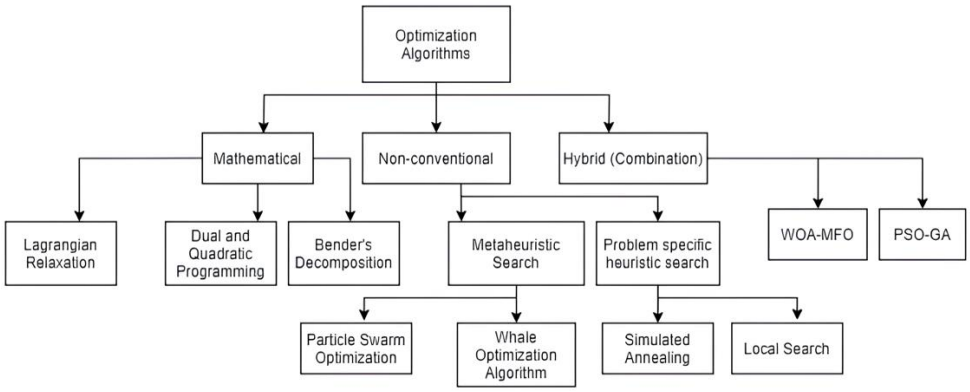

Fig. 1 Classification of Methods for solving optimization problems

In this paper, we focus on the hybridization of two popular and efficient metaheuristics, namely genetic algorithms and penguin swarm optimization (PSeOA), to solve the FSSP. Genetic algorithms inspired by the process of natural selection and evolution of species, use operators such as selection, crossover, and mutation to evolve progressively towards optimal or near-optimal solutions. On the other hand, penguin swarm optimization (PSeOA) is a recent and promising metaheuristic based on the foraging behavior of penguins. PSeOA combines individual and collective foraging behavior to explore the solution space and converges to optimal solutions through a dynamic adaptation mechanism.

The main contributions of this work can be summarized as follows:

Proposal of an innovative hybrid approach combining genetic algorithms (GA) and penguin swarm optimization (PSeOA) to solve the multistage flow shop problem (FSSP).

Exploring the hybridization of GAs and PSeOA, leveraging the strengths of each metaheuristic, including the genetic diversity and exploration of the solution space offered by GAs, and the rapid convergence and adaptability of PSeOA.

Modifications and improvements to the basic mechanisms of GAs and PSeOA to strengthen their performance in the context of the FSSP, including the adaptation of selection, crossover, and mutation operators for GA, as well as the introduction of dynamic adaptation mechanisms for PSeOA.

Extensive experimental evaluation of the proposed hybrid approach on a diverse set of multistage flow shop problems of different sizes and complexities, demonstrating its effectiveness and competitiveness over pure GAs and PSeOAs as well as other state-of-the-art metaheuristics.

Discussion of the findings and implications of this study for future research on metaheuristics applied to combinatorial optimization and, more specifically, to the FSSP, including the limitations of the hybrid approach and the possibilities for improvement and extension to address other combinatorial optimization problems and new challenges in the scheduling domain.

The rest of this paper is structured as follows. Section 2 presents a review of the literature on metaheuristics applied to the FSSP, focusing on genetic algorithms, PSeOA, and their hybridizations. Section 3 provides a detailed description of the flow shop problem, as well as the main mathematical formulations and evaluation criteria used. Section 4 presents the genetic

algorithms and PSeOA, explaining the key concepts, mechanisms, and parameters involved in each.

Section 5 describes the proposed hybrid approach, highlighting aspects of hybridization between genetic algorithms and PSeOA, as well as modifications and improvements made to improve the overall performance of the hybrid approach in the context of FSSP. Section 6 presents the experimental results obtained by applying the hybrid approach to a set of flow shop problems of different sizes and complexities. A comparison of the performance of the hybrid approach with that of pure GAs and PSeOAs as well as other state-of-the-art metaheuristics is also provided, demonstrating the effectiveness and competitiveness of the proposed hybrid approach.

Section 7 discusses the conclusions drawn from this study, as well as the implications for future research in the field of metaheuristics applied to combinatorial optimization and, more specifically, to the FSSP. We also discuss the limitations of the hybrid approach and the possibilities for improvement and extension to address other combinatorial optimization problems and new challenges in scheduling.

2. Related Work

The field of multi-objective optimization has become an important area of research, especially for planning problems in manufacturing. Various algorithms for multi-objective planning problems have been developed over the years, such as genetic algorithms (GA), particle swarm optimization (PSO), and ant colony optimization (ACO). This paper focuses on multi-objective planning in hybrid flow shop systems where the allocation of production resources considers multiple objectives.

Shao et al. [6] propose an original multi-objective evolutionary algorithm (MOEA-LS) for the multi-objective distributed hybrid flow shop scheduling problem (MDHFSP) with multiple neighborhood-based local searches, where each MDHFSP contains a set of factories. They solve the MDHFSP phase in a hybrid flow shop-scheduling problem using parallel machines. They start the solution with a sophisticated weighting mechanism and generate subsequent solutions using multiple local search operators. An adaptive weight update mechanism is also used to avoid stalling at the local optimum. Their results confirm the efficiency and effectiveness of the proposed algorithm in processing MDHFSP.

Han et al. [7] considered a hybrid flow shop scheduling problem with worker constraints (HFSSPW) and constructed a mixed integer linear programming model. They proposed seven multi-objective evolutionary algorithms with heuristic decoding (MOEAHs) to solve the problem. The results showed that the proposed MOEAHs performed excellently in terms of the makespan objective and demonstrated highly effective performance compared to other methods.

Amirteimoori et al [8] presented a parallel hybrid PSO-GA method for task and carrier organization in a flexible flow shop environment. The researchers used the Gurobi solver, parallel genetic algorithm (PGA), parallel particle swarm optimization (PPSO), and hybrid parallel PSO-GA approach (PPSOGA) to address the problem instance. The results show that the PPSOGA approach outperforms the other algorithms in terms of solution quality and computational efficiency.

Qiao et al. [9] proposed a genetic algorithm to adapt the flow to a warehouse system and planned a two-stage hybrid with a large setup time in the first stage and stop requirements in

the second stage. The researchers identified feasible planning conditions and created a local search method to improve the accuracy of the solution. The results showed that the algorithm could find a satisfactory solution within 1% of the lower bound in three minutes, proving its effectiveness.

Authors Vali et al. [10] introduced a flexible job shop scheduling problem with an aim to optimize patient flow and minimize the total carbon footprint. The authors proposed a metaheuristic optimization algorithm, named Chaotic Salp Swarm Algorithm Enhanced with Opposition-Based Learning and Sine Cosine (CSSAOS). The method was executed in a real-world scenario study and showed superior performance in contrast to other established metaheuristic algorithms.

Miyata and Nagano [11], in their paper "An Iterated Greedy Algorithm for Distributed Blocking Flow Shop with Setup Times and Maintenance Operations to Minimize Makespan," present the Variable Search Neighborhood (VNS) algorithm, referred to as IG_NM. This algorithm is designed to tackle the Distributed Blocking Flow Shop Scheduling Problem, taking into account sequence-dependent setup times and maintenance operations. A comparative study between the IG_NM algorithm and Mixed Integer Linear Programming (MILP), along with recent methodologies from the literature, was conducted through computational experiments. The outcomes indicate that IG_NM surpasses other literature-based metaheuristics.

Cui et al. [12] introduced an improved multi-population genetic algorithm (IMPGA) for tackling the distributed heterogeneous flow shop-scheduling problem (DHHFSP), a complex, NP-hard problem. The proposed algorithm features a strategic inter-factory neighborhood structure based on greedy job insertion and a novel move evaluation method for efficient neighborhood movements. The IMPGA also incorporates a guided sub-population information interaction and a re-initialization procedure with an individual resurrection strategy to enhance convergence speed and robustness. The results demonstrated that the IMPGA outperformed state-of-the-art algorithms for DHHFSP, proving its effectiveness in finding optimal solutions.

Furthermore, Hou et al. [13] investigated the distributed blocking flow-shop sequence-dependent scheduling problem (DBFSDSP) with the objectives of minimizing makespan, total tardiness, and total energy consumption. They proposed a cooperative whale optimization algorithm (CWOA) to solve the problem. Computational experimentation on an extensive benchmark demonstrated that CWOA outperforms state-of-the-art algorithms in terms of efficiency and significance in solving DBFSDSP.

3. Flow Shop Scheduling Problem

The Flow Shop Scheduling Problem (FSSP) [14] is a crucial component of organizing tasks over a specific period while taking into account temporal constraints (like deadlines and chaining constraints) and limitations on resource utilization and availability. A scheduling problem can be defined as follows: given a set of tasks that need to be executed and a collection of machines (resources) available for performing these tasks, the goal is to identify the most efficient sequence of tasks on the machines, thereby minimizing the total processing time for all tasks across all machines. The majority of scheduling problems are classified as NP-hard optimization problems.

Scheduling problems are typically defined by four main elements: tasks, resources, constraints, and objectives [15]. Hence, it is crucial to discuss these four fundamental elements before delving into scheduling problems

- **Tasks**

The manufacturing of products in a production workshop requires the execution of a set of elementary operations or tasks. A task is located in time by a start date t_i and/or by an end date c_i and an execution duration $p_i = c_i - t_i$ Some technical or economic constraints can associate with the task's earliest start dates r_i or latest end dates d_i.

- **Resources**

A resource is anything that is used to perform a task. They are available in a limited quantity and time. In the manufacturing context, resources can be machines, workers, equipment, facilities, energy, etc. In the case of the flow shop problem, a machine represents a resource.

- **Constraint**

Constraints are the conditions that must be met when constructing a schedule for it to be feasible.

- **Objective**

The objectives of companies are diversified and the scheduling process has become more and more multi-criteria. The criteria that must be satisfied by a scheduling process are varied. There are several classes of scheduling objectives:

Time-related objectives: for example, minimization of total execution time (C_{max}), average completion time, total set-up time or delay to delivery dates (L_{max} or T_{max})

Resource-related objectives: maximize the load on a resource or minimize the number of resources needed to complete a set of tasks.

Cost-related objectives: minimize launch, production, storage, transportation, etc. costs.

Energy or flow-related objectives.

Let $J=\{1,...,n\}$ be a set of n jobs. Each job must be executed on m machines in $M=\{1,...,m\}$ in the same order. Each job $j \in J$ consists of m ordered operations $O_{j1},...,O_{jm}$, and each operation must be executed on one of the m machines. Let $O=\{O_{ji},\ j \in \{1,...,n\}$ and $i \in \{1,...,m\}\}$ be the set of all operations to be ordered. Each operation Oji $\in O$ is associated with a fixed execution time p_{ji} The operations are interdependent by two types of constraints. Each machine can only process one job at a time and each job must pass through each machine only once. The flow shop problem (FSSP) consists in finding a feasible order that minimizes the makespan Cmax, i.e., the time needed to complete all the jobs.

An order can be represented by a vector of completion times of all operations ($C_{ji}, i \in \{1,...,m\}$ and $j \in \{1,...,n\}$). By adopting the following notations:

- O_{ji}: operation i of job j,
- P_{ji}: execution time of O_{ji},
- C_{ji}: Completion time of O_{ji}.

We can propose the objective function of FSSP, which is:

$$\text{Minimize } C_{max} = C_{n \times m} \tag{1}$$

4. METHODOLOGY

4.1. Inspiration

Penguins are seabirds that are adapted to aquatic life and cannot fly. Their wings are modified to function as flippers that help them swim, and they are highly efficient in the water, being able to dive more than 520 meters to search for food. While swimming underwater is less tiring for them, penguins still need to return to the surface regularly to breathe. They can breathe by swimming rapidly at speeds of 7 to 10 kilometers per hour. During dives, their heart rate slows down, and their hunting eyes remain open, with a protective nictitating membrane covering their cornea. Their retina enables them to distinguish shapes and colors.

Penguins primarily feed on fish and squid, and to do so, they hunt in groups and coordinate their dives to maximize their search for food. Penguins use vocalizations to communicate with one another, and each penguin's vocalizations are unique, like fingerprints in humans. This uniqueness allows for the identification and recognition of individual penguins within large colonies, where they can otherwise look quite similar. The amount of food required by penguins varies depending on their species, age, the variety of available food, and the amount of food available in their specific area. Studies have shown that a colony of 5 million penguins can consume up to 8 million pounds of krill and small fish daily.

4.2. Description of the Penguin Search Optimization Algorithm (PeSOA)

Penguin Search Optimization [16] introduces a novel metaheuristic inspired by the collaborative hunting strategies employed by penguins. This concept is particularly intriguing as penguins exhibit the remarkable ability to synchronize their diving patterns to enhance overall energy efficiency. The underlying foundation of optimizing feeding behavior typically revolves around an economic premise: if the energy acquired outweighs the energy expended to secure it, the pursuit becomes a fruitful endeavor.

Penguins, as living organisms, draw from this concept to gather insights into foraging time, costs, and prey energy content. These factors guide their decisions on whether to forage in a specific area, contingent upon the available resources and travel distances.

The algorithm's core principles are encapsulated in the following rules:

Rule 1:A penguin population is organized into multiple groups.

Rule 2: Each group accommodates a variable number of penguins, adaptable based on localized food availability.

Rule 3: Penguins collectively hunt and wander randomly, ceaselessly seeking sustenance while oxygen reserves permit.

Rule 4: Simultaneous dives to the same depth are possible.

Rule 5: Penguins within a group initiate search from a designated position ("hole i") and various depths ("j_1, j_2, ..., j_n").

Rule 6: Individual penguins within a group randomly forage and share findings with peers after several dives, fostering intra-group communication.

Rule 7: Each level can accommodate varying numbers of penguins based on food availability.

Rule 8: Inadequate food prompts a group, or even its entirety, to migrate to a different hole, ensuring inter-group communication.

Rule 9: The group with the highest fish consumption discloses the location of the prosperous food source, denoted by the hole and level.

In this algorithm, the penguins are characterised by their position in the holes and levels, and by the number of fish they have eaten. The distribution of penguins is based on the probability of fish being present in the various holes and levels. The penguins are grouped together and start their quests from random starting positions.

The penguins use a communication system to share information about the food they have found during their dives. Groups that have found little food follow those that have found a lot of food on their next dive. The penguins are divided into groups and search for food in holes at different levels. Their position is adjusted according to the previous solutions. After several dives, the penguins determine the best solution in terms of food found. The distribution probabilities of the holes and levels are then updated according to this best solution.

Using Eq. (2), an updated solution is calculated for each penguin in each group.

$$\mathrm{D_{new}} = \mathrm{D_{lastlast}} + \mathrm{rand} * |\mathrm{X_{localBest}} - \mathrm{X_{locallast}}| \tag{2}$$

Rand() is used to generate a random number for the distribution. There are three options: the best local solution, the most recent solution, and the new solution. The calculations of the updated solution equation (Eq. (2)) are repeated for every penguin in each group. After several dives, penguins communicate which solution worked best, represented by the number of fish consumed. Lastly, the probability of the new distribution of holes and levels is calculated.

4.3. Motivations

The effectiveness of a metaheuristic is associated with its ability to strike a balance between exploiting promising areas and exploring the global search space, as well as achieving an ideal balance between intensification and diversification. PSeOA demonstrates better control over local intensive search strategies and more efficient exploration of the entire search space. Additionally, the fewer parameters make PSeOA less complex and potentially more versatile.

Comparisons between PSeOA, PSO, and GA reveal that PSeOA is more robust and efficient than the other algorithms due to its group-based search strategy rather than solely relying on updating the next best-found position. Simulations also indicate that the distribution of penguins in the final step is well-balanced between the global and other local minima. Given a sufficiently large number of penguins, the algorithm can detect all local minima and the global minimum in the search space.

5. Proposed Hybrid and Discrete PeSOA

PeSOA is a metaheuristic defined for solving continuous optimization problems. The application of standard PeSOA to solve discrete (combinatorial) optimization problems is impossible, since there is a difference in the type of variable and also in the techniques for finding the solution. Consider the objective function of optimization problem $f(x)$. The variable x for a continuous optimization problem represents a real value but for a combinatorial optimization problem represents a scheduling or configuration. If we consider the traveling salesman problem, the variable x represents the Hamiltonian cycle.

the search space in the continuous case is an interval, while for discrete optimization, is the set of solutions meeting the constraints of the problem.

The work in this paper will focus on the design of the adaptation of PeSOA to combinatorial optimization problems. The result is a discrete version of PeSOA adaptable to various POCS. The adaptation process requires a decomposition of our problem into the set of key elements: group, penguin, position, objective function, and search space.

a) Group

- A group is a set of penguins
- The number of penguins in all groups is equal to the population size
- The number of groups in a population is not fixed.
- Individuals in a group are not stable.
- In a combinatorial optimization problem, a group represents a subset of the problem solutions.

b) Penguin

- A penguin is an individual of the population, the number of penguins equals the population size. A penguin is characterized by the position and quantity of food.

c) Position

- A position is a solution adopted by an individual in the population.
- Moving from one position to another is a movement in the given problem solution space.

d) Objective function

In PeSOA, the amount of food consumed by the penguin is used as the objective function.

In our problem, the objective function, or fitness, is a numerical value assigned to every solution in the search space. The most optimal solution is the one with the highest objective function value.

e) Search space

The search space represents the potential positions of the penguins. Generally, the positions are coordinates *(x,y)* in R^2 .to move to another position, it is enough simply, to modify the current position by adding a real value to (d') a coordinate (or two coordinates).

The search space in a combinatorial optimization problem can be represented by the given problem solutions and the neighboring solutions.

- **Coordinates:** The coordinates are the elements that directly affect the quality of the solution through the objective function. This function must be well defined in order to simplify the representation of the coordinates.
- **Displacement:** In the combinatorial case, the coordinates of a solution in the search space are changed through the properties of the problem being treated. In general, the change of position in the combinatorial space is done by a change in the order of the elements of the solution, by a combination, a permutation, or a set of methods or operators called perturbations or movement (purple).

- **Neighborhood:** The notion of neighborhood requires that the neighboring solution of a given solution must be generated by the smallest possible perturbation. This perturbation must make the minimum of changes to the current solution. It is, therefore, necessary to specify subsets of solutions called regions, according to the metric of the search space.
- **Step:** The step is the distance between two solutions. It is based on the topology of the space and the notion of neighborhood. In this work, we have classified the steps according to their length, which is the nature and the number of successive perturbations.

5.1. Adaptation of GA-PeSOA to FSSP

The objective of solving the Flow Shop Scheduling Problem (FSSP) is to determine a feasible task sequence for a set of machines that optimizes an objective function. The hybrid Genetic and Penguin Search Optimization Algorithm (GA-PeSOA) adopts a search procedure to obtain optimal solutions when solving FSSP. GA-PeSOA has been utilized to solve FSSP to illustrate its superior effectiveness when compared to other metaheuristic algorithms. This section will demonstrate how to represent a solution in the search space and how to transition from the current solution to a new one.

5.2. FSSP Solution

In the AG-PeSOA context, the location of a penguin in a population is considered as a potential solution in the search space.

Take, for example, Table 1, which illustrates a 4-task, 3-machine scheduling problem. Each operation is linked to a particular machine with its corresponding processing time.

Table 1 Example for 4 × 3 FSSP Instance

M1	M2	M3
6	2	4
3	6	2
1	3	1
2	1	5

In the context of the FSSP, a sequence of tasks representing a solution consists of a permutation of tasks, which is essentially a sequence of n integers denoted by S = {1, 2, 3, ..., n}. This sequence is symbolically represented by S = {1, 2, 3, ..., n}. Each integer "i" in the sequence represents a task index, which is responsible for organizing the order of tasks on the designated machines according to their structural arrangement within "S". For example, let's consider a hypothetical initial solution: 2-1-4-3, which specifies the sequential execution of tasks on each machine.

The information in Table 1, together with the initial solution provided, forms the basis for creating a visual representation similar to a Gantt chart (as shown in Figure 2). This visual representation is essential for calculating the objective function (Cmax) corresponding to the specific sequence (2-1-4-3). This analytical process results in an important measure, which is a significant reflection of the solution's efficiency. In this context, the determined value of 20 represents the efficiency of the solution.

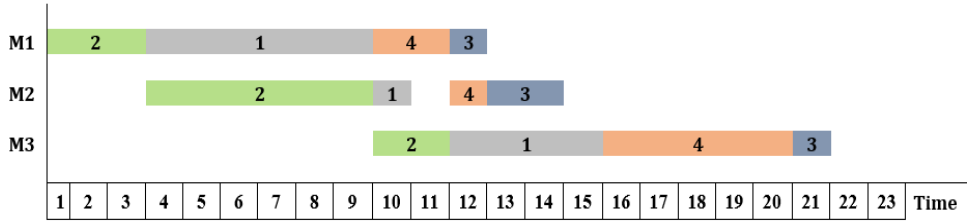

Fig. 2 An example of a Gantt chart for the initial 2-1-4-3 solution

5.3. Moving in Space

Moving from one solution to another in the search space involves an operation based on concepts of length and topology. Length is represented by the cost of the solution, and techniques to move through topology are performed by the operators of Equation 3.

$$S_{new} = S_{id} + r * (S_{best} - S_{id}) \quad (3)$$

5.4. Operation Subtraction (-)

This operation involves two solutions, S_1 and S_2, and yields a displacement vector Q. It also extracts the permutations that were applied to S_2 to obtain S_1:

let $S_1 = \{j_1,j_2,j_3,j_4,\dots j_{n-1},j_n\}$ and $S_2 = \{j_2,j_3,j_1,j_4,\dots j_n,j_{n-1}\}$

$Q=S_1-S_2$ then $Q= \{(j_1,j_2),(j_2,j_3),\dots,(j_n,j_{n-1})\}$

In other words, $S1-S_2=Q'$ $S_1+Q=S_2$

5.5. Operation Addition ⊕

The operation involves solution S_2 and vector Q, resulting in a new solution S_{new}.

This involves applying permutations of vector Q to solution S_2 to obtain a new solution S_{new}, as exemplified by:

Either $S_2 = \{j_1,j_2,j_3,j_4,j_5,\dots,j_n\}$ and $Q = \{(3,1),(4,3),(2,5)\}$

$S_{new} = S_2 \oplus Q$ then $S_{new} = \{j_4,j_5,j_1,j_3,j_2,\dots,j_n\}$

5.6. Operation Multiplication ⊗

The operation involves a multiplication between a real random number r $(r\epsilon[0,1])$ and a displacement vector Q. Its role is to decrease the number of permutations of the vector Q based on the value of r.

We consider a displacement vector Q with n permutations.

- $Q=\{(c_1,c_2),(c_3,c_4),(c_5,c_6)\dots,(c_{n-1},c_n)\}$ and real number $0\leq r\leq 1$.
- $Q'=r\otimes Q$ then $m= n\times r \leq n$, $Q'=\{(c_1,c_2),(c_3,c_4),\dots,(c_{m-1},c_m)\}$
- The fig. 3 describe the mathematical operators.

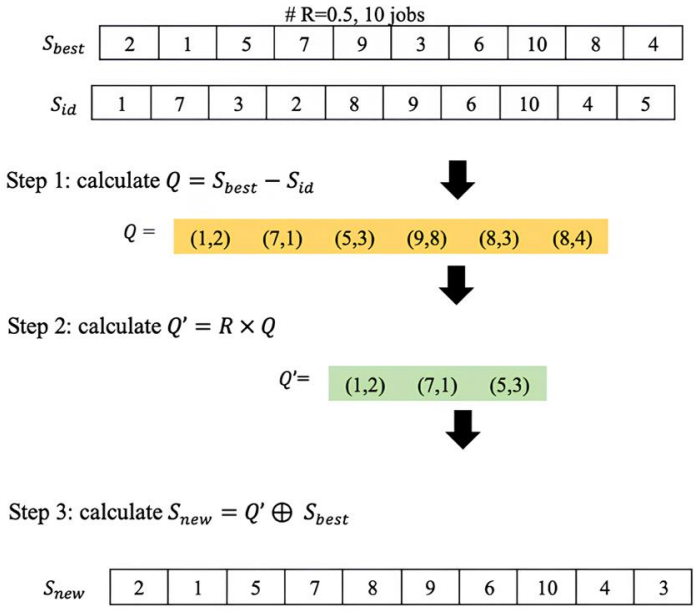

Fig. 3 Illustration of the function of the GA-PeSOA operators at the FSSP

5.7. Genetic Algorithm Operators

In GA-PeSOA, the generation of new solutions for solving the FSSP is performed by combining the operators of the genetic algorithm and the PeSOA mechanism. The genetic algorithm operators (selection, crossover, and mutation) are used to create a new population. Two different solutions are chosen from the population to represent the parents of the new population, and crossover is applied to produce two new offspring (solutions). Crossover is used to recombine the genetic makeup of two solutions (parents) to produce two solutions that inherit the characteristics of their parent solutions. After the crossover operation, a mutation is applied to the solutions to maintain diversity within the population. The structures of the genetic algorithm operations (crossover, mutation) are illustrated in Fig. 4.

In the example in Fig. 4, we have two parent solutions (parent1, parent2). Suppose that the two randomly chosen positions for the crossover are 4 and 7; we then obtain the children child1 and child2, but child1 and child2 are not legitimate. We consider the duplicated tasks in the child 1 as superfluous tasks, which are 2 and 10, and deprived tasks, which are 3 and 9. In child 2 the duplicated tasks are 3 and 9, and the deprived tasks are 2 and 10. Then we exchange them to create the new legitimate children child1' and child2'.

The mutation operator creates random changes in the order of the tasks. To do this, we randomly select a solution and two positions, then swap the first position of one task with the second position of another task. Fig. 4 shows an example of mutating a child1 solution with positions 6 and 10.

To integrate the PeSOA mechanism with the genetic algorithm, we can use the PeSOA search process to guide the exploration of the solution space. For instance, after applying the genetic algorithm operators and obtaining new offspring, we can employ PeSOA to refine the offspring's solutions further. This can be achieved by simulating the penguins'

collaborative hunting strategy and synchronizing their dives to optimize global energy. By combining both approaches, GA-PeSOA can potentially find better solutions for the FSSP than using either method alone.

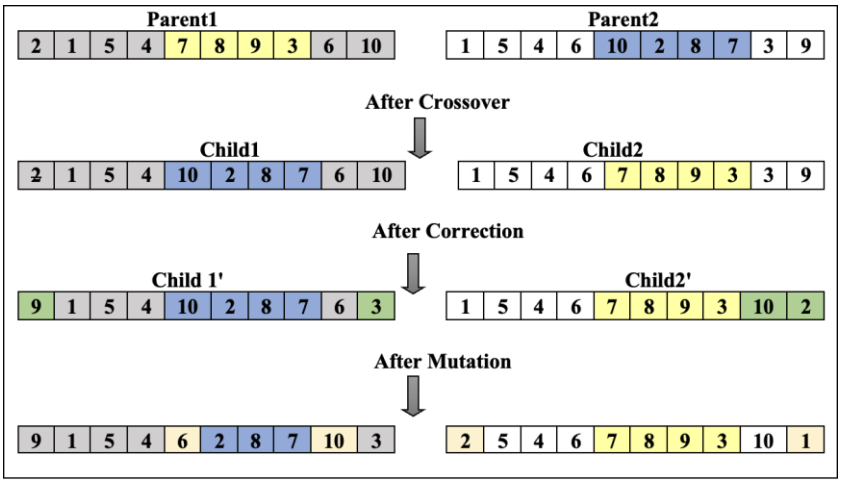

Fig. 4 Example of application of GA operators (crossover - mutation)

5.8. The 2-opt Local Search Technique Operator

The 2-opt technique is a local search optimization method commonly used to solve the Traveling Salesman Problem (TSP). However, it can also be adapted to the shop floor scheduling problem (FSSP) to improve the solutions generated by the GA-PeSOA algorithm.

In the FSSP, the objective is to find an optimal schedule of tasks to be processed on a set of machines. Suppose we have a current schedule (solution) represented by a permutation of tasks and we want to improve it using the 2-opt technique.

Here is a step-by-step example of how to apply the 2-opt technique to update a penguin's solution for FSSP:

1. Begin with the current schedule (solution) for a penguin: [1, 2, 3, 4, 5, 6]
2. Select two random positions in the permutation (excluding the first and the last positions). For example, positions 2 and 4 (tasks 2 and 4).
3. Reverse the order of the tasks between the two selected positions, resulting in a new schedule: [1, 4, 3, 2, 5, 6]
4. Calculate the objective function value (e.g., makespan) for both the original and the new schedules.
5. If the new schedule has a better objective function value, accept it as the new solution for the penguin. Otherwise, keep the original schedule.
6. Repeat steps 2-5 for a predefined number of iterations or until a termination criterion is met (e.g., no improvement in the objective function value for a certain number of consecutive iterations).

By applying the 2-opt technique to each penguin's solution for the FSSP, we can potentially find better schedules by exploring local neighborhoods of the current solutions. This can be a helpful addition to the GA-PeSOA algorithm, allowing it to further refine and improve the generated solutions.

The final algorithm incorporates all the mechanisms described as follows:

Algorithm 3: Discrete PeSOA

1: Generate a random population of P solutions $X=\{x_1,x_2,x_3,...,x_p\}$;
2: Objective function $f(x_i), x_i \in \{x_1,x_2,x_3,...,x_p\}$;
3: Select the best solution from P (x_{best} and x_{gbest}).
4: **While** ($t < MaxCeneration$) or (the stop criterion) **do**
5: **For** $i=0$ *to* P **do**
6: **While** ($reserve\_oxygene > 0$) **do**
7: Calculate $x_i(t+1)$ using equation $x_i(t+1) = x_i(t+1) + rand*(x_{best} - x_i(t))$
8: **End While**
9: **If** ($f(x_i(t+1)) < f(x_{best}(t+1))$) then
10: $x_i(t+1) = x_i(t)$
11: **End if**
12: **Else then**
13: Cross-over the current solution x_i^t with $x_{best}(t+1)$);
14: Update the $x_i(t+1)$ by the best child of crossing $x_i(t)$ and $x_{best}(t+1)$.
15: **End Else**
16: $x_i(t+1) = 2\text{-}opt(x_i(t))$
17: **End for**
18: Select the best $x_i(t+1)$)
19: Updating population solutions $x(t)$ by $x(t+1)$
20: **If** ($f(x_{best}(t+1)) < f(x_{best})$) then
21: $x_{best} = x_{best}(t+1)$
22: **End if**
23: **If** ($f(x_{best}) < f(x_{gbest})$) then
24: $x_{gbest} = x_{best}$
25: **End if**
26: **End While.**
27: Post-processing of results and visualization

6. Experiments and Calculation Results

In this section, an evaluation was carried out on the GA-PeSOA using a set of test cases taken from the OR library and executed in the C++ programming language. Data for these experiments was collected on a MacBook Air equipped with an Intel M1 processor and 8.00 GB RAM.

The results of this experiment are shown in Tables 4, 5, 6 and 7, revealing the numerical results as follows: The initial column serves to identify the instance under examination, called "Instance", while the "n × m" column elucidates the specific configuration involving n tasks spread over m machines.

The "BKS" column gives an overview of the most optimal results, as indicated by the alternative algorithms.

The "Best" columns effectively summarize the outstanding results obtained by applying the GA-PeSOA approach to the particular instances in question.

The GA-PeSOA parameter values are shown in Table 2 :

Table 2 Values of the parameters used

Parameters	Meaning	values
MaxCeneration	Maximum number of iterations	1000
P	Population size	60
RO2	Oxygenated reserve	10

To evaluate the performance of the GA-PSeOA, we will compare its results with those of various existing algorithms, as listed in Table 3. In Table 4, 5, 6, and 7, we present a comparison between GA-PSeOA and other methods.

Additionally, Figs. 5-8 illustrate the graphical representation of the corresponding results.

Table 3 Nomenclature used

Abbreviation	Full Name	Source
ISDH	Improved std dev heuristic	[17]
ISDH-LS	Improved std dev heuristic with local search	
IBH	Improved best-so-far heuristic	
GA	Genetic Algorithm	
SSO	Social Spider Optimization (SSO) algorithm	[18]
NS-SGDE	Self-guided Differential Evolution with Neighborhood Search	[19]
HWA	Hybrid Whale Optimization Algorithm	[20]
MASC	Memetic Algorithm with Novel Semi-Constructive Evolution Operators	[21]
NEH	Nawaz-Enscore-Ham	[22]
SGA	Standard Genetic Algorithm	
NEH-NGA	Nawaz-Enscore-Ham with Neighborhood Generation Algorithm	
IHSA	Improved Harmony Search Algorithm	[23]
PSO	Particle Swarm Optimization	[17]

Table 4 Comparison in small instances

Instance	(nxm)	Bks	NEH	SGA	NEH-NGA	IHSA	PSO	GA-PSeOA
Car01	11 × 5	7038	7038	7038	7038	N/A	N/A	7 038
Car02	13 × 4	7166	7376	7166	7166	N/A	N/A	7 166
Car03	12 × 5	7312	7399	7312	7312	N/A	N/A	7 312
Car04	14 × 4	8003	8129	8003	8003	N/A	N/A	8 003
Car05	10 × 6	7720	7835	7720	7720	N/A	N/A	7 720
Car06	8 × 9	8505	8773	8505	8505	N/A	N/A	8 505
Car07	7 × 7	6590	6590	6590	6590	N/A	N/A	6 590
Car08	8 × 8	8366	8564	8366	8366	N/A	N/A	8 366
Rec01	20 × 5	1247	1320	1249	1247	17 874	19 556	1 247
Rec03	20 × 5	1109	1116	1111	1109	15 098	17 417	1 109
Rec05	20 × 5	1242	1296	1245	1242	17 793	19 210	1 242
Rec07	20 × 10	1566	1626	1584	1566	25 647	28 407	1 566
Rec09	20 × 10	1537	1583	1561	1537	24 347	26 796	1 537
Rec11	20 × 10	1431	1550	1473	1438	22 706	25 362	1 431
Rec13	20 × 15	1930	2002	1956	1935	33 136	36 669	1 930
Rec15	20 × 15	1950	2025	1982	1950	33 066	35 905	1 950
Rec17	20 × 15	1902	2019	1959	1907	31 901	35 215	1 902
Rec19	30 × 10	2093	2185	2175	2098	51 080	59 231	2 093
Rec21	30 × 10	2017	2131	2076	2022	48 935	57 782	2 017
Rec23	30 × 10	2011	2110	2070	2016	47 921	56 316	2 011
Rec25	30 × 15	2513	2644	2636	2518	65 926	76 201	2 513
Rec27	30 × 15	2373	2505	2470	2378	63 788	73 432	2 373
Rec29	30 × 15	2287	2391	2415	2292	59 655	N/A	2 287
Rec31	50 × 10	3045	3171	3249	3150	118 184	N/A	3 045
Rec33	50 × 10	3114	3241	3189	3149	125 914	N/A	3 114
Rec35	50 × 10	3277	3313	3327	3282	124 035	N/A	3 277

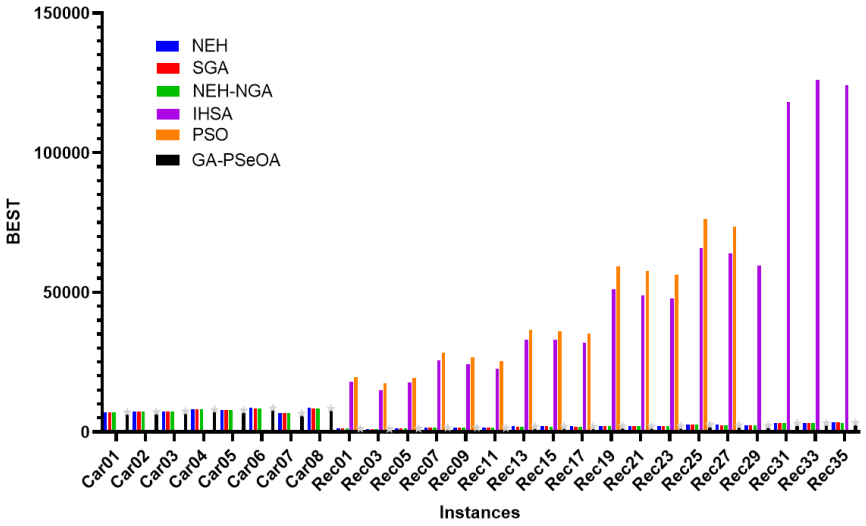

Fig 5. Comparison of the best-obtained solution for small instances

Table 5 Comparison in complex instances (50x5) from [24]

Instance	BKS	ISDH	ISDH-LS	IBH	GA	SSO	NS-SGDE	HWA	MASC	GA-PSeOA
tail31	2724	82 183	79 471	79 562	80 701	2724	2724	2724	2724	2724
tail32	2834	90 846	88 454	86395	86 105	2839	2834	2834	2834	2834
tail33	2621	2 738	82 218	81 122	80 561	2621	2621	2621	2621	2621
tail34	2751	86 173	84 250	83 257	84 991	2753	2751	2751	2751	2751
tail35	2863	87 367	85 680	85 763	86 789	2863	2863	2863	2863	2863
tail36	2829	89 192	85 739	86 354	84 781	2832	2829	2829	2829	2829
tail37	2725	85 884	82 335	83 010	81 998	2725	2725	2725	2725	2725
tail38	2683	85 103	83 365	84 082	81 934	2703	2683	2683	2683	2683
tail39	2552	80 444	79 978	77 992	77 916	2561	2552	2552	2552	2552
tail40	2782	88 675	85 946	84 142	85 670	2782	2782	2782	2782	2782

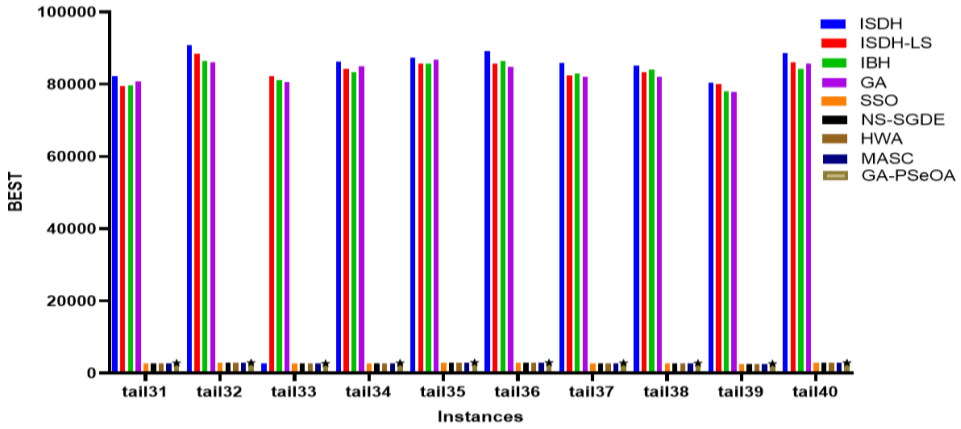

Fig. 6 Comparison of the best-obtained solution for complex instances (50x5)

Table 6 Comparison in complex instances (50x10) from [24]

Instance	BKS	ISDH	ISDH-LS	IBH	GA	SSO	NS-SGDE	HWA	MASC	GA-PSeOA
tail41	2991	12 0090	11 6969	11 6122	11 7654	3053	3021	3021	3024	2991
tail42	2867	11 8203	11 3873	11 2619	11 7445	2938	2896	2891	2882	2867
tail43	2839	11 7403	11 4235	11 4880	11 0999	2890	2888	2869	2852	2839
tail44	3063	12 2769	11 8586	11 6836	11 7599	3071	3075	3063	3063	3063
tail45	2976	12 0773	12 0242	11 9499	12 0528	3024	3027	3001	2982	2976
tail46	3006	12 0201	11 6570	11 8467	11 6090	3050	3029	3006	3006	3006
tail47	3093	12 2457	11 9751	12 0075	12 2151	3133	3124	3126	3099	3093
tail48	3037	11 6975	11 6003	11 5720	11 8636	3046	3055	3046	3038	3037
tail49	2897	11 8063	11 6323	11 6140	11 5648	2927	2928	2897	2902	2897
tail50	3065	12 1804	11 8031	11 6781	11 8053	3131	3092	3078	3077	3065

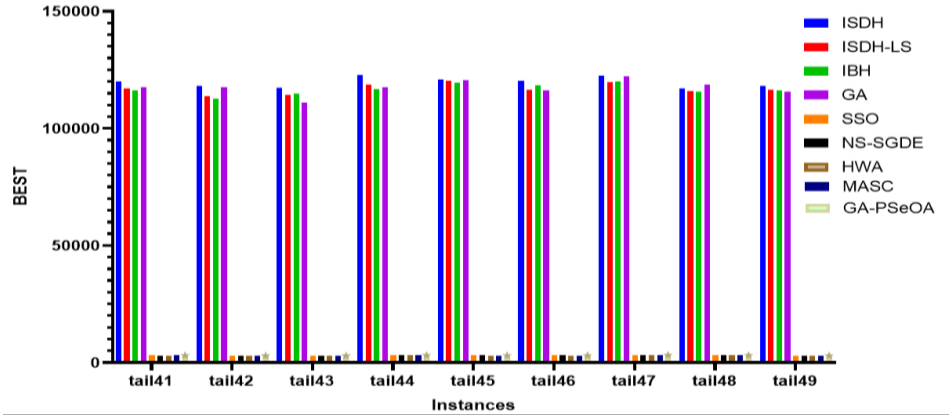

Fig. 7 Comparison of the best-obtained solution for complex instances (50x10)

Table 7 Comparison for complex instances (50x20) from [24]

Instance	BKS	ISDH	ISDH-LS	IBH	GA	SSO	NS-SGDE	HWA	MASC	GA-PSeOA
tail51	3850	178954	173683	172254	178630	3974	3916	3876	3889	3850
tail52	3704	169880	166390	166792	166887	3808	3744	3715	3720	3704
tail53	3640	175244	172739	170554	167089	3772	3702	3653	3667	3640
tail54	3720	172895	168989	171055	167904	3849	3793	3755	3754	3720
tail55	3610	172514	166783	169655	173415	3746	3677	3649	3644	3610
tail56	3681	172492	169714	170539	168755	3795	3743	3703	3708	3681
tail57	3704	177382	171602	174218	173165	3835	3784	3723	3754	3704
tail58	3691	169268	165782	165601	173020	3829	3757	3704	3711	3691
tail59	3743	174213	169524	171078	172826	3870	3795	3763	3772	3743
tail60	3756	178270	173446	173635	175483	3875	N/A	3767	3778	3756

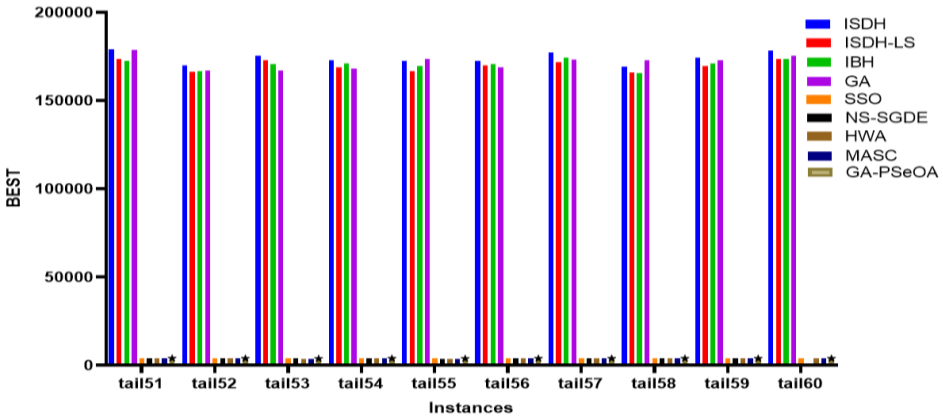

Fig. 8 Comparison of the best-obtained solution for complex instances (50x20)

6.1. Discussion

The following section presents a comprehensive comparison of various algorithms, including the Hybrid Genetic Algorithm and Penguin Search Algorithm (GA-PSeOA), on distinct instances and complexities of scheduling problems. The best-known solutions (BKS) are provided as a benchmark for each instance.

6.1.1. Comparison of Distinct Instances of the Scheduling Problem

Table 4 compares the performance of GA-PSeOA with other algorithms (NEH, SGA, NEH-NGA, IHSA, and PSO) on distinct instances of the scheduling problem. GA-PSeOA demonstrates exceptional performance, often matching the best-known solutions and consistently outperforming IHSA and PSO when results are available. Fig. 5 further illustrates GA-PSeOA's consistent capability to achieve high-quality solutions through a low curve representing the best solution values discovered across all instances.

6.1.2. Comparison of Complex Instances (50x5)

Table 5 presents a comparison of GA-PSeOA with other algorithms (ISDH, ISDH-LS, IBH, GA, SSO, NS-SGDE, HWA, and MASC) on complex instances (50x5) from Taillard (1993). GA-PSeOA consistently matches or closely approaches the best-known solutions for each instance. Fig. 6 displays GA-PSeOA's consistent ability to obtain high-quality solutions through a small curve representing the best solution values found across all instances.

6.1.3. Comparison of Complex Instances (50x10)

Table 6 compares GA-PSeOA with other algorithms (ISDH, ISDH-LS, IBH, GA, SSO, NS-SGDE, HWA, and MASC) on complex instances (50x10) from Taillard [24]. GA-PSeOA again demonstrates exceptional performance, consistently matching or closely approaching the best-known solutions for each instance. Fig. 7 further emphasizes GA-PSeOA's consistent ability to obtain high-quality solutions through a small curve representing the best solution values found across all instances.

6.1.4. Comparison of Complex Instances (50x20)

Table 7 compares GA-PSeOA with other algorithms (ISDH, ISDH-LS, IBH, GA, SSO, NS-SGDE, HWA, and MASC) on complex instances (50x20) from Taillard [24]. GA-PSeOA consistently matches the best-known solutions, underlining its ability to efficiently tackle complex scheduling problems. Fig. 8 highlights GA-PSeOA's consistent capability to obtain high-quality solutions through a small curve representing the best solution values found across all instances.

6.2. Validation of GA-PSeOA Performance Using the ANOVA Test

The performance of the GA-PSeOA method is validated using the ANOVA test, followed by Dunnett's multiple comparison tests. These tests compare the performance of GA-PSeOA with other algorithms by examining the average differences. The following information is provided for each comparison:

- Dunnett's multiple comparison tests: Algorithm pairs being compared (GA-PSeOA vs. another algorithm).
- Mean Diff: The average difference between the two compared algorithms.
- 95.00% CI of difference: The 95% confidence interval for the difference between the mean values.
- Below threshold? Indicates if the difference is below the threshold of significance (Yes or No).
- Adjusted P Value: The adjusted p-value for comparison.

6.2.1. Cases of lower complexity

In Table 8, the ANOVA test with Dunnett's multiple comparison tests is used to examine the mean differences in performance between GA-PSeOA and four other algorithms (NEH, SGA, NEH-NGA, IHSA, and PSO). The test yields a p-value, which indicates the statistical significance of the differences between the means.

Table 8 ANOVA test comparison for less complex instances

Dunnett's multiple test	Mean Diff,	95,00% CI of diff,	Below threshold?	Adjusted P Value
GA-PSeOA vs. NEH	-99,54	-132,5 to -66,55	Yes	<0,0001
GA-PSeOA vs. SGA	-41,65	-69,07 to -14,24	Yes	0,0017
GA-PSeOA vs. NEH-NGA	-7,385	-18,56 to 3,789	No	0,2950
GA-PSeOA vs. IHSA	-47756	-69484 to -26029	Yes	<0,0001
GA-PSeOA vs. PSO	-36792	-50266 to -23317	Yes	<0,0001

The test results reveal that, except for NEH-NGA, there are significant average differences between GA-PSeOA and all other algorithms. The average difference between GA-PSeOA and NEH is -99.54, with a 95% confidence range spanning from -132.5 to -66.55. The average difference between GA-PSeOA and SGA is -41.65, with a 95% confidence range spanning from -69.07 to -14.24. The average difference between GA-PSeOA and IHSA is -47,756, with a 95% confidence range of -69,484 to -26,029. The average difference between GA-PSeOA and PSO is -36,792, with a 95% confidence range between -50,266 and -23,317. The test also indicates that there is no statistically significant difference between the mean values of GA-PSeOA and NEH-NGA. This implies that there is no evidence to suggest that GA-PSeOA is more or less effective than NEH-NGA.

In conclusion, the ANOVA test with Dunnett's multiple comparisons demonstrates that GA-PSeOA is significantly more effective than NEH, SGA, IHSA, and PSO, but not significantly different from NEH-NGA. The results suggest that GA-PSeOA is a more competitive approach than other algorithms for the given task.

The QQ-Plot in Fig. 9 further illustrates the distribution of average differences between GA-PSeOA and the other algorithms, highlighting the distinct behavior of GA-PSeOA compared to other researched algorithms.

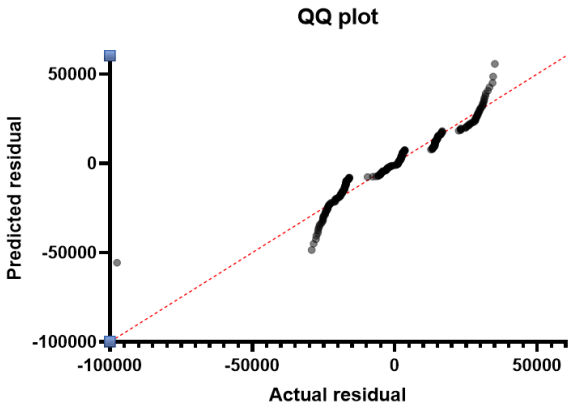

Fig. 9 QQ-Plot distribution displaying the distinct performance of GA-PSeOA compared to other algorithms for less complex instances

6.2.2. Cases of higher complexity

The differences in mean performance between GA-PSeOA and the other seven algorithms (ISDH, ISDH-LS, IBH, GA, SSO, NS-SGDE, HWA, and MASC) are compared using Dunnett's multiple comparison test. This test gives a p-value that indicates the statistical significance of the differences between the means.

Table 9 ANOVA test comparison for the more complex instances

Dunnett's multiple test	Mean Diff,	95,00% CI of diff,	Below threshold?	Adjusted P Value
GA-PSeOA vs. ISDH	-120877	-143540 to -98213	Yes	<0,0001
GA-PSeOA vs. ISDH-LS	-120600	-139703 to -101498	Yes	<0,0001
GA-PSeOA vs. IBH	-120558	-139943 to -101174	Yes	<0,0001
GA-PSeOA vs. GA	-121177	-140849 to -101505	Yes	<0,0001
GA-PSeOA vs. SSO	-57,10	-85,64 to -28,57	Yes	<0,0001
GA-PSeOA vs. NS-SGDE	-8,283	-27,30 to 10,73	No	0,7390
GA-PSeOA vs. HWA	-12,41	-19,40 to -5,428	Yes	0,0002
GA-PSeOA vs. MASC	-13,00	-20,98 to -5,024	Yes	0,0006

The test results reveal that, except for NS-SGDE, there are significant average differences between GA-PSeOA and all other algorithms. The average difference between GA-PSeOA and ISDH is -120,877, with a 95% confidence interval ranging from -143,540 to -98,213. The average difference between GA-PSeOA and ISDH-LS is -120,600, with a 95% confidence interval spanning from -139,703 to -101,498. The average difference between GA-PSeOA and IBH is -120,558, with a 95% confidence interval between -139,943 and -101,174.

The average difference between GA-PSeOA and GA is -121,177, with a 95% confidence interval covering -140,849 to -101,505. The average difference between GA-PSeOA and SSO is -57.10, with a 95% confidence interval ranging from -85.64 to -28.57. The average difference between GA-PSeOA and HWA is -12.41, with a 95% confidence

interval between -19.40 and -5.428. The average difference between GA-PSeOA and MASC is 13.00, with a 95% confidence interval ranging from 20.98 to 5.024.

The test also indicates that the mean difference between GA-PSeOA and NS-SGDE is not significant, meaning that there is no evidence to suggest that GA-PSeOA is more or less effective than NS-SGDE.

In conclusion, Dunnett's multiple comparison test demonstrates that GA-PSeOA performs significantly better than ISDH, ISDH-LS, IBH, GA, SSO, HWA, and MASC, but is not significantly different from NS-SGDE. These results suggest that GA-PSeOA is a competitive method in comparison to other algorithms for the given task.

In Fig. 10 the QQ plot displays a bimodal distribution of performance differences between GA-PSeOA and the other algorithms, indicating that GA-PSeOA performs considerably better than other algorithms in some cases, while its results are significantly worse in others. The points furthest from the edges of the line reveal that these performance differences can be substantial.

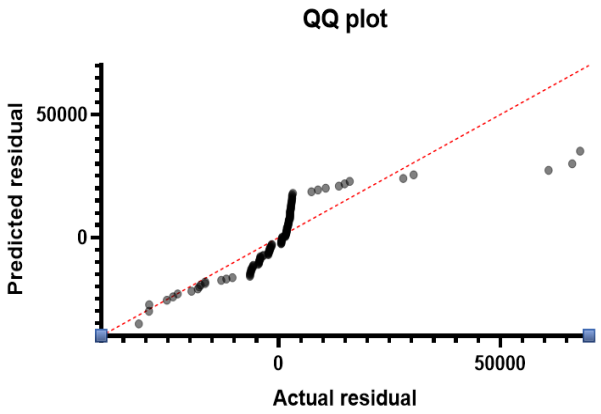

Fig. 10 QQ-plot distribution showing the distinct performance of GA-PSeOA compared to other algorithms for more complex instances

7. CONCLUSIONS

In conclusion, the innovative hybrid approach presented in this paper, which combines the strengths of Genetic Algorithms (GA) and Penguin Search Optimization (PSeOA), has proven to be highly effective in solving the multistage flow shop scheduling problem (FSSP). By integrating the genetic diversity and solution space exploration capabilities of GA with the rapid convergence and adaptability of PSeOA, this hybrid method significantly outperforms pure GA, PSeOA, and other state-of-the-art metaheuristics in comprehensive experimental evaluations.

The promising results of this study pave the way for further research in metaheuristics applied to combinatorial optimization, particularly in the context of FSSP, and emphasize the need to address the limitations of the hybrid approach while exploring avenues for improvement and extension to tackle other combinatorial optimization problems and challenges in the scheduling domain.

The successful application of the hybrid GA-PSeOA approach in the context of FSSP highlights its potential for enhancing efficiency and productivity in manufacturing systems. The implementation of this method can contribute to substantial advancements in manufacturing processes, leading to more streamlined and cost-effective operations.

Future research will focus on several key aspects to advance the current state of knowledge. Firstly, refining the performance of the hybrid GA-PSeOA approach for FSSP will be prioritized to improve its efficiency and impact. Additionally, the exploration of potential applications for this hybrid optimization technique in other optimization tasks will expand its scope and influence across various domains. Furthermore, the development of hybrid optimization algorithms that blend the strengths of GA-PSeOA with other optimization techniques has the potential to yield significant improvements in the overall efficiency and effectiveness of the method.

Acknowledgement: *The authors extend their appreciation to the Deputyship for Research & Innovation, Ministry of Education in Saudi Arabia, for funding this research work through the project no. (IFKSUOR3–340-2).*

REFERENCES

1. Peres, F., Castelli, M., 2021, *Combinatorial optimization problems and metaheuristics: Review, challenges, design, and development*, Applied Sciences, 11(14), 6449.
2. Arik, O. A., 2022, *Genetic Algorithm Application for Permutation Flow Shop Scheduling Problems*, Gazi University Journal of Science, 35(1), pp. 92–111.
3. Kennedy, J., Eberhart, R., 1995, *Particle Swarm Optimization*, Proceedings of ICNN'95-International Conference on Neural Networks, Perth, WA, Australia, 4, pp. 1942-1948.
4. Mzili, T., Riffi, M. E., Mzili, I., Dhiman, G., 2022, *A novel discrete Rat swarm optimization (DRSO) algorithm for solving the traveling salesman problem,* Decision Making: Applications in Management and Engineering, 5(2), pp. 287–299.
5. Blum, C., 2005, *Ant colony optimization: Introduction and recent trends*, Physics of Life Reviews, 2(4), pp. 353–373.
6. Shao, W., Shao, Z., Pi, D., 2021, *Multi-objective evolutionary algorithm based on multiple neighborhoods local search for multi-objective distributed hybrid flow shop scheduling problem*, Expert Systems with Applications, 183, 115453.
7. Han, W., Deng, Q., Gong, G., Zhang, L., Luo, Q., 2021, *Multi-objective evolutionary algorithms with heuristic decoding for hybrid flow shop scheduling problem with worker constraint*, Expert Systems with Applications, 168, 114282.
8. Amirteimoori, A., Mahdavi, I., Solimanpur, M., Ali, S. S., Tirkolaee, E. B., 2022, *A parallel hybrid PSO-GA algorithm for the flexible flow-shop scheduling with transportation*, Computers & Industrial Engineering, 173, 108672.
9. Qiao, Y., Wu, N., He, Y., Li, Z., Chen, T., 2022, *Adaptive genetic algorithm for two-stage hybrid flow-shop scheduling with sequence-independent setup time and no-interruption requirement*, Expert Systems with Applications, 208, 118068.
10. Vali, M., Salimifard, K., Gandomi, A. H., Chaussalet, T. J., 2022, *Application of job shop scheduling approach in green patient flow optimization using a hybrid swarm intelligence*, Computers & Industrial Engineering, 172, 108603.
11. Miyata, H. H., Nagano, M. S., 2022, *An iterated greedy algorithm for distributed blocking flow shop with setup times and maintenance operations to minimize makespan*, Computers & Industrial Engineering, 171, 108366.
12. Cui, H., Li, X., Gao, L., *An improved multi-population genetic algorithm with a greedy job insertion inter-factory neighborhood structure for distributed heterogeneous hybrid flow shop scheduling problem*, Expert Systems with Applications, 222, 119805.
13. Hou, Y., Wang, H., Fu, Y., Gao, K., Zhang, H., 2023, *Multi-Objective brain storm optimization for integrated scheduling of distributed flow shop and distribution with maximal processing quality and minimal total weighted earliness and tardiness*, Computers & Industrial Engineering, 179, 109217.

14. Mzili, T., Mzili, I., Riffi , M. E., *2023, Optimizing production scheduling with the Rat Swarm search algorithm: A novel approach to the flow shop problem for enhanced decision making*, Decision Making: Applications in Management and Engineering, 6(2), pp. 16–42.
15. Mzili, T., Mzili, I., Riffi, M. E., Pamucar, D., Kurdi, M., Ali, A. H., 2023, *Optimizing production scheduling with the spotted hyena algorithm: A novel approach to the flow shop problem*, Reports in Mechanical Engineering, 4(1), pp. 90–103.
16. Gheraibia, Y., Moussaoui, A.,Yin, P., Papadopoulos, Y., Maazouzi, S., 2019, *PeSOA: Penguins Search Optimisation Algorithm for Global Optimisation Problems*, The International Arab Journal of Information Technology, 16(3), pp. 49–57.
17. Chaudhry, I. A., Elbadawi, I. A., Usman, M., Chughtai, M. T., 2018, *Minimising total flowtime in a no-wait flow shop (NWFS) using genetic algorithms*, Ingenieria e Investigacion, 38(3), pp. 68–79.
18. Kurdi, M., 2021, *Application of Social Spider Optimization for Permutation Flow Shop Scheduling Problem*, Journal of Soft Computing and Artificial Intelligence, 2(2), pp. 85-97.
19. Shao, W., Pi, D., 2016, *A self-guided differential evolution with neighborhood search for permutation flow shop scheduling*, Expert Systems with Applications, 51, pp. 161–176.
20. Chakraborty, S., Saha, A.K., Sharma, S., Chakraborty, R., Debnath, S., 2023, *A hybrid whale optimization algorithm for global optimization*, J Ambient Intell Human Comput 14, pp. 431–467.
21. Kurdi, M., 2020, *A memetic algorithm with novel semi-constructive evolution operators for permutation flowshop scheduling problem*, Applied Soft Computing, 94, 106458.
22. Liang, Z., Zhong, P., Liu, M., Zhang, C., Zhang, Z., 2022, *A computational efficient optimization of flow shop scheduling problems*, Scientific Reports, 12(1), 845.
23. Gao, K. Z., Pan, Q. K., Li, J. Q., 2011, *Discrete harmony search algorithm for the no-wait flow shop scheduling problem with total flow time criterion*, International Journal of Advanced Manufacturing Technology, 56(5–8), pp. 683–692.
24. Taillard, E., 1993, *Benchmarks for basic scheduling problems*, European Journal of Operational Research, 64(2), pp. 278–285.