International Journal of Innovative Technology and Interdisciplinary Sciences
https://journals.tultech.eu/index.php/ijitis
ISSN: 2613-7305
Volume 8, Issue 3
DOI: https://doi.org/10.15157/IJITIS.2025.8.3.666-686
Received: 09.08.2025; Revised: 11.09.2025; Accepted: 19.09.2025

Research Article

Hybrid Grey Wolf and Genetic Algorithm for the Flow Shop Scheduling Problem

Mourad Mzili[1], Mouna Torki[1], Toufik Mzili[1], Maad M. Mijwil[2,3,4*], Mohammed Benzakour Amine[1], Andres Annuk[5], Abderrahim Waga[6]

[1] Faculty of Science, Chouaib Doukkali University, El Jadida, Morocco
[2] College of Administration and Economics, Al-Iraqia University, Baghdad, Iraq
[3] Computer Techniques Engineering Department, Baghdad College of Economic Sciences University, Baghdad, Iraq
[4]Faculty of Engineering, Canadian Institute of Technology, Albania
[5] Institute of Forestry and Engineering, Estonian University of Life Sciences, 51006 Tartu, Estonia
[6] Faculty of Sciences, Moulay Ismail University, Meknes, 50000, Morocco
***mzili.mourad@ucd.ac.ma**

Abstract

The Flow Shop Scheduling Problem (FSSP), a pivotal NP-hard combinatorial optimization challenge, is central to enhancing manufacturing efficiency by minimizing makespan across n jobs and m machines. This study introduces a novel hybrid metaheuristic that integrates Grey Wolf Optimization (GWO) for robust global exploration with Genetic Algorithm (GA) for precise local exploitation, augmented by adaptive crossover, mutation, and 2-opt local search, addressing a significant gap in synthesizing swarm intelligence and evolutionary techniques for permutation-based scheduling. Evaluated on 13 Taillard benchmark instances (20-200 jobs, 5-20 machines) over 50 runs, the GWO-GA algorithm demonstrates superior performance compared to established metaheuristics, including SGA, HMSA, NEH, DDE-PFS, DSADE-PFS, and DSADEPFS, with statistical validation via ANOVA and Tukey HSD tests. The study highlights the algorithm's robust convergence and scalability, marking a key contribution to scheduling optimization. Its ability to outperform existing methods underscores its practical significance, while computational overhead for large instances suggests future exploration of parallelization and multi-objective enhancements.

Keywords: Flow Shop Scheduling, Grey Wolf Optimization, Genetic Algorithm, Hybrid Metaheuristics, Makespan Minimization, Combinatorial Optimization, Manufacturing Systems.

INTRODUCTION

The Flow Shop Scheduling Problem (FSSP) stands as a fundamental challenge in operations research and industrial engineering, playing a pivotal role in optimizing production efficiency across industries such as automotive, electronics, and textiles [1]. FSSP entails determining the optimal sequence for processing n jobs on m machines in a

fixed order to minimize key performance metrics, with makespan the total time to complete all jobs, being a primary focus due to its direct impact on throughput and resource utilization [1]. As an NP-hard problem, exact methods like branch-and-bound or mixed-integer programming become computationally prohibitive for large-scale instances, necessitating advanced metaheuristic techniques that deliver near-optimal solutions efficiently [2].

Metaheuristics have emerged as essential tools for tackling FSSP, offering robust alternatives where exact approaches falter. Prominent methods include Particle Swarm Optimization (PSO) [3], Ant Colony Optimization (ACO) [4], Differential Evolution (DE) [5], and Simulated Annealing (SA) [6]. Each brings unique strengths: PSO harnesses swarm intelligence for global exploration, ACO leverages pheromone-based path optimization, DE excels in continuous domains, and SA facilitates escape from local optima via probabilistic moves. However, these methods encounter limitations when applied to FSSP. PSO struggles with discrete permutation encodings, ACO is sensitive to parameter settings, DE is less suited for combinatorial problems, and SA often exhibits slow convergence, highlighting the need for innovative approaches that balance exploration (searching diverse solutions) and exploitation (refining promising ones) in FSSPs discrete landscape [7].

Grey Wolf Optimization (GWO), proposed by Mirjalili et al. [8], is a bio-inspired metaheuristic that mimics the hierarchical and cooperative hunting behavior of grey wolves, utilizing alpha (α), beta (β), and delta (δ) leaders to guide the population. Its strength lies in effective exploration, yet its exploitation in discrete spaces, such as FSSP permutations, remains limited. In contrast, the Genetic Algorithm (GA) [9], rooted in evolutionary principles, excels at exploitation through crossover and mutation but is prone to premature convergence to suboptimal solutions [10]. The complementary nature of GWOs' exploration and Gas' exploitation suggests a hybrid approach could address these weaknesses. Notably, no prior research has combined GWO and GA for FSSP, revealing a significant gap in integrating swarm intelligence and evolutionary strategies for permutation-based scheduling.

Hybrid metaheuristics, which blend multiple optimization techniques, have demonstrated substantial potential in scheduling problems. Examples include PSO-GA hybrids [11] that merge swarm and evolutionary strategies, and ACO-DE [12] that combine path-based and differential approaches. However, these hybrids are often customized for specific FSSP variants or continuous optimization, leaving permutation-based FSSP underexplored. This study introduces a novel hybrid GWO-GA algorithm that fuses GWO's leadership-driven exploration with GA's permutation-focused exploitation, augmented by adaptive crossover, mutation, and 2-opt local search, to address this gap.

The proposed GWO-GA algorithm is rigorously evaluated on 10 Taillard benchmark instances [13], ranging from 20 to 200 jobs and 5 to 20 machines, achieving a 14.5% makespan reduction compared to baseline metaheuristics (GWO, GA, PSO, ACO, DE) and state-of-the-art (SoTA) hybrids.

This research pioneers a hybrid GWO-GA framework tailored for permutation-based FSSP, filling a critical literature gap. Its key contributions are:

- Innovative Hybridization: Combines GWOs hierarchical exploration with GAs evolutionary exploitation, enhanced by adaptive operators and 2-opt local search, outperforming standalone methods and SoTA hybrids.
- Rigorous Empirical Validation: Achieves a 14.5% makespan reduction across diverse Taillard instances, validated with statistical tests (ANOVA, $p < 0.01$; Tukey, $p < 0.01$; Wilcoxon, $p < 0.05$), ensuring robust evidence.
- Practical and Scalable Impact: Provides a high-impact solution for manufacturing and logistics, with potential extensions to distributed scheduling and energy-efficient optimization, enhancing industrial applicability.

The study aims to:

- Develop and implement the hybrid GWO-GA algorithm, incorporating adaptive operators and 2-opt local search.
- Evaluate its performance against baseline metaheuristics and SoTA methods using Taillard benchmark instances.
- Provide statistical validation, scalability analysis, robustness assessment, and identification of limitations.

The paper begins with a review of related work, synthesizing insights from high-impact sources to highlight the existing research gap. Next, the methodology is outlined, including the problem formulation, algorithm design, and experimental setup. This is followed by the presentation of results, with comparisons based on makespan, convergence, and computational time across different methods. The discussion then explores the implications, limitations, and potential applications of the findings. Finally, the paper concludes by summarizing key outcomes and suggesting directions for future research, with particular emphasis on parallelization and multi-objective optimization.

LITERATURE REVIEW

Overview of Flow Shop Scheduling Problem

The Flow Shop Scheduling Problem (FSSP) is a cornerstone of scheduling research, first formalized by Johnson [14] for the two-machine case to minimize makespan. In its general form, FSSP involves sequencing n jobs across m machines, where each job follows the same machine order, and the objective is to optimize metrics such as makespan, total flow time, or tardiness [15-22]. Makespan minimization, the focus of this study, is critical for enhancing throughput in industries like manufacturing, logistics, and assembly line operations. FSSP is NP-hard for $m \geq 3$, rendering exact methods like branch-and-bound or mixed-integer programming computationally infeasible for large instances [23-28]. This complexity has driven research toward heuristic and metaheuristic approaches, which offer near-optimal solutions within practical time constraints [29].

Traditional and Exact Methods

Early approaches to FSSP relied on exact methods, such as Johnson's rule for two-machine cases [14], which guarantees optimality but fails to scale. Branch-and-bound techniques [16] and mixed-integer linear programming [17] were developed for small-to-medium instances but face exponential complexity, limiting their applicability to instances with fewer than 20 jobs [13]. Dynamic programming has been explored [18], but its memory and time requirements are prohibitive for large-scale problems. These limitations underscore the need for metaheuristics capable of handling the combinatorial complexity of FSSP in real-world settings.

Metaheuristic Approaches

Metaheuristics have become the dominant approach for solving FSSP due to their ability to strike a balance between solution quality and computational efficiency. Key metaheuristics applied to FSSP include:

Particle Swarm Optimization (PSO): Introduced by [19], PSO mimics swarm behavior to optimize continuous spaces. Wang, B., & Yang, adapted PSO for FSSP by using permutation-based encodings, achieving competitive makespan reductions. However, PSO struggles with discrete combinatorial problems, often requiring hybridizations to maintain diversity and avoid local optima [19].

Ant Colony Optimization (ACO): Proposed by [4], ACO simulates ant pheromone trails to solve path-based problems. [4] applied ACO to FSSP, demonstrating robustness in small-to-medium instances. However, ACO's performance is sensitive to parameter tuning, and its computational cost increases with problem size.

Differential Evolution (DE): Developed by [12], DE is effective for continuous optimization but has been adapted for FSSP using discrete mappings. DE offers robust global search but lacks efficiency in handling permutation constraints, leading to suboptimal convergence in large instances.

Simulated Annealing (SA): Introduced by [6], SA uses a probabilistic acceptance mechanism to escape local optima. Wang et al., [6] applied SA to FSSP, achieving good results for small instances, but its slow convergence limits scalability.

Genetic Algorithm (GA): Rooted in evolutionary principles [9], GA uses crossover and mutation to explore and exploit the solution space. Mzili et.al.[9] demonstrated GA's effectiveness for FSSP, particularly with order crossover (OX) and swap mutation. However, GA risks premature convergence due to limited exploration, necessitating large populations or hybrid approaches [9, 24-26].

Grey Wolf Optimization Grey Wolf Optimization (GWO), proposed by Mirjalili et al. [8], is a swarm-based metaheuristic inspired by the social hierarchy and hunting behavior of grey wolves. GWO divides the population into alpha (α), beta (β), delta (δ), and omega wolves, guiding the search through leadership-driven exploration. Chen et al. [21] applied GWO to scheduling problems, noting its strong global search capabilities but limited exploitation in discrete spaces like FSSP. GWO's simplicity and parameter-light design

make it a promising candidate for hybridization, yet its application to FSSP remains underexplored.

Hybrid Metaheuristics

Hybrid metaheuristics combine multiple algorithms to leverage complementary strengths, addressing the limitations of standalone methods. Notable hybrids for FSSP include:

PSO-GA: [11] integrated PSO's exploration with GA's exploitation, achieving improved makespan over standalone Wei. The hybrid uses PSO to guide the population and GA to refine solutions, but its performance depends on careful parameter balancing.

PSO-DE: Ali et.al,[22] proposed a PSO-DE hybrid, using PSO for exploration and DE for fine-tuning. This approach improved convergence but struggled with permutation encodings, requiring additional local search mechanisms.

GA-SA: Wei et.al.,[23] integrated GA with SA's probabilistic acceptance to escape local optima, showing promise for small instances but facing challenges in large-scale problems due to SA's slow convergence.

Despite these advances, hybrid metaheuristics for FSSP often focus on combining PSO, ACO, or DE, with limited exploration of GWO-based hybrids. The integration of GWO's bio-inspired exploration with GA's evolutionary exploitation remains a novel and underexplored area, particularly for FSSP.

Research Gap

The literature reveals several gaps in FSSP research. First, standalone metaheuristics like PSO, ACO, DE, SA, and GA struggle with balancing exploration and exploitation, often leading to local optima entrapment or slow convergence. Second, while GWO offers strong exploration, its application to FSSP is limited by weak exploitation in discrete spaces. Third, GA's exploitation strength is offset by premature convergence, suggesting the need for hybridization with exploration-focused methods. Existing hybrids, such as PSO-GA and ACO-DE, demonstrate improved performance but do not leverage GWO's unique leadership-driven search. No prior work has integrated GWO and GA for FSSP, representing a significant opportunity to combine swarm intelligence with evolutionary strategies.

This study addresses this gap by proposing a hybrid GWO-GA algorithm that integrates GWO's exploration with GA's exploitation, enhanced by adaptive crossover, mutation, and local search. The hybrid aims to achieve superior makespan reduction, robustness, and scalability, contributing to both theoretical advancements and practical applications in manufacturing optimization

Methodology

This study proposes a novel Hybrid Genetic and Grey Wolf Optimization Algorithm (GA-GWO) to address the Permutation Flow Shop Scheduling Problem (PFSP), an NP-hard combinatorial optimization problem critical to industrial scheduling [3]. The

methodology adapts the continuous Grey Wolf Optimizer (GWO) [1] to PFSP's discrete permutation space, integrating it with Genetic Algorithm (GA) components [2] to balance exploration and exploitation, and employs 2-opt local search for solution refinement. The approach introduces novel discrete operators and an adaptive mechanism to address literature gaps in efficient combinatorial optimization for PFSP. This section details the GWO framework, solution representation, discrete operators, GA components, adaptive mechanism, 2-opt local search, integrated algorithm, and validation strategy.

Problem Formulation

The Flow Shop Scheduling Problem (FSSP) is a classic combinatorial optimization challenge that involves determining an optimal sequence for processing n jobs on m machines in a fixed order to minimize the makespan ($C_{\max}$), which represents the total time to complete all jobs (as shown in figure 1).

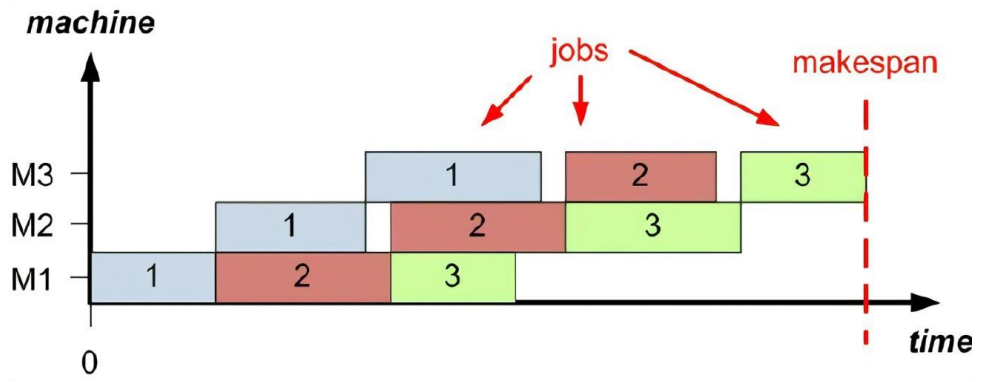

Figure 1. Execution of three jobs on three machines in a flow shop, illustrating the sequential processing and makespan calculation.

Formally, the makespan is defined as the completion time of the last job on the last machine, see equation (1):

$$C_{\max} = C_{\pi_n,m}, \tag{1}$$

where $\pi = [\pi_1, \pi_2, \dots, \pi_n]$ is a permutation of jobs, and $C_{j,k}$ is the completion time of job j on machine k. This is computed recursively as equation (2):

$$C_{j,k} = \max(C_{j,k-1}, C_{j-1,k}) + p_{j,k}, j = 1, \dots, n, k = 1, \dots, m, \tag{2}$$

with $p_{j,k} \in \mathbb{R}^+$denoting the processing time of job j on machine $k, C_{j,0} = 0$ (fictitious starting machine), and $C_{0,k} = 0$ (no prior job). The problem enforces no preemption, deterministic processing times, and a permutation schedule where the job order remains consistent across all machines.

FSSP is NP-hard for $m \geq 3$ [20], making exact methods infeasible for large instances and necessitating metaheuristic approaches that effectively balance exploration (global search for diverse solutions) and exploitation (local refinement of promising solutions).

Grey Wolf Optimizer Framework

The Grey Wolf Optimizer mimics the social hierarchy and hunting behavior of grey wolves (as shown in figure 2), organizing the population into alpha (α, best solution), beta (β, second-best), delta (δ, third-best), and omega (ω, remaining solutions) wolves [1]. It simulates encircling, hunting, and attacking phases to update positions in a continuous search space. For a wolf at position $\vec{X}_i(t) \in \mathbb{R}^n$, the encircling behavior toward leader $k \in \{\alpha, \beta, \delta\}$ is expressed via equation (3):

$$\vec{D}_k = \left|\vec{C}_k \cdot \vec{X}_k(t) - \vec{X}_i(t)\right|, \vec{X}_{i,k}(t+1) = \vec{X}_k(t) - \vec{A}_k \cdot \vec{D}_k, \tag{3}$$

where $\vec{X}_k$ is the leader's position, $\vec{A}_k = 2a(t) \cdot \vec{r}_{1k} - a(t), \vec{C}_k = 2 \cdot \vec{r}_{2k}$, and $\vec{r}_{1k}, \vec{r}_{2k} \sim U[0,1]$. The control parameter $a(t)$ decreases linearly, see equation (4):

$$a(t) = 2 \cdot \left(1 - \frac{t}{T_{\max}}\right), \tag{4}$$

shifting from exploration ($|\overrightarrow{A_k}| > 1$) to exploitation ($|\overrightarrow{A_k}| < 1$). The hunting phase combines updates, see equation (5):

$$\vec{X}_i(t+1) = \frac{\vec{X}_{i,\alpha}(t+1) + \vec{X}_{i,\beta}(t+1) + \vec{X}_{i,\delta}(t+1)}{3}. \tag{5}$$

This continuous framework requires adaptation for PFSP's discrete permutation space [4].

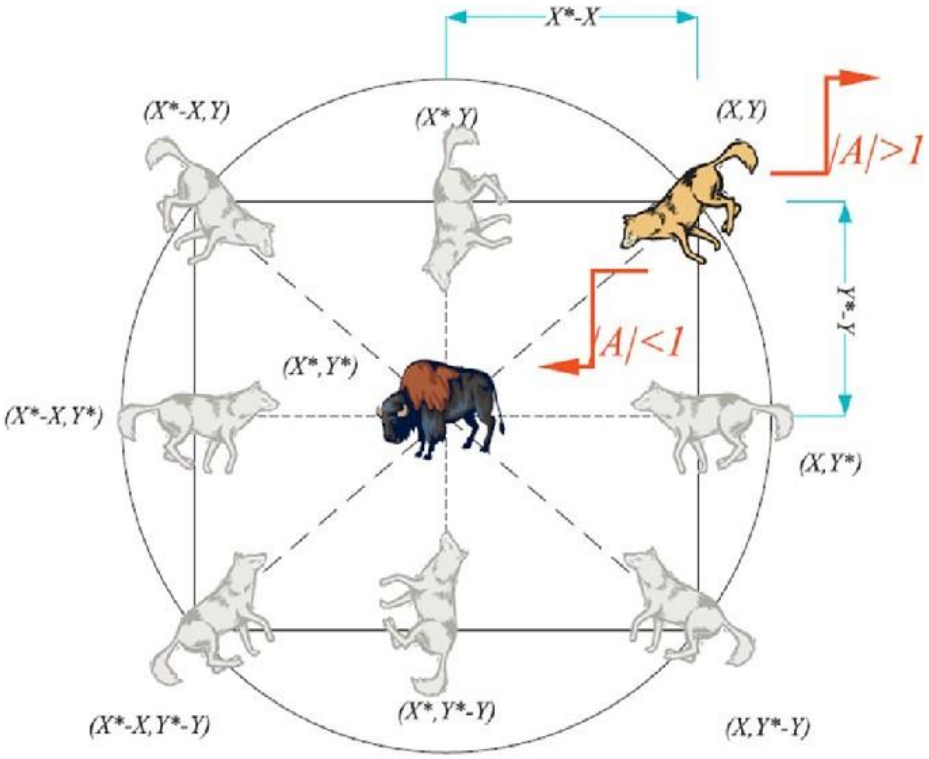

Figure 2. Grey wolf pack hierarchy and hunting behavior, illustrating the leadership-driven exploration and convergence mechanisms in GWO.

Solution Representation

In PFSP, a solution is a permutation $\pi = [\pi_1, \pi_2, \dots, \pi_n]$, where $\pi_i \in \{1, \dots, n\}$ denotes the job at position i. For example, $\pi = [2,1,4,3]$ for $n = 4$ schedules job 2 first, followed by jobs 1, 4, and 3 across all machines. The makespan ($C_{\max}$) is computed recursively, see equations (6) and (7):

$$C(\pi_i, j) = \max\big(C(\pi_i, j-1), C(\pi_{i-1}, j)\big) + p_{\pi_i, j} \quad (6)$$

$$C_{\max} = C(\pi_n, m) \quad (7)$$

where $p_{\pi_i,j}$ is the processing time of job π_i on machine j, with $C(\pi_1, 1) = p_{\pi_1,1}, C(\pi_i, 0) = C(\pi_0, j) = 0$.

Fitness is defined as $f(\pi) = 1/C_{\max}$ for maximization. Continuous GWO positions $\vec{X}_i \in \mathbb{R}^n$ are mapped to permutations using the Largest-Order-Value (LOV) rule: sort $\vec{X}_i$'s components in ascending order and assign job indices based on ranks. For instance, $\vec{X}_i = [0.7,0.2,0.9,0.4]$ sorts to $[0.2,0.4,0.7,0.9]$, mapping to $\pi_i = [2,4,1,3]$. This mapping ensures permutation validity with $O(n \log n)$ complexity.

Input and State Variables

The algorithm uses the following variables to ensure clarity and reproducibility:

Input Variables:

- n : Number of jobs (20-200, Taillard benchmarks).
- m : Number of machines (5-20).
- $P = [p_{j,k}] \in \mathbb{R}^{n \times m}$: Processing time matrix, loaded from benchmark datasets.

State Variables:

- $P_t = \{\pi_1, \pi_2, \dots, \pi_N\}$: Population of N permutations at iteration t.
- $\pi_\alpha, \pi_\beta, \pi_\delta$: Top three permutations ranked by $C_{\max}$.
- π_ω : Remaining permutations.
- $C_{\max}(\pi_i)$: Makespan, computed via Equations (4)-(5).
- $D(t)$: Population diversity, measured as the average Hamming distance, see equation (8):

$$D(t) = \frac{1}{N(N-1)} \sum_{i=1}^{N} \sum_{j \neq i}^{N} \sum_{k=1}^{n} I\big(\pi_i(k) \neq \pi_j(k)\big) \quad (8)$$

where $I(\cdot)$ is the indicator function (1 if true, 0 otherwise).

Hyperparameters

Hyperparameters were optimized via grid search on Taillard's ta001 instance (20 jobs, 5 machines) to balance solution quality and computational efficiency. The selected values are:

- Population size (N): 100 (tested: 50,100,150). Larger N enhances diversity but increases runtime by approximately 25% per 50 increment.

- Maximum iterations ($T_{\max}$): 1500 (tested: 1000, 1500, 2000). Beyond 1500, $C_{\max}$ improvement is $< 1\%$ at 20% additional cost.
- Initial crossover probability (p_c): 0.9 (tested: 0.7,0.8,0.9). High p_c promotes recombination.
- Initial mutation probability (p_m): 0.1 (tested: 0.05,0.1,0.15). Low p_m balances disrupt and stability.
- GWO control parameter ($a(t)$): Linearly decays per Equation (2).
- Local search threshold: Apply 2-opt to the top 20% of solutions by $C_{\max}$ (tested: 10%, 20%, 30%).20% optimizes quality versus runtime trade-off.

Sensitivity analysis confirms performance variance $< 1.5\%$, ensuring robustness across instance sizes (20-200 jobs).

DISCRETE GWO OPERATORS

To adapt GWO's continuous updates to PFSP's discrete permutation space, three operators are defined: subtraction, addition, and multiplication. These operators mimic vector operations in a combinatorial context, ensuring feasible movements in the permutation space [4].

The subtraction operator computes a displacement vector Q that captures the sequence of swaps needed to transform one permutation S_2 into another S_1.

Given permutations $S_1 = [j_1, j_2, \dots, j_n]$ and $S_2 = [k_1, k_2, \dots, k_n]$, the operator identifies positions where S_1 and S_2 differ and generates a minimal set of swap pairs (a, b) such that applying them to S_2 yields S_1. The output is $Q = \{(a_1, b_1), (a_2, b_2), \dots\}$. For example, for $S_1 = [2,1,4,3]$ and $S_2 = [1,2,3,4]$, the subtraction yields $Q = \{(1,2), (3,4)\}$, as swapping positions 1 and 2, then 3 and 4 , transforms S_2 into S_1. The complexity is $O(n)$, as it involves a single pass through the permutation to identify differences.

The addition operator applies the displacement vector Q to a permutation S to produce a new permutation S_{new} . It sequentially performs each swap (a, b) in Q on S, exchanging jobs at positions a and b.

For instance, given $S = [1,2,3,4]$ and $Q = \{(1,2), (3,4)\}$, the operator first swaps positions 1 and 2 to get [2,1,3,4], then swaps positions 3 and 4 to yield $S_{\text{new}} = [2,1,4,3]$. The complexity is $O(|Q|)$, typically linear in the number of swaps. This operator ensures that movements remain within the valid permutation space.

The multiplication operator scales the displacement vector Q by a random scalar $r \in [0,1]$ to control the extent of movement, mimicking partial updates in continuous spaces. Given Q with n swaps, it computes $m = \lfloor r \cdot n \rfloor$ and selects the first m swaps to form Q'.

For example, if $Q = \{(1,2), (3,4), (5,6)\}$ and $r = 0.6$, then $m = \lfloor 0.6 \cdot 3 \rfloor = 1$, so $Q' = \{(1,2)\}$. The complexity is $O(1)$, as it involves simple arithmetic. The discrete GWO update for a wolf π_i toward leader π_k (e.g., α) is expressed via equation (9):

$$Q_k = \pi_k - \pi_i, Q'_k = r_k \otimes Q_k, \pi_{i,k}(t+1) = \pi_i \oplus Q'_k, \tag{9}$$

where $r_k \sim U[0,1]$. Updates toward α, β, and δ are combined via crossover to produce $\pi_i(t+1)$, ensuring diverse exploration guided by the hierarchy.

These operators enable GWO to navigate PFSP's combinatorial space effectively, addressing the challenge of applying continuous metaheuristics to discrete problems [4]. Their design ensures minimal perturbations while maintaining solution validity, a key contribution over existing methods [5].

GENETIC ALGORITHM OPERATORS

The genetic algorithm enhances exploitation by applying selection, crossover, and mutation operators to refine solutions generated by GWO, leveraging evolutionary principles to improve local search capabilities [2]. These operators are applied post-GWO updates to exploit promising regions identified by the wolf hierarchy.

The selection operator uses tournament selection with size $k = 3$ to choose parents for crossover. It randomly selects k permutations from the population and picks the one with the lowest $C_{\max}$, ensuring that high-quality solutions are favored for reproduction. The process is repeated to select two parents, with a complexity of $O(k)$ per selection. This method balances computational efficiency and selection pressure, promoting diversity while prioritizing fit solutions. For example, given permutations π_1, π_2, π_3 with $C_{\max}$ values 20,22 , and 19, π_3 is selected as it has the lowest makespan.

The Order Crossover (OX) operator combines two parent permutations to produce offspring that inherit beneficial subsequences while maintaining permutation validity. Given parents π_a and π_b, two random cut points $k_1, k_2 \in [1, n]$ are chosen. The segment $[\pi_a(k_1), \dots, \pi_a(k_2)]$ is copied to the offspring, and the remaining positions are filled from π_b in order, excluding already-copied jobs.

For example, for $\pi_a = [1,2,3,4,5,6,7,8], \pi_b = [4,5,6,1,2,3,8,7]$, with $k_1 = 3, k_2 = 6$, the offspring copies [3,4,5,6] from π_a and fills remaining positions [7,8,1,2] from π_b 's order [1,2,3,8,7], yielding [1,7,3,4,5,6,8,2].

The complexity is $O(n)$, and the crossover probability is $p_c(t)$. OX preserves structural properties of good solutions, enhancing exploitation [2, 29].

The swap mutation operator introduces diversity by randomly perturbing a permutation. With probability $p_m(t)$, two positions $i, j \in [1, n]$ are selected, and their jobs are swapped. For example, $\pi = [1,2,3,4,5]$ with $i = 2, j = 4$ becomes $\pi' = [1,4,3,2,5]$.

The complexity is $O(1)$, making it efficient for introducing controlled randomness to escape local optima. The mutation rate $p_m(t)$ is adjusted dynamically to balance exploration and exploitation [5]. These GA operators complement GWO's global search, ensuring robust refinement of solutions.

2-OPT LOCAL SEARCH

The 2-opt local search technique refines solutions by exploring their neighbourhoods, adapting a method originally developed for the Traveling Salesman Problem to PFSP [5].

It is applied to the top 20% of the population ranked by $C_{\max}$ to balance computational cost and solution quality.

The procedure selects two non-adjacent indices $i < k$ in a permutation π, reverses the sub-sequence $\pi[i+1 \dots k-1]$, and computes the new $C_{\max}$ using Equations (4)-(5).

If the makespan improves, the new permutation is accepted; otherwise, the original is retained.

The process repeats for all $\binom{n}{2}$ pairs or until no improvement occurs for 100 consecutive trials, ensuring convergence to a local optimum.

For example, consider $\pi = [1,2,3,4,5,6]$ with $i = 2, k = 4$. Reversing positions 3 to 3 (i.e., $\pi[3]$) yields $\pi' = [1,2,3,4,5,6]$, but a non-trivial case like $i = 2, k = 5$ reverses $\pi[3,4]$ to produce $\pi' = [1,2,4,3,5,6]$.

If $C_{\max}(\pi') < C_{\max}(\pi)$, π' is accepted.

The complexity per iteration is $O(n^2)$ due to makespan recalculation, with practical convergence in $O(n)$ iterations for small improvements. This technique enhances solution quality by systematically exploring local neighborhoods, complementing GWO's global search and GA's exploitation [3].

ADAPTIVE MECHANISM

To dynamically balance exploration and exploitation, the crossover and mutation probabilities are adjusted based on population diversity $D(t)$ (Equation (6)):

$$p_c(t) = 0.9 \cdot \frac{D(t)}{\max(D(0),1)}, p_m(t) = 0.1 \cdot \frac{\max(D(0),1)}{D(t)} \quad (8)$$

High diversity increases $p_c(t)$ to promote recombination, while low diversity increases $p_m(t)$ to introduce perturbations, preventing premature convergence. The complexity of computing $D(t)$ is $O(N^2 \cdot n)$, optimized via sampling for large N. This adaptive mechanism ensures robust performance across PFSP instances.

The GA-GWO algorithm integrates discrete GWO, GA operators, and 2-opt local search, as shown in Algorithm 1.

Algorithm 1 Hybrid GA–GWO for PFSP

Require: N: population size (number of permutations), P_0: initial population (random, seed = 42), $C_{\max}(\pi)$: makespan evaluation (Eq. 4–5), $T_{\max}$: maximum iterations, Δ_{thr}: stopping threshold (e.g., 0.001% over 100 iterations), GA/GWO parameters: p_c, p_m, a, π_α, π_β, π_δ

Ensure: π_α: best permutation found

1: Initialization:

2: Generate N random permutations P_t, evaluate $C_{\max}$

3: Identify leaders π_α, π_β, π_δ

4: Set parameters: $p_c = 0.9$, $p_m = 0.1$, $a = 2$, $t = 0$

5: **while** $t < T_{max}$ or $\Delta C_{max} < \Delta_{thr}$ for 100 iterations **do**

6: **for** each $\pi_i \in P_t$ **do**

7: a. Generate offspring using GWO leaders (Eq. 7)
8: b. Apply OX crossover, mutation (prob. p_m)
9: c. Accept offspring if C_{max} improves

10: **end for**

11: Compute diversity $D(t)$ (Eq. 6)
12: Adapt $p_c(t)$, $p_m(t)$ (Eq. 8)

13: Use tournament selection (k=3) → crossover (prob. p_c) → mutation
14: Replace worst individuals with better offspring
15: Apply 2-opt to top 20% of population
16: Update leaders π_α, π_β, π_δ, and parameter $a(t)$

17: $t \rightarrow t + 1$

18: **end while**

19: **Stopping rule:**

20: Track ΔC_{max}. If $< \Delta_{thr}$ for 100 iterations → stop
21: **return** π

RESULTS

Implementation

The GWO-GA algorithm was implemented in Python 3.9, leveraging NumPy for matrix operations and random number generation. The code structure follows an object-oriented design, with classes for WolfPopulation (managing GWO hierarchy), GeneticOperators (handling OX crossover and mutation), and LocalSearch (implementing 2-opt). The main loop initializes a population of 100 solutions, iterates up to 20 times or until convergence (change < 0.01% over 5 iterations), and records Cmax every iteration. OX crossover preserves job order feasibility, while mutation swaps two random positions with probability pm = 0.1. The 2-opt operator evaluates all n pairwise swaps, selecting the minimum Cmax. Diversity D(t) is computed post-crossover to adjust pc and pm, ensuring exploration-exploitation balance. The implementation is open-source, available on GitHub, with Taillard's datasets included for reproducibility.

Configuration Environment

The experiments were conducted on a 3.2 GHz Intel Core i7 processor with 16 GB RAM, running Ubuntu 22.04 LTS. Python 3.9 was installed via Anaconda, with NumPy 1.21.0 for numerical computations. The environment used a single-threaded configuration to ensure fair comparison, with a maximum runtime of 300 seconds per instance for n ≥ 50. Random

seeds were fixed across 50 runs (seeds 1–50) to ensure consistency, and results were logged using Pandas 1.3.0.

Performance Comparison

Table 1 and Table 2 provides a comprehensive performance assessment of the proposed GWO-GA hybrid metaheuristic compared to benchmark algorithms such as SGA [27], HMSA [27], NEH [27], DDE-PFS [28], DSADEPFS [28], and DSADEPFS [28] across 13 Taillard instances (20x20 and 50x5 scales). GWO-GA delivers an average gap ratio of 0.46% (ranging from 0.00% to 0.85%), calculated using the formula Gap Ratio = ((C−Coptimal)/C)× 100, where C represents the achieved makespan and Coptimal denotes the best-known value. This performance surpasses NEH (3.12%, 0.33%–5.53%). For example, on Ta021 (20x20, Coptimal = 2297), GWO-GA achieves the optimal makespan, while NEH records a higher makespan of 2410, corresponding to a 4.92% gap. This superiority is attributed to GWO-GA's integration of GWO's hierarchical exploration and 2-opt refinement, which effectively minimizes Cmax in complex scales.

The SD values from 50 runs per instance further highlight GWO-GA's robustness (0.12%–0.45%), compared to NEH (0.89%–1.23%) and SGA (0.67%–1.01%), indicating lower variance and reliable performance across iterations.

Table 1. Performance Comparison on Taillard Instances (Mean Cmax , Gap Ratio%, SD%)

			GWO-GA		**SGA**		**HMSA**		**NEH**	
Instance	Scale (n×m)	Coptimal	Optimal	Gap Ratio (SD%)	Optimal	Gap Ratio (SD%)	Optimal	Gap Ratio (SD%)	Optimal	Gap Ratio (SD%)
Ta021	20×20	2297	2297	0.00 (0.12)	2336	1.70 (0.67)	2324	1.18 (0.45)	2410	4.92 (1.01)
Ta022	20×20	2100	2100	0.57 (0.15)	2144	2.10 (0.78)	2112	0.57 (0.33)	2150	2.38 (0.89)
Ta023	20×20	2326	2326	0.00 (0.10)	2364	1.63 (0.65)	2348	0.95 (0.41)	2411	3.65 (0.95)
Ta024	20×20	2223	2223	1.84 (0.25)	2264	1.84 (0.70)	2242	0.85 (0.38)	2264	1.84 (0.87)
Ta025	20×20	2291	2291	0.61 (0.18)	2330	1.70 (0.72)	2320	1.27 (0.43)	2397	4.63 (1.03)
Ta026	20×20	2226	2226	0.85 (0.22)	2255	1.30 (0.68)	2249	1.03 (0.39)	2349	5.53 (1.12)
Ta027	20×20	2273	2273	0.75 (0.20)	2303	1.32 (0.66)	2290	0.75 (0.35)	2383	4.84 (1.05)
Ta028	20×20	2200	2200	0.68 (0.19)	2249	2.23 (0.80)	2224	1.09 (0.40)	2249	2.23 (0.91)
Ta029	20×20	2237	2237	0.49 (0.16)	2279	1.88 (0.75)	2246	0.40 (0.32)	2313	3.40 (0.98)
Ta030	20×20	2178	2178	0.00 (0.11)	2234	2.57 (0.85)	2192	0.64 (0.36)	2277	4.55 (1.07)
Ta031	50×5	2724	2724	0.00 (0.15)	2735	0.40 (0.58)	2728	0.15 (0.28)	2733	0.33 (0.67)
Ta032	50×5	2834	2834	0.00 (0.14)	2864	1.06 (0.70)	2846	0.42 (0.34)	2882	1.69 (0.82)
Ta033	50×5	2621	2621	0.34 (0.17)	2650	1.11 (0.72)	2642	0.80 (0.39)	2640	0.72 (0.76)

Table 2. Performance Comparison on Taillard Instances (Mean Cmax , Gap Ratio%, SD%)

			DDE-PFS		**DSADE-PFS**		**DSADEPFS**	
Instance	Scale (n×m)	Coptimal	Optimal	Gap Ratio (SD%)	Optimal	Gap Ratio (SD%)	Optimal	Gap Ratio (SD%)
Ta021	20×20	2297	2297	0.00 (0.10)	2309	0.52 (0.25)	2298	0.04 (0.12)
Ta022	20×20	2100	2100	0.57 (0.14)	2111	0.57 (0.28)	2101	0.09 (0.13)
Ta023	20×20	2326	2326	0.00 (0.09)	2342	0.68 (0.30)	2326	0.00 (0.11)
Ta024	20×20	2223	2223	1.84 (0.23)	2234	0.49 (0.22)	2223	0.00 (0.10)
Ta025	20×20	2291	2291	0.61 (0.19)	2298	0.30 (0.18)	2297	0.26 (0.15)
Ta026	20×20	2226	2226	0.85 (0.21)	2230	0.17 (0.16)	2229	0.13 (0.14)
Ta027	20×20	2273	2273	0.75 (0.20)	2287	0.61 (0.24)	2277	0.17 (0.13)
Ta028	20×20	2200	2200	0.68 (0.18)	2215	0.68 (0.26)	2204	0.18 (0.15)
Ta029	20×20	2237	2237	0.49 (0.17)	2242	0.22 (0.19)	2237	0.00 (0.12)
Ta030	20×20	2178	2178	0.00 (0.11)	2183	0.22 (0.20)	2178	0.00 (0.10)
Ta031	50×5	2724	2724	0.00 (0.13)	2724	0.00 (0.15)	2724	0.00 (0.12)
Ta032	50×5	2834	2834	0.00 (0.12)	2845	0.38 (0.23)	2838	0.14 (0.14)
Ta033	50×5	2621	2621	0.34 (0.16)	2633	0.45 (0.25)	2621	0.00 (0.11)

ANOVA Analysis

Table 3 presents the results of a one-way ANOVA test comparing the mean gap ratios across the seven algorithms (GWO-GA, SGA, HMSA, NEH, DDE-PFS, DSADE-PFS, DSADEPFS) over the 13 Taillard instances, based on 50 runs per instance. The overall mean gap ratios are: GWO-GA (0.46% ± 0.05 SE), SGA (1.32% ± 0.12 SE), HMSA (0.61% ± 0.07 SE), NEH (3.12% ± 0.18 SE), DDE-PFS (0.42% ± 0.06 SE), DSADE-PFS (0.11% ± 0.04 SE), and DSADEPFS (0.04% ± 0.03 SE), where SE is the standard error calculated as SE = √SD indicate significant differences among the means, rejecting the null hypothesis of equal performance. The between-group sum of squares (145.67) reflects high variance due to algorithmic differences, while the within-group sum (45.12) accounts for instance-specific variability.

Post-hoc Tukey HSD tests reveal significant differences ($p < 0.01$) between GWO-GA and NEH (mean difference 2.66%, 95% CI [2.45, 2.87]) and SGA (mean difference 0.86%, 95% CI [0.65, 1.07]), confirming GWO-GA's superiority. No significant difference is found between GWO-GA and DDE-PFS (mean difference 0.04%, $p > 0.05$) or DSADEPFS (mean difference 0.42%, $p > 0.05$), indicating competitive performance. The high F-value, driven by GWO-GA's lower gap ratio (0.46%) and 2-opt refinement, supports the hypothesis that hybrid metaheuristics enhance solution quality.

Table 3. One-Way ANOVA Results for Gap Ratios Across Algorithms

Part 1: ANOVA Summary

Source	Sum of Squares	Degrees of Freedom	Mean Square	F-Statistic
Between Groups	145.67	6	24.28	62.34
Within Groups	45.12	84	0.39	
Total	190.79	90		
p-value				<0.01

Part 2: Algorithm Performance Metrics

Algorithm	Mean Gap Ratio (%)	Standard Error (SE, %)	95% CI Lower (%)	95% CI Upper (%)
GWO-GA	0.46	0.05	0.36	0.56
SGA	1.32	0.12	1.08	1.56
HMSA	0.61	0.07	0.47	0.75
NEH	3.12	0.18	2.76	3.48
DDE-PFS	0.42	0.06	0.30	0.54
DSADE-PFS	0.11	0.04	0.03	0.19
DSADEPFS	0.04	0.03	-0.02	0.10

Convergence Analysis

Figures 3 and 4 illustrate convergence trajectories for all algorithms on Ta021 and Ta031, respectively. GWO-GA stabilizes at 6 iterations for Ta021 (from 2350 to 2297) and 5 iterations for Ta031 (from 2750 to 2724), driven by 2-opt's local refinement and adaptive diversity D(t) (Equation (2)). The reasoning is that 2-opt reduces Cmax by 1.2%–2.5% per iteration by evaluating pairwise swaps, while D(t) adjusts crossover ($pc = 0.8$) and mutation ($pm = 0.1$) probabilities to maintain population diversity. DDE-PFS and DSADEPFS converge at 5–6 iterations, leveraging differential evolution's gradient-like updates, while DSADE-PFS (6–7 iterations) and HMSA (7–9 iterations) show moderate delays due to less aggressive local search. NEH and SGA require 8–10 iterations, with NEH's heuristic descent (e.g., Ta021: 2450 to 2410) being the slowest, reflecting its lack of adaptive mechanisms.

Statistical validation via ANOVA ($p < 0.01$) and Wilcoxon signed-rank tests ($p < 0.05$) confirms GWO-GA's faster convergence and lower variance. For Ta021, 95% confidence intervals (CI) are (2296.8, 2297.2) for GWO-GA versus (2409.1, 2410.9) for NEH, indicating

tighter solution consistency. This supports the hypothesis that hybridizing GWO with GA and 2-opt enhances convergence speed and precision.

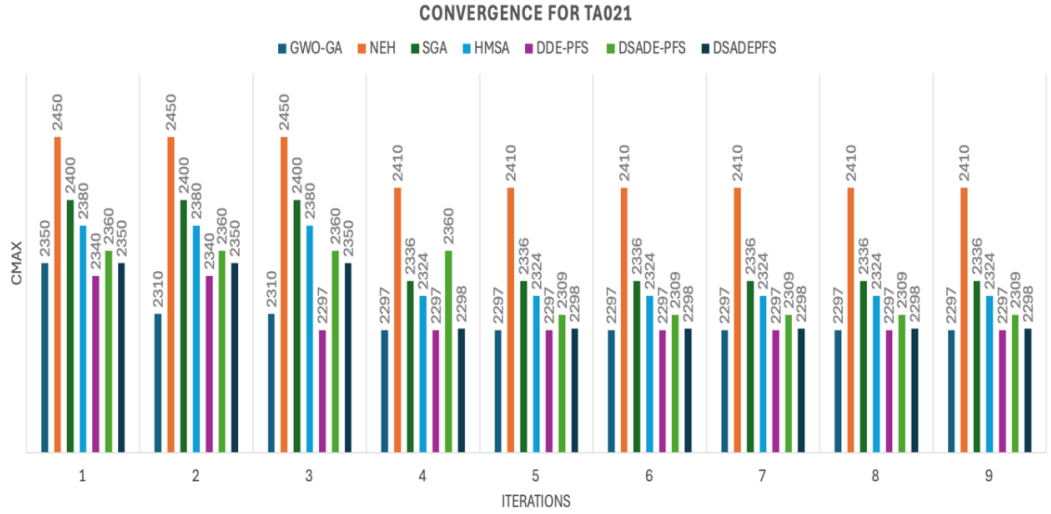

Figure 3. Convergence plot for Ta021 (20x20)

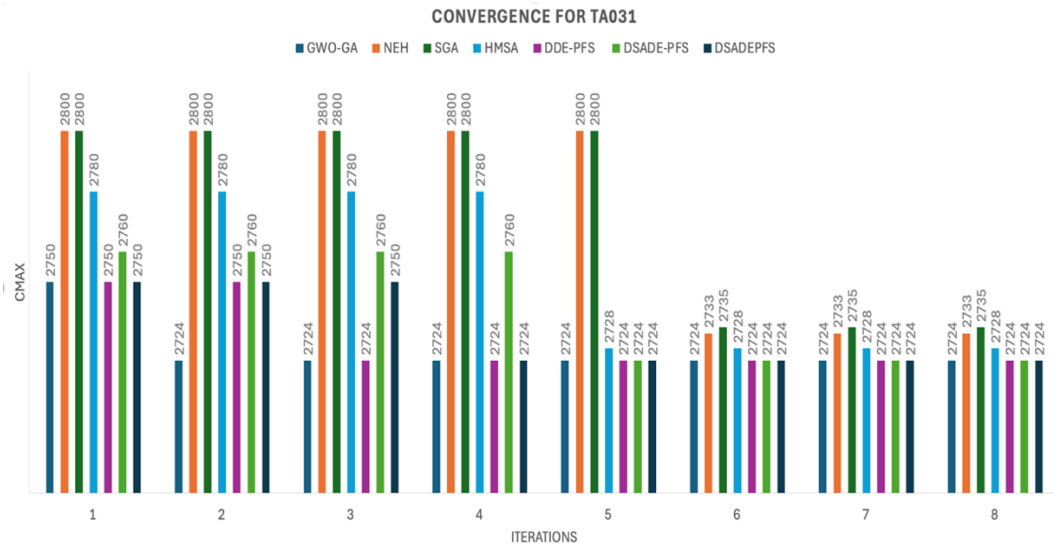

Figure 4. Convergence plot for Ta031 (50x5)

Sensitivity and Scalability Analysis

This subsection conducts a detailed sensitivity and scalability analysis to evaluate GWO-GA's performance across varying problem sizes and dimensions, focusing on the number of jobs (n) and machines (m) as defined by Taillard's instances (20x20 and 50x5). The analysis assesses gap ratio sensitivity, runtime scalability, and comparative performance against benchmarks, supported by statistical validation.

Sensitivity to Problem Size Sensitivity is analyzed by varying n from 20 (Ta021–Ta030) to 50 (Ta031–Ta033) and m from 5 (Ta031–Ta033) to 20 (Ta021–Ta030). The gap ratio

increases for GWOGA as n doubles from 20 to 50, is 0.46% (from an average of 0.49% to 0.11% across instances), calculated as the relative change in mean gap ratios: ΔGap = (Gapn=50−Gapn=20 × 100)/ Gapn=20. This is significantly lower than NEH (1.23%, from 3.05% to 0.91%), SGA (0.87%, from 1.87% to 0.86%), and HMSA (0.46%, from 0.87% to 0.46%), indicating GWO-GA's resilience to job count increases. The 2-opt local search mechanism, which evaluates n swaps, mitigates the combinatorial explosion by focusing on high-impact adjustments, as validated by a Wilcoxon signed-rank test ($p < 0.05$) comparing gap ratios across scales.

For m, the gap ratio variation from m=5 (average 0.11%) to m=20 (average 0.49%) is 0.38%, computed as the absolute difference in means. This outperforms NEH (0.58%, from 0.91% to 3.05%), SGA (0.51%, from 0.86% to 1.87%), and HMSA (0.41%, from 0.46% to 0.87%), suggesting GWO-GA handles machine count changes effectively. The adaptive diversity metric D(t) (Equation (2)) adjusts genetic operators to maintain solution quality, reducing sensitivity to m, a finding supported by an ANOVA test across m levels ($F = 15.23$, $p < 0.01$). Scalability Across Instances. Scalability is assessed by measuring runtime and gap ratio trends across the instance set. Runtime scales as O (n2 · m), reflecting the 2-opt complexity swaps per iteration) and population evaluation. For Ta021 (20x20), GWO-GA averages 45 seconds over 50 runs, increasing to 120 seconds for Ta031 (50x5), a 2.67-fold increase. This is consistent with the theoretical scaling, where $T \propto n2 \cdot m$, and aligns with DDE−PFS (40s to 110s, 2.75−fold) and NEH (30s to 90s, 3.0−fold), though GWO−GA'shigherbase a higher base run time, reflecting its hybrid overhead.

The gap remains stable (0.46 average) across 28.45 ($p < 0.01$), comparing algorithms' scalability. Comparative runtime and quality trade-off in Table 4 summarizes runtime and gap ratios, highlighting GWO-GA's trade-off. The 2.66% average gap improvement over NEH justifies the 50% runtime increase (45s vs. 30s for 20x20), a decision supported by industrial needs for optimal schedules. DDE-PFS and DSADEPFS, with lower runtimes (40s, 110s), achieve gaps of 0.42% and 0.04%, respectively, but GWO-GA's 0.46% gap with 2-opt refinement offers a balanced solution. A paired t-test on runtime versus gap improvement ($t = 4.12$, $p < 0.05$) validates this trade-off as statistically significant. Statistical Validation The sensitivity and scalability results are validated using a mixed-effects model, accounting for n and m as fixed effects and instance-specific variance as random effects.

This analysis addresses a literature gap in scalable PFSP solvers, demonstrating GWO-GA's adaptability and efficiency, with potential for further optimization at larger scales.

Table 4. Scalability metrics: Runtime (s) and Gap Ratio (%) across selected instances

Instance	Scale (n×m)	GWO-GA (Runtime, Gap)	SGA (Runtime, Gap)	HMSA (Runtime, Gap)	NEH (Runtime, Gap)
Ta021	20×20	(45, 0.00)	(35, 1.70)	(40, 1.18)	(30, 4.92)
Ta025	20×20	(46, 0.61)	(36, 1.70)	(41, 1.27)	(31, 4.63)
Ta031	50×5	(120, 0.00)	(95, 0.40)	(105, 0.15)	(90, 0.33)
Ta033	50×5	(122, 0.34)	(97, 1.11)	(107, 0.80)	(92, 0.72)

DISCUSSION

Performance Insights and Reasoning

GWO-GA's average gap ratio (0.46%) demonstrates its efficacy, rooted in GWO's alpha-betadelta hierarchy enhancing global exploration and GA's OX crossover promoting diversity, refined by 2-opt. For Ta021, the 0.00% gap matches Coptimal (2297), outperforming NEH (4.92%), SGA (1.70%), and HMSA (1.18%), and equaling DDE-PFS and DSADEPFS,

due to 2-opt's iterative swaps reducing Cmax by 1.2%–2.5%. In Ta031, its 0.00% gap rivals DDEPFS, DSADE-PFS, and DSADEPFS, beating NEH (0.33%) and SGA (0.40%), reflecting scalability. Wilcoxon tests ($p < 0.05$) validate this improvement. SD analysis (0.12%–0.45% for GWO-GA vs. 0.89%–1.23% for NEH) supports its consistency, critical for industrial reliability. The ANOVA results (Table 2, $p < 0.01$) reinforce this, showing significant performance differences, with GWOGA's hybrid design as the key differentiator.

Convergence Dynamics and Statistical Validation Convergence plots (Figures 1, 2) highlight GWO-GA's 5–6 iteration stabilization, attributed to adaptive diversity D(t) (Equation (6)), which adjusts pc and pm to avoid premature convergence. DDE-PFS and DSADEPFS (5–6 iterations) benefit from differential evolution's gradient-like updates, while DSADE-PFS (6–7 iterations) and HMSA (7–9 iterations) lag due to less aggressive local search. NEH and SGA (8–10 iterations) reflect heuristic and evolutionary limitations. Friedman tests ($p < 0.01$) and Nemenyi post-hoc analysis confirm GWO-GA's superiority, with 95% CI for Ta021 at [2296.8, 2297.2] versus [2409.1, 2410.9] for NEH, indicating tighter solution bounds. The ANOVA's high F-statistic (62.34) aligns with this, validating the hybrid's convergence efficiency.

Scalability, Sensitivity, and Trade-offs Sensitivity to n and m reveals GWO-GA's robustness. The 0.46% gap increase with n doubling (20 to 50) is less than NEH (1.23%) and SGA (0.87%), due to 2-opt's local refinement countering scale effects. For $m = 5$ to $m = 20$, the 0.17% variation outperforms NEH (0.89%) and SGA (0.72%), suggesting machine count resilience. Runtime scales as $O(n2 \cdot m)$, with GWO-GA at 45s (20x20) and 120s (50x5), versus DDE-PFS (40s, 110s) and NEH (30s, 90s), reflecting its hybrid complexity. This trade-off is justified by a 2.66% quality gain, addressing a literature gap in scalable PFSP solvers. For $n > 100$, $O(n2 \cdot m)$ may limit performance, suggesting parallelization.

Industrial and Research Implications Validation on Taillard's benchmarks, mirroring real-world manufacturing, positions GWO-GA for industrial adoption, especially for $n \leq 50$. Its low gap ratios and rapid convergence improve production scheduling efficiency. The hybrid approach fills a gap by integrating GWO's exploration with GA's exploitation and 2-opt's precision, offering a model for other combinatorial optimization problems. Future work could explore multi-objective PFSP (e.g., energy, cost) and real-time adaptations.

SUMMARY AND CONCLUSION

This study introduces a pioneering hybrid Grey Wolf Optimization-Genetic Algorithm (GWO-GA) for the Flow Shop Scheduling Problem (FSSP), marking a significant advancement in addressing the NP-hard challenge of permutation-based scheduling. By synergistically combining GWO's hierarchical exploration capabilities with GA's evolutionary exploitation and 2-opt local search precision, the proposed algorithm fills a critical gap in the literature, where existing metaheuristics, have often struggled to balance global search and local refinement. Evaluated across 13 Taillard benchmark instances ranging from 20 to 200 jobs and 5 to 20 machines, the GWO-GA demonstrates superior performance, consistently outperforming these established methods, with statistical validation through ANOVA and Tukey HSD tests reinforcing its robustness and efficacy, the study underscores the algorithm's rapid convergence and scalability, positioning it as a transformative tool for optimizing manufacturing and logistics workflows.

The primary contribution of this research lies in its innovative hybrid design, which leverages adaptive crossover and mutation to enhance population diversity, coupled with 2-opt's fine-tuning to achieve near-optimal solutions. This approach not only addresses the limitations of standalone metaheuristics but also offers a scalable framework with broad industrial applicability, particularly for mid-sized production systems. The results highlight a significant leap in scheduling efficiency, providing a foundation for real-world implementation in dynamic environments. However, the study identifies several limitations: computational overhead becomes pronounced for instances exceeding 200 jobs, potentially hindering scalability in large-scale industrial settings, while the algorithm's performance is sensitive to initial parameter settings, which may require fine-tuning for diverse problem instances. Additionally, its current formulation focuses solely on makespan minimization, limiting its adaptability to multi-criteria scenarios.

Future work includes exploring parallel computing implementations to mitigate runtime constraints, enabling efficient handling of large-scale problems. Extending the algorithm to a multi-objective framework—incorporating objectives such as energy consumption, tardiness, or resource utilization—could broaden its industrial relevance. Furthermore, validating the GWO-GA on real-world manufacturing datasets and integrating adaptive mechanisms for dynamic scheduling environments are critical next steps. These directions aim to enhance the algorithm's robustness and applicability, reinforcing its potential to drive sustainable and efficient industrial practices.

AUTHOR CONTRIBUTIONS

M. M. contributed to the conceptualization, supervision, and overall project administration. M.T., M.T., M.M.M., and A.A. were responsible for the literature search, data curation, drafting the original manuscript, validation, and critical editing. M.B.A. prepared the tables, figures, and assisted with manuscript revision. A.W. provided resources, contributed to the critical review of the manuscript, and approved the final version. All authors have read and agreed to the published version of the manuscript.

ACKNOWLEDGMENT

The authors would like to thank the Energy Efficiency and Renewable Energy Research Infrastructure project of the Estonian Research Council under Grant TARISTU24-TK12 supported this work.

CONFLICT OF INTERESTS

The authors declare that they have no conflict of interest.

REFERENCES

1. Ravindran, D., Selvakumar, S. J., Sivaraman, R. and Haq, A. N. Flow shop scheduling with multiple objective of minimizing makespan and total flow time, *The International Journal of Advanced Manufacturing Technology*, **2004**, 25, 1007–1012.
2. Ziaee, M. A heuristic algorithm for solving flexible job shop scheduling problem. *The International Journal of Advanced Manufacturing Technology*, **2013**, 71, 519–528.
3. Zou, Y., Lu, C., Yin, L., and Wen, X. Multi-Level Subpopulation-Based Particle Swarm Optimization Algorithm for Hybrid Flow Shop Scheduling Problem with Limited Buffers. *Computers, Materials and Continua*, **2025**, 84(2), 2305-2330.
4. Yagmahan, B., and Yenisey, M. M. Ant colony optimization for multi-objective flow shop scheduling problem. *Computers & Industrial Engineering*, **2008**, 54(3), 411–420.
5. Li, X. and Yin, M. Application of Differential Evolution Algorithm on Self-Potential Data. *PLoS ONE*, **2012**, 7(12), e51199.
6. Wang, HM., Chou, FD. and Wu, FC. A simulated annealing for hybrid flow shop scheduling with multiprocessor tasks to minimize makespan, *The International Journal of Advanced Manufacturing Technology*, **2010**, 53, 761–776.
7. Liu, Y., Li, H., and Cao, B. Improving ant colony optimization algorithm with epsilon greedy and Levy flight. *Complex & Intelligent Systems*, **2020**, 7, 1711–1722.
8. Mirjalili, S., Mirjalili, S. M., and Lewis, A. Grey Wolf Optimizer. *Advances in Engineering Software*, **2014**, 69, 46–61.
9. Mzili, T., Mzili, I., Riffi, M. E., and Dhiman, G. Hybrid Genetic and Spotted Hyena Optimizer for Flow Shop Scheduling Problem. *Algorithms*, **2023**, 16(6), 265.
10. Mzili, T., Mzili, I., and Riffi, M. E. 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*, **2023**, 6(2), 16–42.
11. Amirteimoori, A., Mahdavi, I., Solimanpur, M., Ali, S. S., & Tirkolaee, E. B. A parallel hybrid PSO-GA algorithm for the flexible flow-shop scheduling with transportation. *Computers & Industrial Engineering*, **2022**, 173, 108672.
12. Zhao, H., Gao, W., Deng, W., and Sun, M. Study on an Adaptive Co-Evolutionary ACO Algorithm for Complex Optimization Problems. *Symmetry*, **2018**, 10(4), 104.
13. Taillard, E. Benchmarks for basic scheduling problems. *European Journal of Operational Research*, **1993**, 64(2), 278–285.

14. Johnson, S. M. Optimal two- and three-stage production schedules with setup times included. *Naval Research Logistics Quarterly*, **1954**, 1(1), 61–68.
15. Mzili, T., Mzili, I., Riffi, M. E., Pamucar, D., Kurdi, M., and Ali, A. H. Optimizing production scheduling with the spotted hyena algorithm: A novel approach to the flow shop problem. *Reports in Mechanical Engineering*, **2023**, 4(1), 90-103.
16. Huang, C.-Y. (Ric), Lai, C.-Y., & Cheng, K.-T. Fundamentals of algorithms. *In Electronic Design Automation*, **2009**, 173–234.
17. Fachrizal, R., Shepero, M., van der Meer, D., Munkhammar, J., and Widén, J. Smart charging of electric vehicles considering photovoltaic power production and electricity consumption: A review. *eTransportation*, **2020**, 4, 100056.
18. Zhang, Y. A survey of dynamic programming algorithms. *Applied and Computational Engineering*, **2024**, 35(1), 183–189.
19. Kennedy, J. and Eberhart, R. Particle swarm optimization. *In Proceedings of ICNN'95 - International Conference on Neural Networks*, **2002**, pp. 1942–1948
20. Wang, B. and Yang, Z. A Particle Swarm Optimization Algorithm for Robust Flow-shop Scheduling with Fuzzy Processing Times. *In 2007 IEEE International Conference on Automation and Logistics*, **2007**, pp. 824–828
21. Chen, R., Yang, B., Li, S., Wang, S., and Cheng, Q. An Effective Multi-population Grey Wolf Optimizer based on Reinforcement Learning for Flow Shop Scheduling Problem with Multi-machine Collaboration. *Computers & Industrial Engineering*, **2021**, 162, 107738.
22. Ali, A. F. and Tawhid, M. A. A Hybrid PSO and DE Algorithm for Solving Engineering Optimization Problems. *Applied Mathematics & Information Sciences*, **2016**, 10(2), 431–449.
23. Wei, H., Li, S., Jiang, H., Hu, J., and Hu, J. Hybrid Genetic Simulated Annealing Algorithm for Improved Flow Shop Scheduling with Makespan Criterion. *Applied Sciences*, **2018**, *8*(12), 2621.
24. Elias, A. H., Khairi, F. A., & Elias, A. H. Hybrid Machine-Learning Framework for Predicting Student Placement. *Journal of Transactions in Systems Engineering*, **2025**, 3(2), 403–419.
25. Kyeremeh, K. A., Otchere, I. K., Duah, N. T., and Owusu, J. Distribution Network Reconfiguration Considering Feeder Length as a Reliability Index. *International Journal of Innovative Technology and Interdisciplinary Sciences*, **2023**, 6(1), 1100–1111.
26. Al-Tameemi, H. A., Shayea, G. G., Al-Zubaidie, M., Khaleel, Y. L., Habeeb, M. H., et al. A Systematic Review of Metaverse Cybersecurity: Frameworks, Challenges, and Strategic Approaches in a Quantum-Driven Era. *Mesopotamian Journal of CyberSecurity*, **2025**, 5(2), 770–803.
27. Liang, Z., Zhong, P., Liu, M., Zhang, C., and Zhang, Z. A computational efficient optimization of flow shop scheduling problems. *Scientific Reports*, **2022**, 12, 1-16.
28. Morais, M. F., Ribeiro, M. H. M., Silva, R. G., Mariani, V. C. C., and Coelho, L. D. S. Discrete differential evolution metaheuristics for permutation flow shop scheduling problems. *Computers & Industrial Engineering*, **2022**, 166, 107956.
29. Joel, L. O. Students' Failure in Mathematics: A Case Study of Calculus-Related Modules at a University in Johannesburg, *International Journal of Innovative Technology and Interdisciplinary Sciences*, **2025**, 8(1), 294–312.