Enhancing Hybrid Flow Shop Scheduling Problem with a Hybrid Metaheuristic and Machine Learning Approach for Dynamic Parameter Tuning

Ahmed Abdulmunem Hussein [1]*
University of Samarra. [1]
Samarra, Iraq.
ahmed.abd@uosamarra.edu.iq [1]

***Abstract*— This paper addresses the Hybrid Flow Shop Scheduling Problem (HFSSP) by integrating metaheuristic (MHs) and machine learning (ML) approaches. Specifically, we propose a hybrid algorithm by combining Ant Colony Optimization (ACO) and Iterated Local Search (ILS) to form ACOILS. To further enhance the performance of this hybrid approach, we employ Proximal Policy Optimization (PPO), which is used for dynamic tuning of key parameters within the hybrid algorithm. The introduction of PPO allows real-time adjustment of key parameters, such as pheromone evaporation rates and local search intensity, to balance exploration and exploitation more effectively. Comparative experiments against the non-learning version of ACOILS and Simulated Annealing (SA) show that the learning based LACOILS significantly reduces the percentage deviation from the lower bound while maintaining stable performance through dynamic tuning. In terms of numerical results, LACOILS consistently outperforms SA and ACOILS. For smaller instances (N=20), it achieves up to 56.52% improvement over ACOILS and 12.5% over SA. For larger instances (N=150), LACOILS shows up to 29.82% improvement over ACOILS and 9.09% over SA, demonstrating its superior solution quality and efficiency.**

Keywords*— *Hybrid Flow Shop Scheduling Problem, Ant Colony Optimization, Iterated Local Search, Proximal Policy Optimization, Machine learning.

I. INTRODUCTION

Combinatorial optimization problems such as the HFSSP are complex and critically problem in real world manufacturing and production settings[1]. These problems involve scheduling jobs of multiple stages under various constraints by which necessitate sophisticated approach to find near optimal solutions. MHs including Genetic Algorithm (GA)[2], Particle Swarm Optimization (PSO)[3], Ant Colony Optimization (ACO)[4], and Iterated Local Search (ILS)[5] have been widely used to solve these problems through exploring large solutions spaces and enhancing those solutions through local optimization. However, MHs often struggles to balance between exploration and exploitation specifically in the dynamic and large scale problems where the search space evolve and problem constraints grow rapidly[6]. In large scale HFSSP the traditional methods might fail to adapt to changes in job priorities or resource availability which is leading to suboptimal scheduling outcomes.

The integration of ML with MHs has recently emerged as promising approach to overcome these limitations through enhancing the adaptability and performance[7]. By dynamically adjusting MHs parameters and components based on real-time feedback the ML can significantly improve the decision-making process within MH frameworks [8]. Reinforcement learning (RL) methods in particular the PPO has been effective in enhancing parameter tuning. PPO iterative learning process allows the algorithm to adjust in dynamic way to change the conditions and diverse problem instances by fine tuning parameters continuously[9]. This capability is crucial for improving algorithms like ACO and ILS enabling them to adapt better to dynamic scheduling problems.

In this paper we propose hybrid algorithm that integrates Ant Colony Optimization and Iterated Local Search (ACOILS) and enhances it with PPO based dynamic parameter tuning. PPO is utilized to adjust critical parameters with the ACOILS such as pheromone evaporation rates and local search. The choice of PPO motivated by its advantages over other RL algorithms like Deep Q-Network (DQN)[10] and Asynchronous Advantage Actor-Critic (A3C)[11] as PPO offers greater stability, efficiency, and faster convergence. DQN and A3C while effective but are typically require long time to train and run which making PPO a more suitable for parameter adjustments in complex optimization problems[12]. This adaptive learning approach enables the hybrid algorithm to fine tune the balance between exploration and exploitation which improving the consistency and quality of solutions across various problem instances.

The integration of PPO with ACOILS provides several advantages which includes the ability to dynamically adapt algorithm behavior and address the limitations of static parameter tuning found in traditional MHs approaches. By allowing real-time adjustments the hybrid algorithm better handle the complexity and variability inherent in HFSSP where problem instances can differ significantly in size and constraints. The main contribution of this work lies in the novel hybridization of ACO and ILS and then enhanced by PPO based parameter tuning in order to offer more efficient and robust solution for HFSSP.

The experimental results demonstrate that the proposed

p-ISSN 2301-7988, e-ISSN 2581-0588
DOI : 10.32736/sisfokom.v13i3.2290, Copyright ©2024
Submitted : October 4, 2024, Revised : October 19, 2024, Accepted : October 21, 2024, Published : November 21, 2024

method able to achieve significant improvements across almost all instances of the scheduling problem which deliver robust framework for solving complex HFSSP scenarios. The adaptive capability of PPO ensures that the algorithm maintains high performance even in large scale dynamic environments.

II. LITERATURE REVIEW

The HFSSP is important area of research in production scheduling due to its complexity and significance in real-world manufacturing. HFSSP involves scheduling multiple jobs of various stages within parallel machines each stage having distinct capabilities which adds layers of complexity such as machine availability constraints, sequence dependent setup times and limited capacity. HFSSP in addition must often handle dynamic environment like random job arrivals, machine breakdowns, and fluctuating processing times, making it challenging to find optimal solutions. Researchers have explored many MHs techniques enhanced by ML to address these challenges effectively.

A Genetic Programming Hyper-Heuristic (GPHH) has been proposed for dynamic energy efficient scheduling of HFSSP where GP employed to generate job sequencing and machine assignment rules. This approach is effective in environments characterized by machine breakdowns and random job arrival. By dynamically generating and optimizing scheduling rules the method is able of adapting to the stochastic nature of real-time scheduling environments [13].

In another approach the Shuffled Frog-Leaping Algorithm (SFLA) integrated with Q-learning (QL) to optimize distributed assembly hybrid flow shop scheduling. The Q-learning component dynamically select search strategies guiding the algorithm global and local search operators. This cooperation between MHs and RL improve both solution diversity and convergence leading to better performance in distributed scheduling environments [14].

A Cooperative Memetic Algorithm (CMA) combined with RL based agent has been developed for energy efficient scheduling in distributed HFSSP settings. The RL agent enhances the algorithm ability to select appropriate local search operators based on the current problem state which significantly reduces energy consumption while maintaining high quality solutions. The method demonstrated significant improvements in achieving energy aware schedule in large scale manufacturing systems [15].

Further, Multiobjective Memetic Algorithm incorporating PSO and QL-based local search have been introduced to optimize energy efficient scheduling in distributed hybrid flow shop. The QL mechanism allow the local search to dynamically adjust its parameters based on real-time feedback ensuring improved performance on energy consumption and scheduling efficiency in large scale distributed environment [16].

The Meta Reinforcement Learning MHs (MRLM) framework use RL to dynamically select search operators allowing the MHs to adapt to changes in worker productivity caused by learning and forgetting effects. This approach proved effective in improving scheduling performance especially in labor intensive industries where job processing times are affected by human factor[17].

Another study introduces a RL based task splitting strategy for HFSSP where QL is used to dynamically split tasks across machines in distributed system. The adaptability provided by RL allowed for significant improvements in makespan and resource utilization particularly in mass personalized manufacturing environments[18].

A novel approach that applies QL-based Teaching-Learning Optimization (TLBO) has been used to address HFSSP with fuzzy processing times. QL dynamically adjusts the phases of the TLBO algorithm, enabling it to handle uncertainties in job processing times more effectively. This integration of ML into the optimization process significantly improved the robustness and efficiency of the scheduling outcomes[19].

Lastly SA has been employed to address HFSSP with complex constraints such as machine availability and delivery times. Although this approach does not incorporate ML it highlights the effectiveness of SA in achieving near-optimal solutions particularly for large-scale problem instances. The study demonstrates how dynamic parameter adjustment in SA contributes to better performance in hybrid flow shop scheduling[20].

III. PROBLEM DESCRIPTION

The HFSSP presented in this paper involves the scheduling of n jobs through two stages of production. The first stage consists of a single machine that processes all jobs, and the second stage includes m dedicated parallel machines, where each machine can process a specific subset of jobs. Each job J_i (where $i = 1,2,\dots,n$) must be processed first on the common machine in Stage 1 and then assigned to one of the dedicated machines in Stage 2 based on the job's type. The primary objective is to minimize the makespan C_{max}, defined as the total time required to complete all jobs[1]. HFSSP has the following assumptions**:**

- All jobs must be processed on the single machine in Stage 1 before being processed on any of the dedicated machines in Stage 2.
- Each job J_i has a specific release date r_i , which is the time when the job becomes available for processing.
- Each job J_i has a specific delivery time d_i , which is the latest time by which the job must be completed.
- All machines in both stages are non-preemptive, meaning that once a job starts processing on a machine, it cannot be interrupted until completion.

Notation

n: Number of jobs to be scheduled.
J_i : The set of jobs $i = 1,2,\dots,n$.
M_i : The single common machine in the first stage.
M_j: Dedicated machines $j = 1,2,\dots,m$ in the second stage.
p_{i1}: Processing time of job i on the common machine.
p_{ij}: Processing time of job i on dedicated machine M_j.
r_i : Release date of job i (time when the job becomes available).
d_i: Delivery time of job i (deadline by which the job must be

p-ISSN 2301-7988, e-ISSN 2581-0588
DOI : 10.32736/sisfokom.v13i3.2290, Copyright ©2024
Submitted : October 4, 2024, Revised : October 19, 2024, Accepted : October 21, 2024, Published : November 21, 2024

completed).
C_{i1}: The completion time of job J_i on the common machine.
C_{i2}: The completion time of job J_i on the dedicated machine.
S_{i1}: The start time of job J_i on the common machine.
S_{i2}: The start time of job J_i on the dedicated machine.
C_{max}: The makespan, or the total time to complete all jobs.

$$\min C_{max} = \max_i\{C_{i2}\} \quad (1)$$

$$C_{i1} \leq S_{i2}, \ \forall i \quad (2)$$

$$S_{i1} \geq r_i, \ \forall i \quad (3)$$

$$C_{i2} \leq d_i, \ \forall i \quad (4)$$

$$C_{i1} = S_{i1} + p_{i1}, \quad C_{i2} = S_{i2} + p_{ij} \quad , \ \forall i,j \quad (5)$$

$$S_{i1} \geq C_{k1}, \ S_{i2} \geq C_{k2} \quad , \ \forall i \neq k \quad (6)$$

The objective of the problem is to minimize the makespan as defined in (1). The first constraint (2) ensures that the first job processed on the common machine in Stage 1 before moving to its assigned dedicated machine in Stage 2. The second constraint (3) impose that no job can start processing before its release date to ensure that the job become available only after its designated release time. The third constraint (4) ensure that each job must be completed before or by its specific delivery time so respecting deadlines for job completion. The non-preemption constraint (5) dictate that once the job start processing on any machine it is must run uninterrupted until completion. This constraint guarantees that no job can be paused or preempted during processing. The last constraint is the machine availability constraint (6) that ensure that each machine can process only one job at a time preventing any overlap in job assignments on the same machine. This ensures that the scheduling adhere to the availability and capacity of the machines with each job waiting until the previous one is completed. These constraints collectively define the problem and guide the scheduling decisions to minimize the makespan[1].

IV. METHODOLOGY

The methodology is applied to solve the HFSSP by integrating ACO for generating initial schedules and ILS for refining these schedules to enhance ACO's exploitation capabilities and escaping local optima. PPO dynamically adjusts ACOILS parameters based on real-time performance improving the balance between exploration and exploitation to achieve more efficient scheduling solutions as shown in Fig. 1.

A. Ant Colony Optimization

ACO is a population based MHs inspired by the foraging behavior of ants. Ants leave pheromone trails on paths they traverse, and subsequent ants are more likely to follow these paths based on pheromone strength, creating a positive feedback loop. In ACO, artificial ants build solutions to optimization problems by probabilistically choosing components (such as in job assignments in scheduling problems) based on the intensity of the pheromone and the heuristic information. Over time, pheromone evaporation ensures that suboptimal solutions are less attractive. ACO is particularly well-suited for combinatorial optimization problems like the job shop scheduling and flow shop scheduling[4].

B. Iterated Local Search

ILS is single-based MHs that focuses on the exploitation of local optima. Starting from an initial solution, ILS repeatedly applies a local search method to explore the neighborhood of the current solution. When a local optimum is found the algorithm perturbs the current solution to escape the local minimum and re-applies local search from the new starting point. The process of local search and perturbation continues iteratively enabling the algorithm to explore a larger search space and escape poor-quality local minima. ILS is often used for fine-tuning solutions in difficult optimization problems[5].

C. Hybrid ACO and ILS

In this hybrid approach ACO is responsible for exploration (diversity) by constructing solutions based on pheromone trails, while ILS is used to exploit the best solutions found by ACO to refine them further using local search. The ILS helps ACO in enhancing the solution quality by

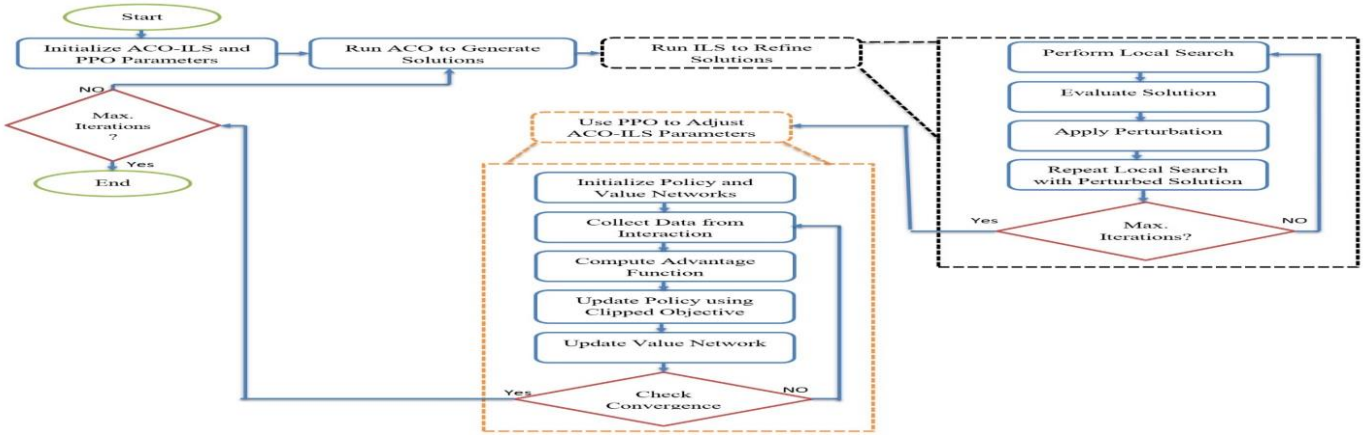

Fig. 1 The flow process of the LACOILS

p-ISSN 2301-7988, e-ISSN 2581-0588
DOI : 10.32736/sisfokom.v13i3.2290, Copyright ©2024
Submitted : October 4, 2024, Revised : October 19, 2024, Accepted : October 21, 2024, Published : November 21, 2024

focusing on exploitation (intensification) to reach near-optimal solutions. The steps of algorithm1 hybrid ACOILS algorithm are as follow:

1) Initialization:

a) Set the ACO parameters such as the pheromone matrix, number of ants, evaporation rate, and heuristic factors.

b) Set ILS parameters including perturbation strength and the local search method.

c) Initialize the pheromone trails for the solution space.

2) Ant Colony Exploration:

a) For each iteration every ant constructs a solution based on pheromone strength and heuristic values.

b) Ants probabilistically choose components to build their solutions using a balance of exploration (pheromone) and exploitation (heuristic values).

3) Local Search (ILS):

a) After each ant constructs a solution, apply the ILS method to refine it.

b) Perform local search to explore the neighborhood of the solution and reach a local optimum.

c) Perturb the local optimum if necessary to escape suboptimal local minima and perform further searches.

4) Pheromone Update:

a) If the solution found by ILS improves upon the best known solution (BKS), deposit pheromone on the solution components.

b) Update the pheromone matrix based on the quality of the solution found.

c) Evaporate pheromone to avoid premature convergence and enable exploration of new areas in the solution space.

5) Termination:

a) Repeat the process until a stopping criterion is met (maximum number of iterations).

b) Output the best solution found.

Algorithm 1: Hybrid ACOILS

Initialize ACO parameters:
  Pheromone matrix (τ): Initial pheromone levels for the solution space
  Number of ants (m): Size of the ant colony
  Evaporation rate (ρ): Controls how quickly pheromones decay
  Alpha (α): Weight of the pheromone influence in decision-making
  Beta (β): Weight of heuristic information influence in decision-making
  Heuristic factors (η): Problem-specific information such as job processing times or machine availability
Initialize ILS parameters:
  Perturbation strength: Determines how much a solution is modified to escape local optima
  Local search method: Defines the technique used to refine solutions
  Number of iterations: Maximum iterations for local search
  Acceptance criterion: Condition to accept or reject new solutions
Generate initial pheromone matrix (initialize pheromone trails with τ_0)
for each ACO iteration do
  for each ant in the ant colony do
    Construct a solution using ACO:
      Select solution components based on pheromone trails (τ) and heuristic values (η)
      Probabilistically balance exploration and exploitation based on α and β
    Apply local search to the constructed solution using ILS:
      Explore the neighborhood of the current solution using local search method
      Perturb the solution if necessary to escape local minima
    if the refined solution improves the current best solution then
      Update the best solution found
    Deposit pheromone on the paths/components of the improved solution:
      Strength of deposition depends on the quality of the solution
  end for
  Evaporate pheromone globally ($\tau = (1-\rho) * \tau$) to prevent stagnation and encourage exploration
end for
Output the best solution found by the hybrid ACO-ILS algorithm

D. Proximal Policy Optimization

PPO is a policy gradient based RL algorithm designed to improve training stability and performance. It optimizes the policy by directly updating it based on gradients using a combination of trust region methods and stochastic gradient descent. PPO simplifies the complexity of Trust Region Policy Optimization (TRPO) while maintaining strong performance[21].

PPO achieves this by using a clipped objective function that prevents the policy from making large updates, ensuring the agent's actions do not change too drastically in a single update. This "proximal" step ensures that the policy update stays within a reasonable range of the current policy, improving stability while encouraging efficient exploration. The key components of PPO are Policy Network which maps the state to an action. The second component is value network (Critic) at which evaluates the expected return (advantage function) of the current state. Thirdly, clipped objective function where PPO uses a clipped surrogate objective to limit how much the policy can change between updates, which avoids large steps that could destabilize learning. The goal is to dynamically tune parameters of the hybrid ACOILS algorithm using PPO, which adjusts parameters like pheromone evaporation rate, exploration-exploitation balance, and perturbation strength in ILS. The PPO agent learns a policy to adjust these parameters based on the performance of the hybrid algorithm. The steps of algorithm2 PPO are as follow:

1) **Initialization**:

a) Initialize PPO parameters, such as the policy network, value network, learning rate, clipping threshold, and batch size.

b) Define the action space for parameter tuning (the parameter of ACOILS).

c) Set the initial policy and value networks (actor and critic).

2) **State Representation**:

a) Define the state as the current performance of the ACOILS algorithm, which could include metrics like the quality of the solution, iteration progress, and algorithm behavior (e.g., how many ants converge to the same solution).

b) Collect performance feedback from the hybrid ACOILS algorithm to form the state.

p-ISSN 2301-7988, e-ISSN 2581-0588
DOI : 10.32736/sisfokom.v13i3.2290, Copyright ©2024
Submitted : October 4, 2024, Revised : October 19, 2024, Accepted : October 21, 2024, Published : November 21, 2024

3) **Action Representation**:

a) Define the actions as adjustments to the ACOILS parameters. For example, actions might include:

b) Adjusting the pheromone evaporation rate.

c) Changing the number of iterations for local search.

d) Modifying the balance between exploration and exploitation.

4) **Reward Calculation**:

a) The reward is based on the improvement in solution quality after tuning the parameters. For example, if the solution found by ACO-ILS improves significantly due to the tuned parameters, a higher reward is given. You can also consider speed or stability improvements.

5) **PPO Policy Update**:

a) For each update step, run ACO-ILS with the current parameters selected by the PPO policy.

b) Compute the reward based on solution improvement.

c) Use the value network to compute the advantage function, which estimates how much better the action was compared to the baseline.

Algorithm 2: Proximal Policy Optimization

Input:
Initialize policy network (πθ) with random weights
Initialize value network (Vθ) with random weights
Set learning rates for actor and critic
Set clipping threshold for PPO updates
Set batch size and number of epochs

Initialize ACO and ILS parameters (pheromone evaporation rate, local search parameters, etc.)
Define action space (possible parameter ranges)
while not converged do

  for each episode (ACO-ILS run) do
    Initialize state S (e.g., solution quality, algorithm performance metrics)

    A = πθ(S) // Select action (tune ACOILS parameters) based on current policy πθ
    Execute action A (adjust pheromone rate, local search settings, etc.)

    Run ACO-ILS for one full iteration using new parameters
    NewState = Current state of ACO-ILS after running iteration
    Reward = Improvement in solution quality, speed, or stability
    Store (S, A, Reward, NewState) in memory

    // Update current state
    S = NewState
  end for
  Compute advantage A(S, A) using the value network
  for each batch of experience do
    Calculate the ratio of probabilities between old and new policies r(θ)
    Use the clipped objective function to compute the surrogate loss:
    L(θ) = min(r(θ) * A(S, A), clip(r(θ), 1 - ε, 1 + ε) * A(S, A))
    Perform gradient descent on L(θ) to update the policy network (actor)
    Update the value network by minimizing the value loss:
    L(V) = (V(S) - target_value)^2
  end for
end while
Output: Best tuned parameters for ACO-ILS

d) Update the policy network using the clipped objective function to ensure that parameter changes remain within a stable range.

6) **Repeat Until Convergence**:

a) Continue iterating through ACO-ILS runs, using the PPO agent to adjust the parameters dynamically.

b) Stop when a predefined stopping criterion is reached (e.g., maximum iterations or convergence of solution quality).

E. Experimental Setup

The experiments were conducted using a laptop equipped with an AMD Ryzen 7 5800H processor, and 16GB of RAM. This hardware configuration ensures sufficient computational power to handle the large-scale optimization problems tackled in this study including the HFSSP. To evaluate the performance of the proposed algorithm the problem instances were generated following the structure described in Hajji et al.'s work[20]. Various instance classes with differing job sizes (N=20,50,100,150) and multiple machine configurations are considered. Processing times, release dates, and delivery times are randomly generated from predefined ranges for each instance to ensure diverse and challenging set of test cases. The experiments included multiple iterations per instance to capture the variability and consistency of the solution quality with comparisons made against known lower bounds and computational benchmarks.

V. RESULTS AND DISCUSSION

The computational results of this study presented in this section. ACOILS and LACOILS applied to the HFSSP. While ACOILS used the Taguchi method for offline parameter tuning[22], LACOILS employed PPO to dynamically tune its parameters in real time. In table I PPO was used to

Table I. Parameters values for LACOILS and ACOILS

Parameter	LACOILS	ACOILS	Role of the parameter
Pheromone matrix (τ)	0.1	0.3	Represents the pheromone trail strength, which influences the probability of choosing certain solution components.
Number of ants (m)	Number of jobs		The number of ants constructing solutions in each iteration.
Evaporation rate (ρ)	0.2	0.1	Controls how much pheromone is retained on each path after every iteration.
Alpha (α)	1	2	A parameter that weights the influence of pheromone trails on decision-making.
Beta (β)	2	2	A parameter that weights the influence of heuristic information (e.g., job durations, machine availability).
Heuristic factors (η)	$\eta = 1/job\ processing\ time$		Provide additional guidance during solution construction, such as job priority or shortest processing time.
Perturbation strength	14%	20%	Determines the magnitude of changes made to the current solution to escape local optima.
Number of iterations	372	484	The number of local search steps applied to improve solutions.

p-ISSN 2301-7988, e-ISSN 2581-0588
DOI : 10.32736/sisfokom.v13i3.2290, Copyright ©2024
Submitted : October 4, 2024, Revised : October 19, 2024, Accepted : October 21, 2024, Published : November 21, 2024

Table II. Comparison results of the proposed algorithms.

Instance	Algorithms	N = 20		N = 50		N = 100		N = 150	
		Dev. %	LB	Dev. %	LB	Dev. %	LB	Dev. %	LB
Cl1	SA	0.11	17	0.20	11	0.15	8	0.28	2
	ACOILS	0.23	12	0.24	9	0.19	7	0.33	0
	LACOILS	**0.1**	**20**	**0.17**	**15**	**0.11**	**10**	**0.23**	**3**
Cl2	SA	0.21	17	0.25	9	0.18	2	0.40	1
	ACOILS	0.27	14	0.29	5	0.26	1	0.47	0
	LACOILS	**0.19**	**21**	**0.2**	**12**	**0.16**	**3**	**0.36**	**2**
Cl3	SA	1.75	1	0.92	1	0.70	1	0.75	0
	ACOILS	1.86	0	1.12	0	0.74	1	0.93	0
	LACOILS	**1.71**	**3**	**0.86**	**2**	**0.64**	**2**	**0.68**	**1**
Cl4	SA	0.18	17	0.26	8	0.34	1	0.34	1
	ACOILS	0.27	10	0.35	2	0.56	0	0.42	0
	LACOILS	0.24	14	0.28	6	0.41	1	0.37	**1**
Cl5	SA	0.48	13	0.58	4	0.46	0	0.44	0
	ACOILS	0.51	8	0.68	1	0.55	0	0.57	0
	LACOILS	**0.42**	**16**	**0.55**	**6**	0.47	0	**0.4**	**1**

continuously adjust the parameters which allow LACOILS to adapt efficiently to various scenarios. This dynamic tuning method helped LACOILS balance exploration and exploitation more effectively, leading to optimized solutions and improved algorithm efficiency.

Table II compares the performance of three algorithms: SA[20], ACOILS, and LACOILS. The performance is evaluated in terms of percentage deviation (Dev.) from the lower bound (LB) across various problem sizes and instance classes.

- For smaller instances (N=20), LACOILS outperforms both SA and ACOILS, showing the lowest percentage deviations. For example, for Class Cl1, LACOILS achieves a deviation of only 0.1%, whereas SA and ACOILS exhibit higher deviations 0.11% and 0.23% respectively. This indicates that incorporating learning for dynamic parameter adjustment in LACOILS improves solution quality for smaller instances.
- As the problem size increases (N=150), LACOILS continues to show competitive performance but the deviations rise for all algorithms. For Class Cl5, LACOILS records a deviation of 0.4%, still outperforming ACOILS 0.57% and SA achieves the closer deviation for LACOILS of 0.44%. This highlights that LACOILS manages to balance exploration and exploitation better than ACOILS and SA, but LACOILS remains slightly more effective for large problem sizes.

LACOILS demonstrates clear improvements over ACOILS in smaller, medium, and large sized problems due to the PPO component that tunes algorithm parameters in real-time. In Table III, LACOILS demonstrates superior computational efficiency across all problem instances compared to both ACOILS and SA, with the success largely attributed to its use of PPO for dynamic parameter tuning. For

Table III. The Computation time of the proposed algorithm

Instance	Algorithm	N = 20	N = 50	N = 100	N = 150
Cl1	SA	<1 s	6 s	32 s	7 s
	ACOILS	1.61 s	7.99 s	31.47 s	7.5 s
	LACOILS	**0.86 s**	**5.82 s**	**28.74 s**	**5.55 s**
Cl2	SA	2 s	2 s	2 s	7 s
	ACOILS	1.83 s	1.94 s	2.92 s	8.35 s
	LACOILS	**1.74 s**	**1.65 s**	**1.95 s**	**6.42 s**
Cl3	SA	3 s	2 min	4 min	14 min
	ACOILS	3.39 s	2.58 min	5.07 min	15.46 min
	LACOILS	**2.89 s**	**1.43 min**	**3.69 min**	**13.38 min**
Cl4	SA	<1 s	5 s	39 s	1 min
	ACOILS	0.78 s	5.43 s	42.97 s	1.25 min
	LACOILS	**0.68 s**	**4.37 s**	**37.97 s**	**52.7 s**
Cl5	SA	2 s	20 s	2 min	13 min
	ACOILS	2.68 s	18.37 s	2.54 min	15.11 min
	LACOILS	**1.91 s**	**14.57 s**	**1.88 min**	**11.26 min**

smaller problem sizes, such as in Cl1 and Cl2, LACOILS consistently achieves the fastest computation times, completing Cl1 (N = 20) in 0.86 seconds, significantly outperforming ACOILS and SA. As the problem size increases, particularly in more complex instances like Cl3 and Cl5, LACOILS continues to maintain a computational advantage. For example, in Cl5 (N = 150), LACOILS finishes in 11.26 minutes, compared to 15.11 minutes for ACOILS and 13 minutes for SA. The ability of LACOILS to adaptively tune critical parameters such as pheromone evaporation and local search intensity in real time, through PPO, enables it to balance exploration and exploitation more effectively, thus reducing unnecessary computation. In contrast, ACOILS, without adaptive tuning, exhibits longer computation times due to its static parameter settings. While SA performs competitively in smaller instances, it struggles with larger problem sizes, highlighting the scalability and efficiency advantages of LACOILS driven by its integration of PPO for parameter optimization.

In Table IV, the performance of heuristic and MHs algorithms for Class Cl1 is compared across problem sizes. Among the heuristics, Hjoh1 shows the poorest performance, with high deviation percentages and rare occurrences of hitting the lower bound, while Hjoh2 performs significantly better, with deviation percentages as low as 1% and frequent occurrences at the lower bound. HNEH also performs well but falls behind Hjoh2. In contrast, the MHs algorithms consistently outperform the heuristics. SA demonstrates strong performance with low deviation percentages and frequent lower bound hits, particularly for larger problem sizes. ACOILS performs similarly to SA but with slightly higher deviations. LACOILS achieves the best results overall, with

Table IV. Comparative results of the proposed algorithm against BKS.

Algorithm	N = 20		N = 50		N = 100		N = 150	
	Dev. %	LB	Dev. %	LB	Dev. %	LB	Dev. %	LB
H_{joh1}	8.1	1	3.99	0	2.07	0	1.40	0
H_{joh2}	1.00	6	0.21	7	0.18	2	0.30	0
H_{NEH}	3.10	4	1.5	3	0.70	2	0.72	1
SA	0.11	17	0.20	11	0.15	8	0.28	2
ACOILS	0.23	12	0.24	9	0.19	7	0.33	0
LACOILS	**0.1**	**20**	**0.17**	**15**	**0.11**	**10**	**0.23**	**3**

p-ISSN 2301-7988, e-ISSN 2581-0588
DOI : 10.32736/sisfokom.v13i3.2290, Copyright ©2024
Submitted : October 4, 2024, Revised : October 19, 2024, Accepted : October 21, 2024, Published : November 21, 2024

the lowest deviations and most frequent lower bound hits across all problem sizes, demonstrating the effectiveness of dynamic parameter tuning in improving both solution quality and consistency.

A. Performance Analysis

Table V presents the Standard Deviation (SD), Mean, and Coefficient of Variation (CV)[23], which is calculated as follow:

$$CV = \frac{SD}{mean} \times 100 \quad (7)$$

for each algorithm. These metrics help assess the quality of solutions along with the consistency and stability of each method.

- LACOILS approach shows lower SD values in compared to ACOILS across multiple instances indicating that the learning based approach lead to more consistent performance. The lower SD indicate that LACOILS has less variability in its results which is crucial in real-world scheduling applications where stability is as important as solution quality.
- The CV for LACOILS approach also lower than ACOILS confirms that LACOILS provide more stable and reliable results relative to the mean solution. This support the argument that learning based parameter tuning in LACOILS reduce the randomness in search behavior which leads to more controlled and predictable outcome.

ACOILS shows higher SD and CV values reflects variability in performance. While ACOILS sometimes finds good solutions but its results are more inconsistent without dynamic learning based parameter adjustments.

The Wilcoxon signed-rank test shown in table VI was conducted to statistically compare LACOILS and ACOILS[24]. This test assesses whether the performance differences between the two algorithms are statistically significant. The test reveals a significant improvement in the performance of LACOILS over ACOILS across different instances. The p-values reported in table VI are below the significance threshold ($p<0.05$) indicating that the enhanced learning mechanism in LACOILS leads to statistically significant better results compared to ACOILS. This statistical validation strengthens the argument that incorporating RL for parameter tuning in LACOILS meaningfully improves the algorithm's performance, leading to both more optimal and more consistent results compared to the non-learning version (ACOILS).

Table V. Descriptive metrics of LACOILS performance

Instance	LACOILS			ACOILS		
	SD	Mean	CV	SD	Mean	CV
Cl1	28.7	1278	2.24%	37.5	1307	2.86%
Cl2	32.64	1397	2.33%	39.72	1427	2.78%
Cl3	40.83	1426	2.86%	48.7	1473	3.30%
Cl4	61.5	1518	4.05%	66.53	1584	4.20%
Cl5	64.37	1678	3.83%	74.62	1719	4.34%

Table VI. Wilcoxon ranked test LACOILS against ACOILS

Instance	Wilcoxon test
Cl1	0.039547
Cl2	0.023861
Cl3	0.037355
Cl4	0.041869
Cl5	0.034826

Box plots in Fig. 2 visually compare the distributions of performance between LACOILS and ACOILS. Box plots consist of several key parts: the box, which represents the interquartile range (IQR) containing the middle 50% of the data points; the whiskers, which extend from the box to the minimum and maximum values, excluding outliers; the median line, located within the box, indicating the median of the data; and outliers, which are points outside the whiskers, representing extreme values[25]. Fig. 2.a LACOILS shows a narrower IQR and fewer outliers compared to Fig. 2.b ACOILS. This indicates that LACOILS not only achieves better median performance but also exhibits less variability in its results. The smaller spread of the whiskers and box demonstrates that LACOILS delivers more consistent outcomes. Fig. 2.b ACOILS shows wider IQR and more frequent outliers indicates the higher variability and less stable performance. This aligns with the findings from the CV and SD metrics where ACOILS show a more fluctuation in its results due to the lack of adaptive parameter tuning. The box plot analysis further support the conclusion that LACOILS provides more consistent and reliable performance in compared to ACOILS with fewer extreme outlier results and narrower range of deviations. This consistency is crucial in scheduling problems where predictability and repeatability are key for practical applications.

B. Discussion

The utilization of PPO plays critical role in dynamically tuning the parameters in LACOILS significantly improving its performance in compared to ACOILS and SA. PPO enable LACOILS to adjust parameters like pheromone evaporation and local search in online real time which optimizing the

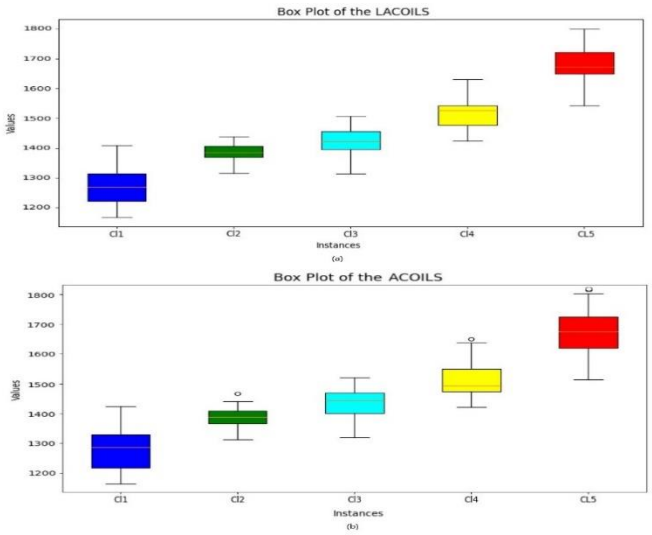

Fig. 2 The box plot distribution of the five groups of instances a) LACOILS b) ACOILS

p-ISSN 2301-7988, e-ISSN 2581-0588
DOI : 10.32736/sisfokom.v13i3.2290, Copyright ©2024
Submitted : October 4, 2024, Revised : October 19, 2024, Accepted : October 21, 2024, Published : November 21, 2024

balance between exploration and exploitation. This dynamic tuning is essential for achieving lower percentage deviations and more consistent results which evidenced by the lower SD and CV values in compared to ACOILS. The Wilcoxon test further demonstrate the advantage of learning based tuning with LACOILS consistently outperforming ACOILS while the box plots emphasize PPO impact by showing a narrower range of performance for LACOILS which indicating more stable and predictable results. The main advantages of LACOILS include its adaptability to different problem instances leading to improved solution quality and consistency particularly in small to medium sized problems and strong statistical validation through various performance metrics. However, LACOILS may be sensitive to the choice of PPO related parameters such as learning rates and exploration and exploitation trade-offs requiring extensive experimentation to fine tune these parameters effectively. This parameter sensitivity introduces additional layer of complexity complicating its application in real world and requiring further refinement in order to ensure robustness across different problem settings.

VI. CONCLUSION

This study presents a hybrid algorithm LACOILS that integrates ACO and ILS with PPO for dynamic parameter tuning to solve the HFSSP which achieving significant improvements over traditional methods like ACOILS and SA. The adaptive learning approach enabled LACOILS to better balance exploration and exploitation which leading to enhanced solution quality and computational efficiency across various problem sizes especially in smaller to medium instances. Compared to other studies which often rely on static parameter tuning the LACOILS real-time adjustments provides more effective way to handle the complexity and variability of HFSSP. The main novelty lies in the dynamic tuning capability enabled by PPO by which transform traditional MHs by making them more responsive to changing problem conditions. However, the approach sensitivity to PPO parameter selection and potential computational limitations for larger instances suggest the need for further optimization. Future work could explore alternative RL methods for parameter tuning and extend the approach to other complex scheduling scenarios.

REFERENCES

[1] R. Ruiz and J. A. Vázquez-Rodríguez, "The hybrid flow shop scheduling problem," *Eur J Oper Res*, vol. 205, no. 1, pp. 1–18, 2010.

[2] J. H. Holland, "Genetic algorithms," *Sci Am*, vol. 267, no. 1, pp. 66–73, 1992.

[3] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proceedings of ICNN'95-international conference on neural networks*, IEEE, 1995, pp. 1942–1948.

[4] M. Dorigo, M. Birattari, and T. Stutzle, "Ant colony optimization," *IEEE Comput Intell Mag*, vol. 1, no. 4, pp. 28–39, 2006.

[5] H. R. Lourenço, O. C. Martin, and T. Stützle, "Iterated local search," in *Handbook of metaheuristics*, Springer, 2003, pp. 320–353.

[6] E.-G. Talbi, *Metaheuristics: from design to implementation*. John Wiley & Sons, 2009.

[7] E.-G. Talbi, "Machine learning into metaheuristics: A survey and taxonomy," *ACM Computing Surveys (CSUR)*, vol. 54, no. 6, pp. 1–32, 2021.

[8] A. A. Hussein, E. T. Yassen, and A. N. Rashid, "Grey Wolf Optimizer for Green Vehicle Routing Problem.," *International Journal of Intelligent Engineering & Systems*, vol. 16, no. 5, 2023.

[9] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017.

[10] J. Fan, Z. Wang, Y. Xie, and Z. Yang, "A theoretical analysis of deep Q-learning," in *Learning for dynamics and control*, PMLR, 2020, pp. 486–489.

[11] M. Sewak and M. Sewak, "Actor-critic models and the A3C: The asynchronous advantage actor-critic model," *Deep reinforcement learning: frontiers of artificial intelligence*, pp. 141–152, 2019.

[12] P. Henderson, R. Islam, P. Bachman, J. Pineau, D. Precup, and D. Meger, "Deep reinforcement learning that matters," in *Proceedings of the AAAI conference on artificial intelligence*, 2018.

[13] J. Duan, F. Liu, Q. Zhang, J. Qin, and Y. Zhou, "Genetic programming hyper-heuristic-based solution for dynamic energy-efficient scheduling of hybrid flow shop scheduling with machine breakdowns and random job arrivals," *Expert Syst Appl*, p. 124375, 2024.

[14] J. Cai, D. Lei, J. Wang, and L. Wang, "A novel shuffled frog-leaping algorithm with reinforcement learning for distributed assembly hybrid flow shop scheduling," *Int J Prod Res*, vol. 61, no. 4, pp. 1233–1251, 2023.

[15] J.-J. Wang and L. Wang, "A cooperative memetic algorithm with learning-based agent for energy-aware distributed hybrid flow-shop scheduling," *IEEE Transactions on Evolutionary Computation*, vol. 26, no. 3, pp. 461–475, 2021.

[16] W. Zhang, C. Li, M. Gen, W. Yang, and G. Zhang, "A multiobjective memetic algorithm with particle swarm optimization and Q-learning-based local search for energy-efficient distributed heterogeneous hybrid flow-shop scheduling problem," *Expert Syst Appl*, vol. 237, p. 121570, 2024.

[17] Z. Zhang, Z. Shao, W. Shao, J. Chen, and D. Pi, "MRLM: A meta-reinforcement learning-based metaheuristic for hybrid flow-shop scheduling problem with learning and forgetting effects," *Swarm Evol Comput*, vol. 85, p. 101479, 2024.

[18] X. Chen *et al.*, "Reinforcement learning for distributed hybrid flowshop scheduling problem with variable task splitting towards mass personalized manufacturing," *J Manuf Syst*, vol. 76, pp. 188–206, 2024.

[19] B. Xi and D. Lei, "Q-learning-based teaching-learning optimization for distributed two-stage hybrid flow shop scheduling with fuzzy processing time," *Complex System Modeling and Simulation*, vol. 2, no. 2, pp. 113–129, 2022.

[20] M. K. Hajji, O. Hamlaoui, and H. Hadda, "A simulated annealing metaheuristic approach to hybrid flow shop scheduling problem," *Advances in Industrial and Manufacturing Engineering*, p. 100144, 2024.

[21] H. Zhong and T. Zhang, "A theoretical analysis of optimistic proximal policy optimization in linear markov decision processes," *Adv Neural Inf Process Syst*, vol. 36, 2024.

[22] J. A. Ghani, I. A. Choudhury, and H. H. Hassan, "Application of Taguchi method in the optimization of end milling parameters," *J Mater Process Technol*, vol. 145, no. 1, pp. 84–92, 2004, doi: https://doi.org/10.1016/S0924-0136(03)00865-3.

[23] Z. Jalilibal, A. Amiri, P. Castagliola, and M. B. C. Khoo, "Monitoring the coefficient of variation: A literature review," *Comput Ind Eng*, vol. 161, p. 107600, 2021.

[24] M. R. Simi, B. K. Bindhu, A. Varghese, and M. R. Rani, "Optimization of DRASTICA vulnerability assessment model by Wilcoxon rank sum non parametrical statistical test," *Mater Today Proc*, vol. 58, pp. 121–127, 2022.

[25] V. Vignesh, D. Pavithra, K. Dinakaran, and C. Thirumalai, "Data analysis using box and whisker plot for stationary shop analysis," in *2017 International Conference on Trends in Electronics and Informatics (ICEI)*, IEEE, 2017, pp. 1072–1076.

p-ISSN 2301-7988, e-ISSN 2581-0588
DOI : 10.32736/sisfokom.v13i3.2290, Copyright ©2024
Submitted : October 4, 2024, Revised : October 19, 2024, Accepted : October 21, 2024, Published : November 21, 2024