Published by :
http://www.ijert.org

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 14 Issue 09, September - 2025

Adaptive Genetic Algorithm for Hybrid Flow Shop Scheduling with Sequence-Dependent Setup Times: Bottleneck Effects and Decision-Making Insights

Ah Saad, Ak El-Kharbotly, Mm El-Beheiry
Department of Design & Production Engineering
Faculty of Engineering, Ain-Shams University
Cairo, Egypt

***Abstract*— This study investigates the hybrid flow shop scheduling problem with sequence-dependent setup times (HFS-SDST), a challenging variant with high industrial relevance. A genetic algorithm with heuristic evaluation (GAH) is proposed to minimize makespan, combining GA search with multiple dispatching rules to enhance stability and performance. The model is verified on benchmark instances without setup times and extended datasets with setup times, with results compared against lower bounds and established metaheuristics.**

Experiments show that GAH achieves optimal or near-optimal solutions in small and medium cases, while maintaining deviations below 1% in large-scale problems. Analysis reveals that problem size, number of stages, and bottleneck position are critical drivers of complexity, and that the effectiveness of priority rules depends on bottleneck location.

The main contributions are a robust GA framework for HFS-SDST, systematic insights into structural complexity, and adaptive rule selection guidelines that support practical scheduling in manufacturing systems.

***Keywords*— Hybrid flow shop scheduling (HFS); Sequence-dependent setup times (SDST); Genetic algorithm (GA); Makespan minimization, Bottleneck analysis; Manufacturing scheduling and decision support**

I. INTRODUCTION

Hybrid Flow Shop (HFS) scheduling is a critical problem in production and operations management, arising when jobs must pass through multiple sequential stages, each stage consisting of one or more parallel machines. Unlike classical flow shops, this structure combines job sequencing across stages with parallel processing flexibility, making scheduling both more versatile and significantly more complex.

HFS systems are widely applied in industries such as semiconductors, automotive, textiles, chemical processing, and food production, where efficient scheduling directly affects throughput, delivery reliability, and costs. The scheduling challenge is inherently NP-hard, since decisions must simultaneously determine both machine allocation and job sequencing across multiple stages, often under constraints such as machine availability, job precedence, or setup requirements.

The optimization objectives in HFS scheduling typically include minimizing makespan, total flow time, tardiness, or energy consumption. Constraints are often simplified by assumptions such as deterministic processing times or negligible transportation. Optimizing HFS schedules provides major benefits: reduced makespan improves throughput and responsiveness, while efficient resource allocation minimizes idle time, lowers costs, and enhances energy efficiency.

The HFS scheduling represents a highly relevant yet computationally demanding problem. Effective solutions are crucial for industries seeking to improve productivity, reduce costs, and achieve sustainable manufacturing outcomes.

II. NOMENCLATURE AND ABBREVIATIONS

A. Nomenclature

Symbol	Description
J	Number of jobs, j ϵ [1: J], i ϵ [1: J]
S	Number of stages, s∈ [1:S]
Ms	Number of machines at stage s, k∈ [1: Ms]
PTj,s	Processing time of job j at stage s
ATi,j,s	Adjusting time from job iii to job j on any machine at stage s
Xj,s,k,r	Binary variable = 1 if job j is processed on machine k at stage s in the rth order
JAj,s	Arrival time of job j at stage s
STj,s,k,r	Starting time of job j at stage s on machine k in the rth order
FTj,s,k,r	Finishing time of job j at stage s on machine k in the rth order
LTj,s	Leaving time of job j from stage s
ITk,s	Idle time of machine k at stage s
Cmax	Makespan, i.e., maximum finishing time
Ps	Processing load at stage s
$\bar{S}_s$	Mean pairwise setup time at stage s
$\hat{S}_s$	Estimated setup load at stage s
Ws	effective workload at stage s
Ls	Load per machine at stage s
LB	The lower bound

B. Abbreviations

Abbreviation	Description
AIS	Artificial immune system
AT	Adjusting (setup) time
BN	Bottleneck
Cmax	Makespan (maximum completion time)
ERP	Enterprise resource management
FCFS	First-Come First-Serve (dispatch rule)
FT	Finishing time
GA	Genetic Algorithm
GAH	Genetic Algorithm Heuristic (as used in document)
HFS	Hybrid Flow Shop
IAIS	immunoglobulin-based artificial immune system
IDBAC	improved discrete artificial bee colony algorithm
NP	Non-deterministic Polynomial (complexity class)
NSGA-II	non-dominated sorting genetic algorithm 2
INSGA-WHE	improved non-dominated sorting genetic algorithm with heuristic evaluation
LAT	least adjusting time
LT	leaving time
LB	Lower Bound
LPT	longest processing time
LWR	least work remaining
MWR	most work remaining
OX2	order crossover 2
PSO	Particle Swarm Optimization
SDST	Sequence-Dependent Setup Time
SOHFSSP	Single Objective Hybrid Flow Shop Scheduling Problem
SPT	Shortest Processing Time
ST	Starting Time
SVNDS	skewed Variable Neighborhood Descent Search

III. LITERATURE REVIEW

Hybrid Flow Shop (HFS) scheduling has been widely studied due to its prevalence in complex manufacturing systems with multiple stages and parallel machines. Given its NP-hard nature, researchers have addressed diverse objectives—including makespan, tardiness, energy, and cost—under constraints such as sequence-dependent setup times, machine eligibility, and limited buffers. Exact approaches (e.g., branch-and-bound, mixed integer programming) have been applied in early studies [1], while metaheuristics such as Genetic Algorithms (GA), Tabu Search, Simulated Annealing, Particle Swarm Optimization, Artificial Bee Colony, and Variable Neighborhood Search now dominate due to scalability.
Comprehensive surveys have mapped the field: [2] reviewed exact and heuristic approaches, [3] classified formulations, and [4] synthesized over 500 papers on scheduling with setup times and costs.
Recent applications highlight methodological variety studying HFS scheduling with no setup times: to minimize makespan, [5] used B&B, [6] used PSO, [7] used IDBAC, [8] used AIS and [9] used IAIS .[10] applied GA to minimize tardiness, [11] proposed priority-rule heuristics and Tabu Search for two-stage HFS, and [12] and [13] used variable neighborhood search for makespan minimization.
Special cases include no-wait/no-store models [14], multiprocessor tasks [15]; and distributed heterogeneous systems [17].
Setup times, especially sequence-dependent setup times (SDST)—have been a major focus in works such as [18], [19], [20], and [21], employing simulated annealing, cooperative GA, and artificial bee colony algorithms. Multi-objective studies remain limited, e.g., [22] on tardiness and setup, and [23] on robust energy-aware scheduling.
In studying the lower bound and bottleneck calculations [24] Introduced new lower bound techniques for flexible flow shop scheduling by estimating minimum idle time per stage (cumulative load per capacity). [25] Presented techniques to compute lower bounds on makespan in project scheduling by distributing work over resource capacities and using precedence propagation. The idea of dividing cumulative work by capacity parallels per-stage lower bound. A comprehensive classification of HFS variants, objectives, and solution methods were presented by [26], and extensive review of setup-time-related scheduling problems by [4].
[27] introduced lower bounds for makespan, highlighted the role of bottlenecks. [28] Discussed shifting bottlenecks—as the bottleneck positions can change dynamically due to variability in processing times—and emphasizes the need for adaptive release policies. Workload variance and bottleneck severity, [29] Showed through simulation that process-time variability causes bottlenecks to shift, and that workload control methods outperform conventional release rules. The unevenness (UE) is used as a complexity factor capturing workload imbalance's impact on difficulty. [30] Introduced methods to detect bottleneck stages in hybrid flow shops with parallel machines and quantified bottleneck severity.
Research on hybrid flow shop (HFS) scheduling covers a range of structures, from two-stage to distributed and multiprocessor systems, with setup times considered in only about one-third of studies and sequence-dependent setup times (SDST) rarely addressed. Genetic algorithms (GAs) are widely applied to HFS but seldom to HFS-SDST, where other metaheuristics dominate. Multi-objective optimization remains underexplored, and most studies metric emphasize algorithm development and benchmarking rather than examining how input parameters such as job size, number of stages, machine count, and bottleneck position affect problem complexity.
Two main research gaps emerge: first, the absence of GA-based approaches capable of effectively solving HFS-SDST problems across different scales; and second, the lack of systematic analysis on the role of input parameters, bottleneck structures and priority rules in shaping solution behavior.
This study develops a single-objective HFS-SDST model using an enhanced NSGA-II framework to minimize makespan with priority rules. The objectives are to benchmark the model against standard problems and theoretical bounds, introduce a single-score complexity, compare priority-rule performance across

varied configurations, and analyze how bottleneck severity, setup times, and other parameters influence solution quality.

The rest of the paper is organized as follows:

- Section IV: problem Description.
- Section V: proposed mathematical model formulation
- Section VI: solution methodology.
- Section VII: design of experiments
- Section VIII: results and discussion
- Section IX: conclusion

IV. PROBLEM DESCRIPTION

This study addresses a single-objective scheduling problem in a hybrid flow shop (HFS), where the objective is to minimize the makespan (Cmax) by determining the optimal job sequence at each stage while accounting for sequence-dependent setup times (SDST).

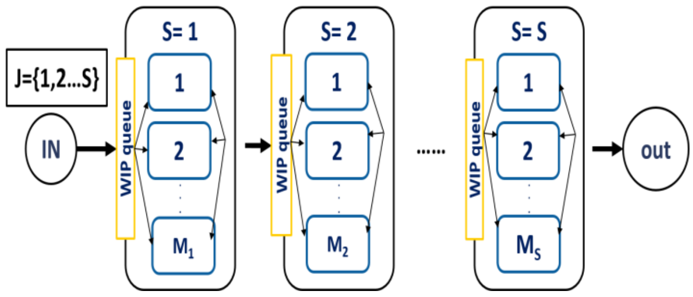


Figure 2 HFS structure

The hybrid flow shop considered Figure 2 consists of S stages, each containing a set of identical parallel machines (Ms) performing the same type of operation. A set of J jobs must be processed in a unidirectional flow, where at each stage a job j can be assigned to any available machine. The processing time of job j at stage s is denoted by PTj,s. Before processing, the machine requires adjustment, represented as a sequence-dependent setup time ATi,j,s when switching from job i to job j. All jobs are assumed to be available at time zero. Once a job completes processing at stage s, it is immediately transferred to stage s+1. If an idle machine exists at stage s+1, processing begins directly; otherwise, the job enters a waiting queue until a machine becomes available. In this case, the machine first undergoes the required setup before processing the next job. When multiple jobs are waiting, their processing order may vary across stages depending on machine availability and setup requirements. The scheduling problem is thus to determine a feasible job sequence across all stages that minimizes the makespan (Cmax).
In this work, a single objective scheduling problem in a Hybrid flow shop (HFS) is studied where the main objective is minimizing the makespan through providing a sequence of job assignment at each stage taking into consideration sequence dependent setup time.

V. MATHEMATICAL FORMULATION

The job cycle describes the events and durations that a job goes through from leaving a stage until going to the next stage. The timeline is divided into time periods which are bounded by certain events. Figure 1 job cycle time at one describes the job cycle on each stage.

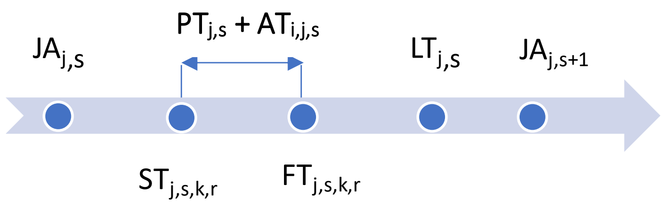


Figure 1 job cycle time at one stage.

First, all jobs are assumed to be available at 0 at stage 1 i.e., their JAJ,1=0. Each job waits until their assignment order is reached. ST is the starting time on the machine. The utilized time in setting and processing are ATi,j,s and PTj,s. the FT is when the job is finished on this stage then it leaves the stage at the leaving time LT and it is moved directly to the next stage at job arrival point JAj,2.

A. Assumptions

The following assumptions are considered in the proposed hybrid flow shop (HFS) model:

- All jobs are available and ready at time zero.
- Waiting between stages is allowed.
- Processing times are fixed at each stage and independent of machine assignment.
- Once a job is assigned to a machine, it cannot be reassigned within the same stage.
- Machine buffers are assumed unlimited; material handling and transfer times between machines are neglected.
- Each machine can process only one job at a time, and once started, a job cannot be interrupted until completion.
- All machines within a stage are identical, both in processing and setup requirements.
- Planned and unplanned downtimes are not considered.
- Energy consumption related to handling and transportation is neglected.
- Upon completion at stage s, each job immediately moves to stage s+1.

B. Proposed model formulation:

The objective is to minimize the makespan (Cmax) which is defined as the maximum completion time among all jobs:
Objective function

$$\min(C_{max}) \quad (1)$$

$$C_{max} = \max(FT_{j,s,k,r}) \quad (2)$$

$$ST_{j,s,k,r} = [\mathrm{MAX}(JA_{j,s,k,r}, FT_{j,s,k,r-1})] * X_{j,s,k,r} \quad (3)$$

$$FT_{j,s,k,r} = ST_{j,s,k,r} + [AT_{i,j,s} + PT_{j,s}] * X_{j,s,k,r} \quad (4)$$

where STj,s,k,r is the start time, JAj,s is the arrival time of job j at stage s, and ATi,j,s is the sequence-dependent setup time from job i to job j.
And, FTj,s,k,r is the finishing time of job j at stage s on machine k in order r.

The model is subject to the following constraints:

1. Each job is assigned a unique order on a machine at each stage:

$$\sum_{j=1}^{J} X_{j,s,k,r} \leq 1 \qquad \forall\, s, k, r \quad (5)$$

2. Each job is processed only once per stage:

$$\sum_{r=1}^{R} \sum_{k=1}^{Ms} X_{j,s,k,r} \leq 1 \qquad \forall\, j, s \quad (6)$$

3. A job cannot start at a stage until it has completed the previous stage:

$$ST_{j,s,k,r} \geq FT_{j,s-1,k',r'} \quad \forall\, j, s \quad (7)$$

4. A machine cannot start a new job until it has finished the preceding job:

$$ST_{j,s,k,r} \geq FT_{j',s,k,r-1} \quad \forall\, j, s, k \quad (8)$$

5. All jobs are initially available at stage 1:

$$JA_{j,1} = 0 \qquad \forall\, j \quad (9)$$

Equations (1)– (2) define the objective of minimizing the makespan. Equations (3)– (4) capture the time relations, integrating both processing and sequence-dependent setup times. Constraints (5)– (6) ensure valid job assignments, while constraints (7)– (8) enforce technological precedence and machine availability. Finally, constraint (9) sets the initial condition of the problem.

VI. SOLUTION METHODOLOGY

The proposed problem, a single-objective hybrid flow shop scheduling problem with sequence-dependent setup times (SOHFSSP), is solved using an Improved NSGA-II with Weighted Heuristic Evaluation (INSGA-WHE). The algorithm is implemented in MATLAB to minimize the makespan under the defined constraints.
Each solution is represented as a chromosome encoding a permutation of jobs at the first stage. Machines follow a dispatching rule whereby the first idle machine processes the next job in the sequence. From the second stage onward, job assignment is determined by one of several heuristics: FCFS, SPT, LPT, LWR, MWR, or LAT. Each heuristic is applied separately across all stages, and the makespan is calculated. The smallest Cmax among all heuristics represents the fitness of the chromosome.
The algorithm begins with a randomly generated initial population. Fitness evaluation simulates job progression across stages, calculating start times, finish times, and leaving times based on machine availability and the selected heuristic. Elite solutions are preserved across generations to prevent loss of high-quality individuals. New offspring are generated using Syswerda's two-point order crossover (OX2), which preserves partial job sequences from each parent. To maintain diversity and avoid premature convergence, a two-step mutation is applied: inversion mutation, followed by a random swap mutation.
The iterative process of selection, crossover, mutation, and evaluation continues until termination criteria are met, either a fixed number of generations or convergence without improvement. By integrating NSGA-II's population-based global search with heuristic-based local evaluation, the INSGA-WHE balances exploration and exploitation, yielding robust solutions to the SOHFSSP.

VII. DESIGN OF EXPERIMENTS:

A structured set of experiments was conducted to verify the proposed optimization model and the GA-based algorithm (INSGA-WHE, hereafter referred to as GAH) for solving the hybrid flow shop scheduling problem. The experiments were designed as a full factorial scheme to investigate the influence of key problem parameters, summarized in Table 1.
The benchmark instances introduced by [5] have been widely applied in the literature for single-objective HFS problems. Since these instances do not incorporate setup times, verification was performed through two complementary approaches. In the first set of experiments, the proposed model was adjusted to solve benchmark instances without setting up times, allowing direct comparison with published results. In the second set, the benchmark datasets were extended by introducing process-dependent setup times through incidence matrices for each job and stage. In this case, the solutions were evaluated against an algebraically derived lower bound (LB) of the makespan, ensuring robustness of the proposed model. For all Carlier and Néron instances, the LB provides either the exact optimum (when reached) or a reference for measuring deviation from the optimal solution.
The naming convention of benchmark problems follows the format "j10c5a01," where "j10" denotes the number of jobs (10 in this case), "c5" denotes the number of stages (5), and the letter indicates machine allocation: (a) all stages have three machines except the last, which has one; (b) all stages have three machines except the first, which has one; (c) all stages have three machines except the middle, which has one; and (d) all stages have variable number of machines.
[30] classified these benchmarks into easy and hard problems, as summarized in Table 2. Easy problems include instances up to 10 jobs and 10 stages, where optimal solutions are typically obtained in under 1600 seconds. Hard problems consist of larger instances with 50–250 jobs and require greater computational effort. All [5] instances used in the verification and the modified instances by adding SDST are in this link: HFS test instances single objective carlier and neron and modified instances with SDST.rar

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 14 Issue 09, September - 2025

Table 1 Parameter values used in experimentation

Parameter	Values
Number of jobs (J)	10, 15, 30, 100
Number of stages (S)	5, 10
Number of parallel machines (Ms)	1, 2, 3
Bottleneck position	First stage, Middle stage, Last stage
Priority rules	FCFS, SPT, LAT, MWR, LWR

Table 2 Reference testing parameters (Néron et al., 2001)

Parameter	Easy problems	Hard problems
Number of jobs	10, 15	50, 100, 250
Population size	50	100
Crossover ratio	0.75	0.75
Elite solution (%)	0.04	0.04
Maximum generations	1000	3000
Maximum stall generations	100	500

The lower bound is calculated using the following procedure.

$$P_s = \sum_{j=1}^{J} PT_j \tag{10}$$

$$\overline{S}_s = mean\big(\min(AT_{i,j,s})\big)\ \forall\ i \in \{1{:}J\} \tag{11}$$

$$\hat{S}_s = (J - M_s) * \overline{S}_s \tag{12}$$

$$W_s = P_s + \hat{S}_s \tag{13}$$

$$L_s = \frac{W_s}{M_s} \tag{14}$$

$$L_{bn} = Max(L_s) \tag{15}$$

$$LB = L_{bn} + Min\left(\sum_{s=1}^{bn-1} PT_j\right) + Min\left(\sum_{s=bn+1}^{S} PT_j\right) \tag{16}$$

Where the processing load (Ps) in (10) is the total processing time at stage s. the mean pairwise setup time at stage s (11) ($\overline{S_s}$). The estimated setup load at stage s ($\hat{S}_s$) is described in (12). (13) is the effective workload which is the sum of (10) and (12) at stage s. (14) describes the normalized load per machine at each stage. The normalized load at the bottleneck stage is determined in (15) where bn is the bottleneck stage number. (16) calculates the lower bound of the instance.
The lower bound for problems with setup times was calculated using equations (10)–(16), where the total processing load, mean pairwise setup times, and effective workload at each stage were combined to derive the normalized load per machine and, ultimately, the bottleneck stage lower bound. A total of 109 extended instances incorporating sequence-dependent setup times were solved using this approach.
For preliminary testing, five hard problems with 10 and 15 jobs were used to tune the GA parameters. Population size (Ps) varied from 40 to 100, and crossover percentage (ox%) was tested between 0.6 and 0.8. The best performance was obtained with Ps=100 and ox=0.75 for hard problems, while Ps=40 was sufficient for smaller instances.
All experiments were implemented in MATLAB and executed on a Ryzen 7 5800 (3.2 GHz) processor with 24 GB RAM. The analysis examines how job size, number of stages, machine allocation, and bottleneck position influence solution quality. Additionally, the performance of different priority rules in estimating (Cmax) is compared across varied problem configurations.

VIII. RESULTS AND DISCUSSION

A. Verification with Benchmark Instances (Without Setup Times)

To establish validity, the proposed GAH algorithm was first tested on standard benchmark instances from [5] without setup times. The results were compared with three widely used optimization methods: PSO, IDABC, and SVNDS.
Representative outcomes are shown in different tables in a spread sheet covering small- to large-scale problems in the link: results summary.xlsx
For small problems (10 jobs, 5 stage), all algorithms, including GAH, consistently reached the lower bound (LB) with zero deviation. Similar behavior was observed for 10-job, 10-stage problems, except for a few complex instances where deviations of 1–18% occurred. In these harder cases, SVNDS demonstrated a slight advantage due to its adaptive queue rearrangement strategy.
The overall performance across instances is summarized in Table 3 and Table 4, showing mean deviations and deviation ranges. For small and medium sizes, GAH achieved results comparable to or better than existing algorithms. In large problems (30 jobs, 5 stages), mean deviations remained below 3% across all methods, with GAH slightly outperforming IDABC but marginally trailing SVNDS in average deviation.
For large-scale problems of 100 jobs and 5 stages, where no exact LB is available, deviations were computed relative to the best-known solutions. As reported in

, GAH achieved a maximum deviation of only 0.041% and a mean deviation of 0.007%, consistently within a narrow range. This indicates strong scalability and reliability of the proposed approach.
Figure 3 further illustrates the comparison between GAH and SVNDS for 100-job instances. While SVNDS achieves marginally lower mean deviation, it also exhibits a wider deviation range, suggesting higher variability. By contrast, GAH provides more stable performance, though at the expense

of slightly longer computational times. This is attributed to the evaluation of multiple priority rules within the GA framework, which increases robustness but extends runtime. The superior performance of SVNDS in certain cases can be explained by its deliberate wait strategy, which increases search space diversity by allowing controlled machine idleness.

Table 3 mean % deviation comparison between different models

instance size		Mean % deviation			
J	**S**	**PSO** [31]	**IDABC** [7]	**SVNDS** [13]	**GAH**
10	**5**	0.00%	0.08%	0.00%	0.00%
10	**10**	2.78%	2.88%	2.68%	2.77%
15	**5**	0.00%	0.00%	0.00%	0.00%
30	**5**	3.09%	1.10%	1.08%	2.39%

Table 4 range of deviation for different models at different problem sizes

instance size		Deviation Range			
J	**S**	**PSO** [31]	**IDABC** [7]	**SVNDS** [13]	**GAH**
10	**5**	0.000	0.014	0.000	**0.000**
10	**10**	18.368	18.368	18.368	**18.368**
15	**5**	0.000	0.000	0.000	**0.000**
30	**5**	0.126	0.105	0.105	**0.077**

Table 5 summary of deviation results solving 100 jobs sized problems

	% deviation			
	AIS [8]	**IAIS** [9]	**SVNDS** [13]	**GAH**
max	0.454	0.026	0.000	0.041
min	0.002	0.000	0.000	0.000
mean	0.201	0.006	0.000	0.007

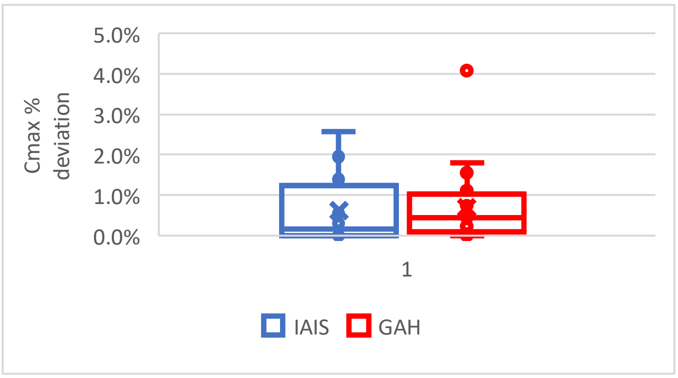


Figure 3 Cmax % deviation comparison between models solving 100 jobs instances.

B. Performance with Sequence-Dependent Setup Times

To assess the model under more realistic conditions, 60 benchmark instances were extended with sequence-dependent setup times (AT) and solved using GAH. Deviations were calculated against algebraically derived lower bounds.

The results revealed deviations ranging from 1% to 59%, depending on problem scale and bottleneck location. Figure 4 shows that for small problems (10 jobs, 5–10 stages), deviations decrease when the bottleneck is shifted to later stages. This occurs because early-stage bottlenecks create queues that propagate downstream, inflating makespan, whereas later-stage bottlenecks allow earlier stages to complete work in parallel, thus buffering the impact.

In contrast, this trend does not hold for larger instances (15–30 jobs; Figure 5), where deviations peaked at middle-stage bottlenecks. This can be explained by congestion in the middle of the system: jobs accumulate at intermediate stages, where frequent setup changes amplify delays while downstream stages remain underutilized. This imbalance increases makespan relative to the lower bound.

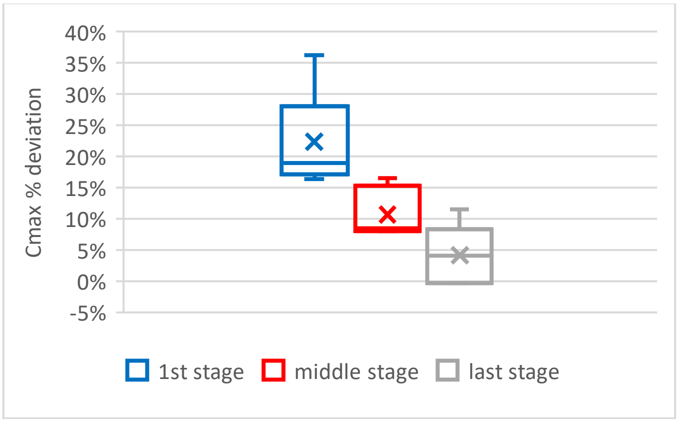


Figure 4 Cmax % deviation from LB for 10 job 5 & 10 stage problems

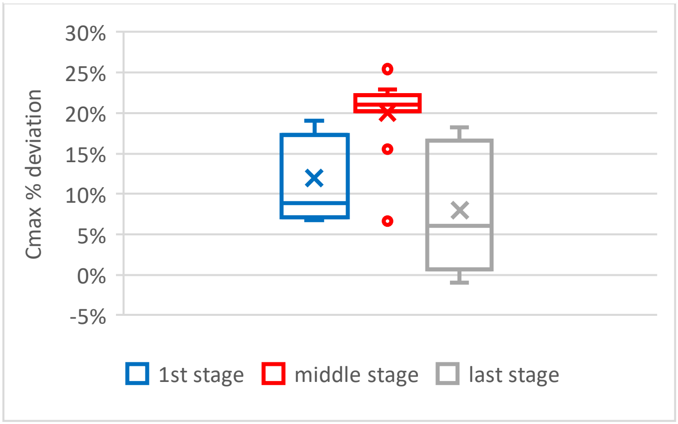


Figure 5 Cmax % deviation from LB for 15 & 30 job 5 stage problems

C. Influence of Problem Parameters

1) Effect of Number of Jobs

As shown in Figure 6 and Figure 7, both mean deviation and deviation range increase with the number of jobs, particularly beyond 15 jobs. This reflects the factorial growth of the solution space (J) as job count rises, making optimal sequences harder to identify. The larger number of jobs also increases the total number of setup transitions, magnifying the effect of SDST.

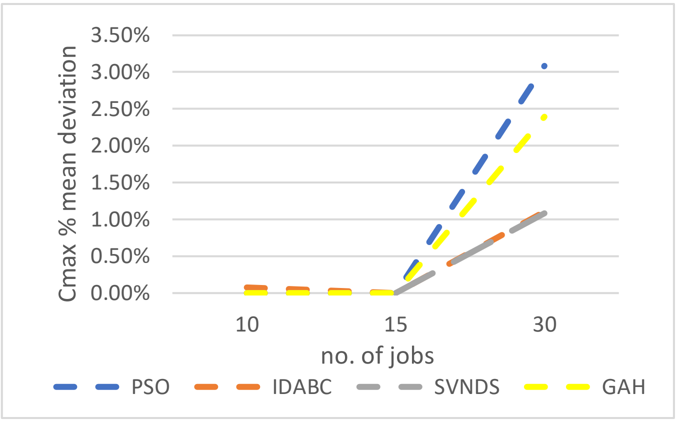


Figure 6 Cmax mean deviation at different number of jobs

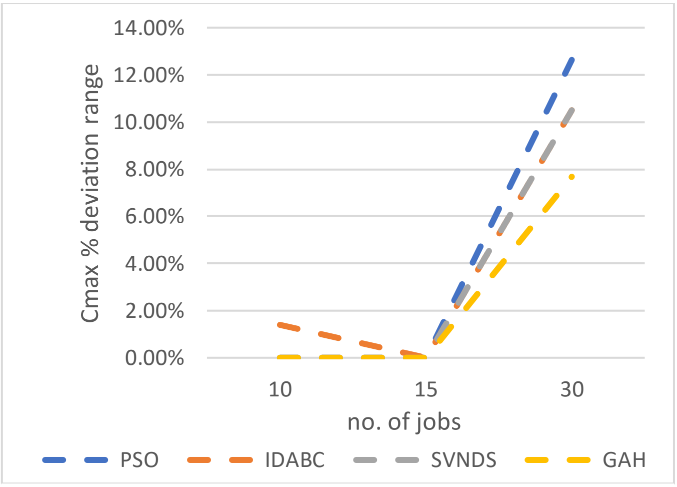


Figure 7 Cmax deviation range at different number of jobs

2) Effect of Number of Stages

Figure 8 and Figure 9 illustrates the impact of increasing the number of stages. For 10 jobs, deviations were negligible with fewer stages but rose sharply when the number of stages reached 10. This is because each additional stage multiplies the number of sequencing and setup interactions, while also increasing the likelihood of stage imbalance, where some stages complete early and others become bottlenecks.

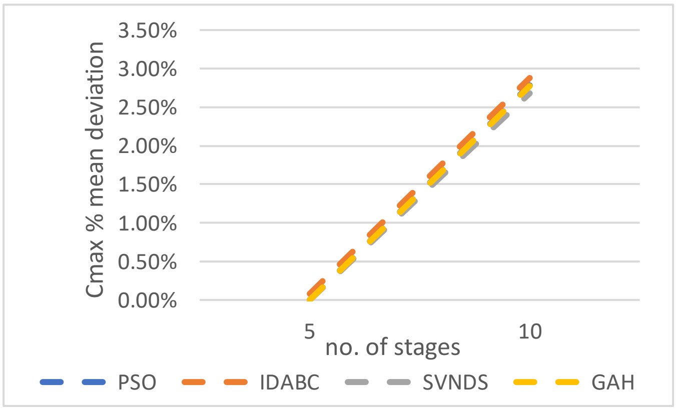


Figure 8 Cmax mean deviation at different number of stages

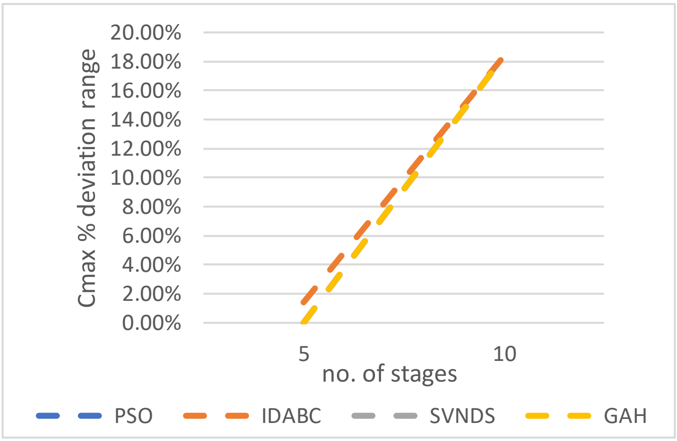


Figure 9 Cmax deviation range at different number of stages

3) Effect of Bottleneck Position

Bottleneck location strongly influences solution quality. Figure 4, Figure 5 and **Error! Reference source not found.** consistently show that advancing the bottleneck toward later stages reduces makespan deviation. When the bottleneck is located at the first stage, waiting times are created immediately, and delays propagate through all downstream stages. In contrast, when the bottleneck occurs at the last stage, earlier stages can process freely and build up buffers, reducing systemic disruption. Middle-stage bottlenecks are particularly problematic in larger problems because they simultaneously block upstream and downstream flows, amplifying the effect of SDST.

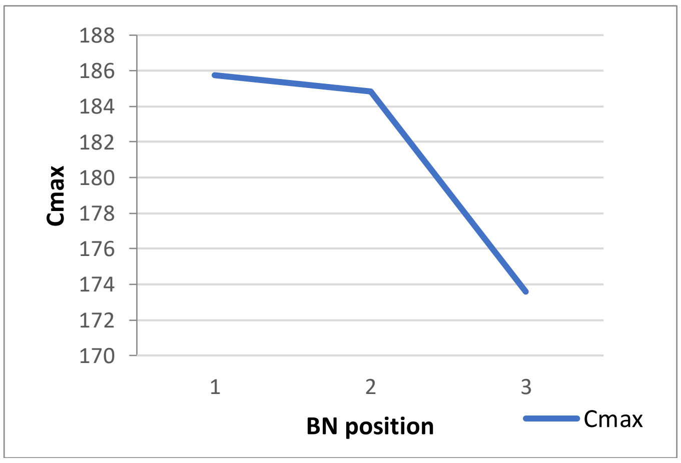
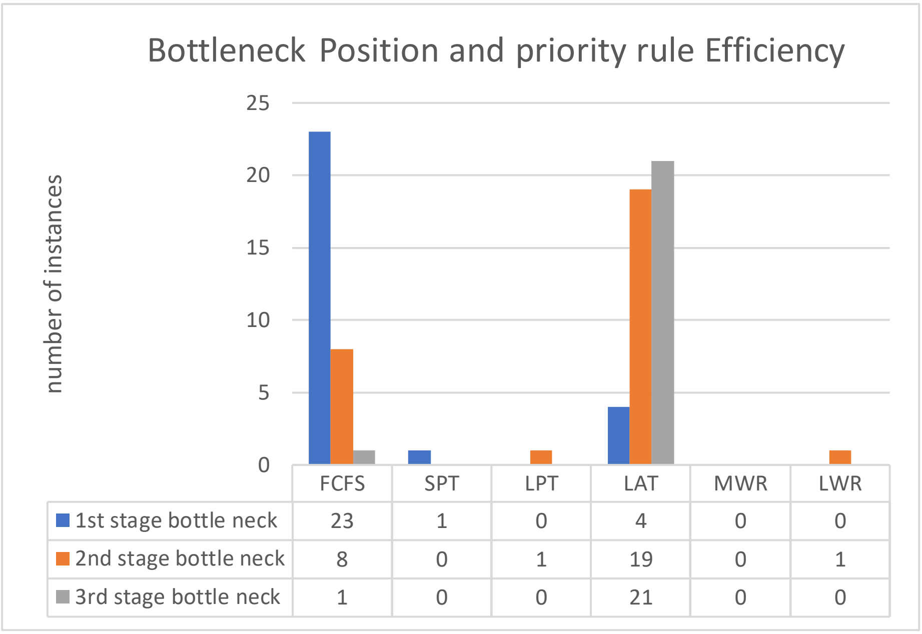


Figure 10 the change in the Cmax with the deviation of BN to later stages

4) Role of Priority Rules

Priority rules embedded in the GA were also analyzed. Figure 11 shows that FCFS and LAT rules most frequently produced the best solutions. FCFS performs best when the bottleneck is located at the first stage, as it reduces queue build-up and stabilizes downstream flow. LAT becomes more effective when bottlenecks occur at middle or final stages, since it prioritizes urgent jobs and reduces delays closer to completion. These findings suggest that dispatching rules should not be static but adaptive, selected based on the position of the bottleneck.



	FCFS	SPT	LPT	LAT	MWR	LWR
■ 1st stage bottle neck	23	1	0	4	0	0
■ 2nd stage bottle neck	8	0	1	19	0	1
■ 3rd stage bottle neck	1	0	0	21	0	0

Figure 11 no. of instances solved by each priority rule at different bottle neck position

IX. CONCLUSIONS

This work developed and validated a single-objective hybrid flow shop scheduling (HFS) model with sequence-dependent setup times (SDST), solved using a genetic algorithm with heuristic evaluation (GAH) to minimize makespan. The algorithm was verified through benchmark comparisons for small- to medium-scale instances (up to 30 jobs and 10 stages) and against algebraically derived lower bounds for larger instances with 100 jobs. In both cases, deviations from optimal or reference values remained negligible, demonstrating that the proposed method consistently achieves optimal or near-optimal solutions across different problem scales.

The results highlight that while problem size and the number of stages increase computational complexity, bottleneck position and machine allocation exert a stronger influence on deviations from the lower bound. Severe bottlenecks, particularly in early or middle stages, caused greater variability, underscoring the structural sensitivity of the problem. Analysis of priority rules further revealed that FCFS performs best when the bottleneck occurs at the first stage, whereas LAT provides superior performance when bottlenecks arise in later stages. These findings suggest that adaptive rule selection based on bottleneck location can significantly improve scheduling outcomes.

The novelty of this study lies in applying a GA-based framework with heuristic evaluation to effectively solve HFS-SDST across both small- and large-scale problems, while systematically analyzing how structural parameters influence solution behavior—an aspect largely overlooked in prior research. The contributions are twofold: methodologically, the GAH algorithm integrates GA search with multi-rule evaluation to deliver robust and stable solutions; practically, the analysis provides decision makers with actionable insights. In particular, the ability to identify bottleneck stages and adapt priority rules in real time supports proactive scheduling adjustments, improves responsiveness, and reduces lead time in manufacturing systems.

Overall, this study not only benchmarks the effectiveness of the proposed GA approach but also advances understanding of structural complexity in hybrid flow shops, offering both theoretical insights and practical decision-support tools.

REFRENCES

[1] J. N. D. Gupta, “Two-Stage, Hybrid Flowshop Scheduling Problem,” Journal of the Operational Research Society, vol. 39, no. 4, pp. 359–3641, 1988, doi: 10.1057/JORS.1988.63.

[2] J. N. D. Gupta, A. M. A. Hariri, and C. N. Potts, “Scheduling a two-stage hybrid flow shop with parallel machines at the first stage,” Ann Oper Res, vol. 69, pp. 171–191, 1997, doi: 10.1023/A:1018976827443.

[3] Jacek. Błażewicz, “Scheduling : Theory, Algorithms, and Systems / M. Pinedo,” p. 485, 2001, Accessed: Aug. 19, 2025. [Online]. Available: https://www.academia.edu/45241856/Scheduling_Theory_Algorithms_and_Systems_M_Pinedo

[4] A. Allahverdi, “The third comprehensive survey on scheduling problems with setup times/costs,” Eur J Oper Res, vol. 246, no. 2, pp. 345–378, Oct. 2015, doi: 10.1016/J.EJOR.2015.04.004.

[5] J. Carlier and E. Néron, “An Exact Method for Solving the Multi-Processor Flow-Shop,” RAIRO - Operations Research, vol. 34, no. 1, pp. 1–25, Jan. 2000, doi: 10.1051/RO:2000103.

[6] C. J. Liao, E. Tjandradjaja, and T. P. Chung, “An approach using particle swarm optimization and bottleneck heuristic to solve hybrid flow shop scheduling problem,” Appl Soft Comput, vol. 12, no. 6, pp. 1755–1764, Jun. 2012, doi: 10.1016/J.ASOC.2012.01.011.

[7] Z. Cui and X. Gu, “An improved discrete artificial bee colony algorithm to minimize the makespan on hybrid flow shop problems,” Neurocomputing, vol. 148, pp. 248–259, Jan. 2015, doi: 10.1016/J.NEUCOM.2013.07.056.

[8] O. Engin and A. Döyen, “A new approach to solve hybrid flow shop scheduling problems by artificial immune system,” Future Generation Computer Systems, vol. 20, no. 6, pp. 1083–1095, Aug. 2004, doi: 10.1016/J.FUTURE.2004.03.014.

[9] T. P. Chung and C. J. Liao, “An immunoglobulin-based artificial immune system for solving the hybrid flow shop problem,” Applied Soft Computing Journal, vol. 13, no. 8, pp. 3729–3736, 2013, doi: 10.1016/J.ASOC.2013.03.006.

[10] C. Yu, Q. Semeraro, and A. Matta, "A genetic algorithm for the hybrid flow shop scheduling with unrelated machines and machine eligibility," Comput Oper Res, vol. 100, pp. 211–229, Dec. 2018, doi: 10.1016/J.COR.2018.07.025.

[11] M. K. Hajji, H. Hadda, and N. Dridi, "Makespan Minimization for the Two-Stage Hybrid Flow Shop Problem with Dedicated Machines: A Comprehensive Study of Exact and Heuristic Approaches," Computation 2023, Vol. 11, Page 137, vol. 11, no. 7, p. 137, Jul. 2023, doi: 10.3390/COMPUTATION11070137.

[12] J. Q. Li, Q. K. Pan, and F. T. Wang, "A hybrid variable neighborhood search for solving the hybrid flow shop scheduling problem," Appl Soft Comput, vol. 24, pp. 63–77, Nov. 2014, doi: 10.1016/J.ASOC.2014.07.005.

[13] F. Vidojević, A. Džamić, D. Džamić, and M. Marić, "A novel artificial intelligence search algorithm and mathematical model for the hybrid flow shop scheduling problem," J Big Data, vol. 12, no. 1, pp. 1–24, Dec. 2025, doi: 10.1186/S40537-025-01085-X/TABLES/5.

[14] J. Grabowski and J. Pempera, "Sequencing of jobs in some production system," Eur J Oper Res, vol. 125, no. 3, pp. 535–550, Sep. 2000, doi: 10.1016/S0377-2217(99)00224-6.

[15] C. Oĝuz and M. F. Ercan, "A genetic algorithm for hybrid flow-shop scheduling with multiprocessor tasks," Journal of Scheduling, vol. 8, no. 4, pp. 323–351, Jul. 2005, doi: 10.1007/S10951-005-1640-Y/METRICS.

[16] H. Cui, X. Li, and L. Gao, "An improved multi-population genetic algorithm with a greedy job insertion inter-factory neighborhood structure for distributed heterogeneous hybrid flow shop scheduling problem," Expert Syst Appl, vol. 222, p. 119805, Jul. 2023, doi: 10.1016/J.ESWA.2023.119805.

[17] M. E. Kurz and R. G. Askin, "Scheduling flexible flow lines with sequence-dependent setup times," Eur J Oper Res, vol. 159, no. 1, pp. 66–82, Nov. 2004, doi: 10.1016/S0377-2217(03)00401-6.

[18] H. S. Mirsanei, M. Zandieh, M. J. Moayed, and M. R. Khabbazi, "A simulated annealing algorithm approach to hybrid flow shop scheduling with sequence-dependent setup times," J Intell Manuf, vol. 22, no. 6, pp. 965–978, Dec. 2011, doi: 10.1007/S10845-009-0373-8/METRICS.

[19] Q. Zheng, Y. Zhang, H. Tian, and L. He, "A cooperative adaptive genetic algorithm for reentrant hybrid flow shop scheduling with sequence-dependent setup time and limited buffers," Complex and Intelligent Systems, vol. 10, no. 1, pp. 781–809, Feb. 2024, doi: 10.1007/S40747-023-01147-8/FIGURES/18.

[20] H. Xue, L. Meng, P. Duan, B. Zhang, W. Zou, and H. Sang, "2024 Growing Science Ltd," International Journal of Industrial Engineering Computations, vol. 15, pp. 473–490, 2024, doi: 10.5267/j.ijiec.2024.1.001.

[21] H. Tian, K. Li, and W. Liu, "A Pareto-Based Adaptive Variable Neighborhood Search for Biobjective Hybrid Flow Shop Scheduling Problem with Sequence-Dependent Setup Time," Math Probl Eng, vol. 2016, no. 1, p. 1257060, Jan. 2016, doi: 10.1155/2016/1257060.

[22] Y. Wang and J. Su, "A data-driven robust optimization method based on scenario clustering for PVC production scheduling under uncertainty," Comput Chem Eng, vol. 188, Sep. 2024, doi: 10.1016/J.COMPCHEMENG.2024.108782.

[23] L. Hidri, "Effective Two-Phase Heuristic and Lower Bounds for Multi-Stage Flexible Flow Shop Scheduling Problem with Unloading Times," Symmetry (Basel), vol. 15, no. 11, Nov. 2023, doi: 10.3390/SYM15112005.

[24] D.-W. Jens and S. Berlin, "Hybrid Solving Techniques for Project Scheduling Problems".

[25] R. Ruiz and J. A. Vázquez-Rodríguez, "The hybrid flow shop scheduling problem," Eur J Oper Res, vol. 205, no. 1, pp. 1–18, Aug. 2010, doi: 10.1016/J.EJOR.2009.09.024.

[26] M. L. Pinedo, "Scheduling: Theory, Algorithms, and Systems, Sixth Edition," Scheduling: Theory, Algorithms, and Systems, Sixth Edition, pp. 1–698, Jan. 2022, doi: 10.1007/978-3-031-05921-6/COVER.

[27] M. Thürer and M. Stevenson, "Bottleneck-oriented order release with shifting bottlenecks: An assessment by simulation," Int J Prod Econ, vol. 197, pp. 275–282, Mar. 2018, doi: 10.1016/J.IJPE.2018.01.010.

[28] A. Prabhu, K. Raghunandana, and Y. P. Pai, "Workload Control in Flow Shops with Bottleneck Shifting and Process Time Variability," Production Engineering Archives, vol. 30, no. 1, pp. 48–56, Mar. 2024, doi: 10.30657/PEA.2024.30.4.

[29] E. Nowicki and C. Smutnicki, "The flow shop with parallel machines: A tabu search approach," Eur J Oper Res, vol. 106, no. 2–3, pp. 226–253, Apr. 1998, doi: 10.1016/S0377-2217(97)00260-9.

[30] E. Néron, P. Baptiste, and J. N. D. Gupta, "Solving hybrid flow shop problem using energetic reasoning and global operations," Omega (Westport), vol. 29, no. 6, pp. 501–511, Dec. 2001, doi: 10.1016/S0305-0483(01)00040-8.

[31] C. J. Liao, E. Tjandradjaja, and T. P. Chung, "An approach using particle swarm optimization and bottleneck heuristic to solve hybrid flow shop scheduling problem," Appl Soft Comput, vol. 12, no. 6, pp. 1755–1764, Jun. 2012, doi: 10.1016/J.ASOC.2012.01.011.