International Journal of Production Management and Engineering

https://doi.org/10.4995/ijpme.2025.24136

Received: 2025-06-16 Accepted: 2026-01-19
Available online: 2026-01-31 Published: 2026-01-31

Combinatorial framework for reducing tardiness in multi-machine scheduling using EDD, NEH and genetic algorithm

Prasad Bari [a1*], Nilaj Deshmukh [a2], Pradeep Yadav [b1], Prasad Karande [b2], Poonam Bari [a3]

[a] Fr. C. Rodrigues Institute of Technology, 400703, Vashi, Navi-Mumbai, India.
[b] Veermata Jijabai Technological Institute, 400019, Mumbai, India.

[a1] prasad.bari@fcrit.ac.in; [a2] nilaj.deshmukh@fcrit.ac.in; [b1] pradeeps.yadav1990@gmail.com;
[b2] pmkarande@me.vjti.ac.in; [a3] poonam.bari@fcrit.ac.in

Abstract:

No-idle flow shop scheduling is a critical challenge in manufacturing, where minimising overall tardiness directly impacts efficiency and customer satisfaction. This study introduces a novel hybrid algorithm, EDD-NEH-GA (ENG), which integrates Earliest Due Date (EDD) and Nawaz-Enscore-Ham (NEH) heuristics with a Genetic Algorithm (GA) to balance global exploration and local optimisation. The objective is to overcome premature convergence and achieve superior tardiness reduction. Computational experiments on Taillard's benchmark instances demonstrate ENG's effectiveness compared to the Mixed Integer Linear Programming (MILP) based approach by Balogh. Across all tested cases, ENG updated 93% of previously best-known solutions, achieving an average tardiness reduction of 18–52%. These results confirm ENG as a robust and efficient solution for complex no-idle flow shop environments, offering significant gains in scheduling performance and operational productivity. After performing statistical analysis it is noted that ENG outperforms. ENG significantly improved scheduling performance, averaging 8430 units of reduction in overall tardiness. According to the standard error of the difference (SE = 1859), this improvement appears to be constant across cases.

Key words:

Genetic algorithm, tardiness, scheduling, combinatorial approach, NEH.

1. Introduction

Scheduling in manufacturing systems, particularly shop scheduling, is a fundamental yet complex optimisation problem within the domain of operations research (Sobeyko & Mönch, 2016). In a typical production environment, multiple jobs must be processed across a series of machines, where each individual processing step is termed an operation. The specific order in which machines execute these operations defines the production (Mei et al., 2017; Nip et al., 2016). The primary objective of scheduling is to determine an optimal sequence of operations that achieves predefined performance goals (Zhang et al., 2019). Attaining an optimal schedule is crucial for enhancing productivity and operational efficiency, yet many shop scheduling problems are classified as NP-hard, highlighting their computational intractability.

To evaluate scheduling performance, researchers have adopted a variety of objective functions. Among these, makespan—the total time required to complete all jobs—is one of the most widely used metrics. Other important performance indicators include the total machine workload, maximum machine

To cite this article: Bari, P., Deshmukh, N., Yadav, P., Karande, P., & Bari, P. (2026). Combinatorial framework for reducing tardiness in multi-machine scheduling using EDD, NEH and genetic algorithm. *International Journal of Production Management and Engineering, 14*(1), e24136. https://doi.org/10.4995/ijpme.2026.24136

workload, maximum lateness, mean flow time, and various forms of tardiness (including total, mean, and weighted tardiness). Tardiness-based metrics capture the deviation between job completion times and their respective due dates, often reflecting real-world penalties or costs for delayed deliveries. A comprehensive classification of shop layout systems, their corresponding scheduling problems, and the solution methodologies employed can be found in the work of Mirmozaffari et al. (2024). In general, solution techniques fall into two broad categories: exact and approximate methods. Exact methods, including dynamic and linear programming models (Kemmoé et al., 2015), mixed-integer formulations (Sawik, 2000), and various Branch-and-X strategies such as Branch and Bound, Branch and Cut, Branch and Price (Jourdan et al., 2009), guarantee optimal solutions. However, their applicability is often limited to small- or medium-sized problem instances due to their high computational demands (Martí & Reinelt, 2011). In contrast, approximate methods are more scalable and practical for large-scale, NP-hard problems. These methods include metaheuristics, which employ higher-level strategies to guide the search process (Bianchi et al., 2009); constructive heuristics, such as the NEH algorithm, which iteratively build a solution based on priority rules (Framinan et al., 2002); and improvement heuristics, including local search and the shifting bottleneck procedure (Afsar et al., 2016; Choo et al., 2016; Lunardi et al., 2021; Mencía et al., 2015; Shi et al., 2020). Although approximate methods do not always guarantee optimality, they are capable of producing high-quality solutions within acceptable time frames making them especially valuable in real-world manufacturing contexts (Williamson & Shmoys, 2011). Given the complexity of shop scheduling problems and the limitations of exact methods for large-scale applications, recent research continues to focus on developing robust and efficient heuristic and metaheuristic algorithms. These approaches aim to balance solution quality and computational efficiency for total tardiness minimisation. In practical applications, minimising makespan and total tardiness are two of the most critical and frequently studied objectives. While makespan reduction has been extensively addressed in the literature, it often neglects due date constraints, potentially leading to delayed job completions. Therefore, this research considers total tardiness objective with the operational realities of modern manufacturing systems, where on-time delivery is paramount.

2. Literature review

The shop scheduling problem has evolved considerably over the past decades, with growing interest in specialized variants that incorporate realistic production constraints. One such variant is the No-idle Permutation Flowshop Scheduling Problem (NPFSP), which is particularly relevant in continuous production systems where interruptions between job operations are not feasible. The NPFSP imposes two strict constraints: (1) all machines must process jobs in the same sequence, and (2) no idle time is allowed on any machine once it begins processing jobs. These constraints reflect practical settings such as chemical processing or steel manufacturing, where machines must operate continuously and synchronously.

The NPFSP was first formally introduced by Adiri and Pohoryles (1982), who examined a two-machine scenario with the objective of minimising the total completion time. Their work laid the foundation for subsequent research that expanded the problem to more machines and alternative objective functions. For two machines, the problem is solvable in polynomial time; however, when extended to three or more machines, the problem becomes NP-hard (Baptiste & Hguny, 1997), making exact solutions computationally intensive. Over time, various approaches have been developed to tackle NPFSPs. Exact algorithms, such as branch-and-bound methods, were initially applied to small instances. However, these methods quickly became impractical for larger problems due to exponential growth in computational effort. Some progress was made in identifying polynomially solvable subcases using machine dominance properties (Wang & Xia, 2005) but the general problem remains intractable for large-scale instances. Due to the limitations of exact approaches, the focus has shifted to heuristic and metaheuristic methods, which can efficiently produce near-optimal solutions. Tasgetiren et al. (2010, 2013) proposed early metaheuristics for the NPFSP, targeting objectives such as makespan and total tardiness. These methods emphasized efficiency and scalability, although they lacked problem-specific tuning or hybridization with local search. Later developments introduced more refined hybrid strategies. For instance, Shen et al. (2015) presented a bi-population-based algorithm that combined a population-based global search with a local search mechanism rooted in one-insertion neighborhood

e-ISSN: 2340-4876

exploration. This method demonstrated significant improvements over earlier heuristics, particularly in solution quality and convergence speed.

Benchmark datasets have played a critical role in evaluating algorithm performance. Notably, Taillard's set (Taillard, 1993) and Ruiz's set (Ruiz et al., 2009) became standard references for experimental studies. On these benchmarks, Shao et al. (2018) developed a hybrid discrete teaching–learning-based optimisation algorithm, which further outperformed prior methods by effectively balancing exploration and exploitation. A more recent contribution by Riahi et al. (2020) introduced an iterated greedy algorithm incorporating two novel features: a destruction–construction phase and a refined insertion-based local search. This method achieved superior results on Taillard's benchmark, delivering improved solutions for approximately 50% of previously best-known instances, showcasing its robustness and adaptability. In recent studies, distributed NPFSP variants include due-window constraints and hybrid metaheuristics. Mousighichi and Avci (2024) introduced a distributed no-idle with due windows, using mathematical models and iterated greedy methods for scenarios up to 500 jobs. Zhao et al. (2024) proposed a Q-Learning evolutionary algorithm for a distributed mixed no-idle variant with sequence-dependent setup times, optimising for makespan, energy consumption, and tardiness. Zhao et al. (2024) also launched an improved fruit fly optimisation for single-factory instances, pushing performance further. Utama et al. (2024) conducted a systematic literature review covering 63 no-idle flowshop scheduling papers from 1981–2023, identifying research gaps and new directions. In classical NPFSP context, extensions in hybrid and multi-objective scheduling include Li et al. (2025) two-stage iterated greedy for distributed flexible assembly flowshops with sequence-dependent setups emphasizing versatility in model structures.

Despite significant progress in solving the NPFSP, existing metaheuristic approaches often face two key limitations: (1) inadequate balance between global exploration and local exploitation, leading to premature convergence, and (2) lack of effective initialization strategies tailored to minimise tardiness. Most prior methods, including iterated greedy and teaching–learning-based algorithms, have focused primarily on makespan or generic optimisation without leveraging due-date information or hybridization with strong constructive heuristics. To address these gaps, this study proposes a novel hybrid algorithm, ENG, which combines EDD and NEH heuristics with a GA. This integration enhances initial solution quality and strengthens local search, thereby improving convergence and solution robustness. The effectiveness of ENG is validated through Taillard's benchmark instances and an industry case study, demonstrating substantial improvements in tardiness compared to existing approaches.

3. Materials and methods

3.1. Problem designs

The basic goal of scheduling is to find an optimal sequence of jobs that satisfies stated performance objectives. Obtaining an ideal schedule is critical for better productivity and cost effectiveness considering overall tardiness. The following is a thorough explanation of flow shop scheduling problems for multiple machines with the goal of minimising overall tardiness.

$$\left\langle \mathrm{Flowshop_m} \mid \mathrm{permutations_j} \right\rangle \sum_{\mathrm{j=1}}^{\mathrm{n}} \mathrm{T_j}$$

Table 1. Sample dataset.

Jobs	Machines				$\mathrm{dd_j} = \sum \mathrm{P_{ji}} \cdot \mathrm{tf}$
	m_1	m_2	m_3	m_4	
j_1	2	1	3	1	7
j_2	2	3	1	3	9
j_3	1	2	3	4	10
j_4	1	2	3	1	7

A sample data has been considered for multiple machines to illustrate and understand the procedure of finding tardiness of the given sequence. The data used is provided in the Table 1.

Let m represent an ordered group of machines (i=1, 2, 3, …, m) and j be the number of jobs (j=1, 2, 3, …, n) which are all accessible at the beginning time period. P_{ji} is the execution time for job j on machine i, and it has a due date of dd_j relative to the last machine's completion processing time. The due date dd_j is defined as the sum of the processing time of job j across all machines multiplied by tightness factor (tf): $\mathrm{dd_j} = \sum \mathrm{P_{ji}} \cdot \mathrm{tf}$. Even the due date might be provided by the client. As it's a flow shop problem, once the previous machine, m-1, has finished processing the job, m can begin processing it. Each machine completes the jobs one at a time, exactly with a similar sequence,

with no interruptions for idle time. Every machine has a limited capacity to handle one job at a time, and each job must be processed only on a single machine at a given time. The set of places where a job can be scheduled is denoted by P, which implies $j = P$, concerning the fact that the number of places in an arrangement equals the number of jobs. The problem is to schedule the jobs so as to minimise the overall tardiness shown in Equation (1).

$$\text{Total Tardiness} = \sum_{j=1}^{n} T_j \tag{1}$$

The tardiness of each job is expressed as

$$T_j = \max\{0, C_j - dd_j\} \tag{2}$$

where C_j is the completion time of a job j on the last machine. The goal is to identify a job sequence that minimises the total tardiness criteria of jobs.

3.2. Numerical illustration for tardiness calculation - EDD

A sample numerical example has been considered to illustrate and understand the procedure. The data used for this purpose is provided in the Table 1. The table shows sample scheduling flowshop problem with four machines (m_1, m_2, m_3, and m_4), four jobs (j_1, j_2, j_3, and j_4), dd_j is due date of job j and the tightness factor *tf* is assumed to be one for simplicity.

The steps below outline how to determine the total tardiness of jobs in the system that are being processed across several machines.

Step 1. Take into consideration a job sequence.

Step 2. All positions are available during the initial time period considering release time = 0.

Step 3. Determine each job's in-time (T_{in}) and out-time (T_{out}) on m_1. The first job's T_{in} on m_1 will be zero, and its T_{out} will be the addition of release time and processing time needed to complete the job on m_1. T_{out} from the prior job will be T_{in} of the next job on m_1, and so forth.

Step 4. Find the in-time (T_{in}) and out-time of jobs (T_{out}) on other machines. T_{in} of the first job on the next machine will be T_{out} of job on the previous machine. T_{in} of the next job will be the maximum value between the time required to complete the previous job on the same machine and T_{out} of the job on the previous machine.

Step 5. C_j is the amount of time needed to complete job j on the last machines, which indicates the job *j's* completion time.

Step 6. Calculate the tardiness T_j of job j using Equation (2).

Step 7. Calculate the total tardiness of jobs in the system using Equation (1).

The above procedure is applied to solve the scheduling problem shown in Table 1 using the EDD priority sequencing rule. The computation of the same is shown in Table 2.

3.3. Numerical illustration for tardiness calculation - NEH

The NEH algorithm solves the scheduling problem in three stages. In the first stage, the *n* jobs are arranged in a non-increasing order called initial sequence S_Π based on the total of their processing times over the *m* machines. The second stage takes the first two jobs in S_Π and creates a sequence for the two jobs that are the least tardy. In the third stage, the jobs that remain from sequence S_Π are subsequently added

Table 2. Computation of total tardiness of jobs with EDD sequencing rule.

Job	Release time	m_1	T_{in}	T_{out}	m_2	T_{in}	T_{out}	m_3	T_{in}	T_{out}	m_4	T_{in}	T_{out}	C_j	dd_j	max(0, $C_j - dd_j$)
j_1	0	2	0	2	1	2	3	3	3	6	1	6	7	7	7	0
j_4	0	1	2	3	2	3	5	3	6	9	1	9	10	10	7	3
j_2	0	2	3	5	3	5	8	1	9	10	3	10	13	13	9	4
j_3	0	1	5	6	2	8	10	3	10	13	4	13	17	17	10	7
Total Tardiness (TT)																14

one after the other to form a full sequence. Each job is added to a location that minimises the tardiness of the provided partial sequence among all viable positions. The procedures mentioned here describe how to calculate the overall tardiness using NEH.

Step 1. To comprehend the NEH technique, the same numerical is considered as presented in Table 1. Arrange the jobs in decreasing order of summation of processing time of jobs across all machines. Consider initial sequence (S_{Π}): $j_3 - j_2 - j_1 - j_4$

Step 2. Total tardiness of job j_3 which is partial sequence is $[j_3] = [0]$

Step 3. Add j_2 in partial sequence at two positions to get two partial sequences as $[[j_2 - j_3], [j_3 - j_2]]$

Step 4. Determine the tardiness of partial sequences as per the calculation given in the Table 2.

Step 5. Compare the tardiness of the two sequences [4, 4] and select the best sequence with the lowest tardiness. (The two sequences in this case are equally tardy, with $[j_3 - j_2]$ being selected at random as the better sequence.) To calculate the total tardiness of sequences, insert one job at a time at every possible location from the initial sequence S_{Π}.

Step 6. Total tardiness of sequences after inserting j_1 and j_4 of initial sequence:

Step 7. $[[j_1 - j_3 - j_2], [j_3 - j_1 - j_2], [j_3 - j_2 - j_1]] = [10, 9, 11]$

Step 8. $[[j_4 - j_3 - j_1 - j_2], [j_3 - j_4 - j_1 - j_2], [j_3 - j_1 - j_4 - j_2], [j_3 - j_1 - j_2 - j_4] = [18, 17, 17, 17]$

Step 9. Select the most ideal sequence that has the least amount of tardiness. In this instance, any combination of the jobs sequences $[[j_3 - j_4 - j_1 - j_2], [j_3 - j_1 - j_4 - j_2], [j_3 - j_1 - j_2 - j_4]]$ is the best combination to lower the overall tardiness.

3.4. ENG the proposed algorithm

The EDD and NEH techniques described in sections 3.2 and 3.3, while efficient, will not offer the best sequence in most instances as they are based on exact and heuristic concepts. The suggested algorithm, EDD-NEH-GA (ENG), is an inventive technique that uses a conventional GA by including sequences that were derived using EDD and NEH priority sequencing of the job. The population of sequences is then evaluated in order to identify the best sequence. An explanation of the suggested approach is given below.

In scheduling and sequencing problems the combinatorial algorithm ENG is used to create the best possible sequence for minimising tardiness in a multiple machine setting. The GA idea serves as the foundation for the sequences of jobs analysis approach. The popular statement "Survival of the Fittest" attributed to Charles Darwin serves as the foundation for GA. Surviving species are those that are most adaptable to change, not the most powerful or smartest. EDD and NEH are used for comparing the proposed technique. The foundation of the traditional GA is the arbitrary selection of possible chromosomes as an initial population. The solutions and the amount of time needed to find the best solution will be impacted by this initial population. In this research, sequences are created using EDD and NEH and added to the original population when the ENG method is used. The remaining sequences of the population are produced in order to counteract any potential predisposition brought on by particular sequences. Following the generation of the initial population by the use of EDD, NEH, and randomness, this population develops new sequences by undergoing crossover and mutation in series. To obtain the total number of sequences, the population and the sequences produced in the previous phases are combined. The total sequences are assessed depending on tardiness by applying Equation (1) and Equation (2). The sequence that has the least amount of overall tardiness is chosen as the best option. The most suitable option from this iteration is now included in the population of the next iteration. The chosen sequence is considered to be near to optimal if the stopping conditions are satisfied, or the procedure described above is carried out for the subsequent iteration.

Total possible sequences for population size = 50 and total tardiness after ENG approach: The jobs in sequences are represented as 0 means j_1, 1 means j_2, 2 means j_3, and 3 means j_4.

[[0, 1, 2, 3], [1, 2, 3, 0], [1, 2, 3, 0], [3, 1, 2, 0], [3, 0, 1, 2], [0, 1, 3, 2], [1, 2, 0, 3], [1, 2, 0, 3], [2, 0, 3, 1], [1, 0, 2, 3], [0, 1, 2, 3], [2, 3, 0, 1], [0, 1, 2, 3], [0, 3, 1, 2], [3, 2, 0, 1], [1, 0, 3, 2], [3, 2, 0, 1], [1, 3, 0, 2],

[1, 3, 2, 0], [3, 0, 2, 1], [2, 1, 0, 3], [3, 2, 0, 1], [3, 0, 2, 1], [2, 1, 0, 3], [0, 1, 2, 3], [2, 1, 0, 3], [2, 0, 3, 1], [0, 1, 2, 3], [2, 1, 0, 3], [0, 1, 2, 3], [0, 1, 3, 2], [1, 2, 0, 3], [1, 2, 0, 3], [2, 1, 3, 0], [1, 3, 0, 2], [0, 3, 1, 2], [3, 2, 0, 1], [1, 2, 0, 3], [3, 1, 0, 2], [3, 2, 0, 1], [1, 0, 3, 2], [0, 3, 1, 2], [1, 0, 3, 2], [1, 3, 2, 0], [3, 0, 2, 1], [3, 2, 1, 0], [2, 3, 0, 1], [0, 1, 2, 3], [0, 1, 2, 3], [2, 0, 3, 1], [0, 3, 1, 2], [2, 3, 0, 1]] = [18, 22, 22, 15, 14, 17, 22, 22, 17, 19, 18, 17, 18, 14, 18, 18, 18, 21, 22, 19, 19, 18, 19, 19, 18, 19, 17, 18, 19, 18, 17, 22, 22, 19, 21, 14, 18, 22, 12, 18, 18, 14, 18, 22, 19, 20, 17, 18, 18, 17, 14, 17]

Least tardiness among all is "12" corresponding to sequence [3, 1, 0, 2] means sequence $j_4 - j_2 - j_1 - j_3$]. This tardiness is better than EDD and NEH.

4. Results and discussions

4.1. Computational experiments

This section assesses the effectiveness of the EDD, NEH, and ENG techniques by analysing Taillard's dataset (Taillard, 1993). The 120 cases in Taillard's dataset have 20, 50, 100, 200, 500 jobs and 5, 10, 20 machines. 20 jobs that were processed on 5, 10, and 20 machines were used in the computations. In this research,the first five examples from Taillard's dataset with a combination of 20 jobs 5 machines; 20 jobs 10 machines and 20 jobs 20 machines are examined in order to demonstrate the suggested method. The due date is not there in these datasets, so the due date is calculated using the summation of the processing time of the job across all machines multiplied by *tf* =1. The results of the analyses have been carried out on a computer with Windows 10, an Intel(R) Core(TM) i3-9100F CPU @ 3.60GHz operating system, 8GB of RAM, and 500GB of storage. The Python programming language is used to write the computing code, and the outcomes of these techniques are compared. To evaluate the effectiveness of the ENG technique, fifteen test datasets with a variety of machine numbers are selected from Taillard's datasets. The EDD priority rule, the NEH heuristic, and the proposed ENG combinatorial technique are used to test these datasets. Table 3 contains the results of the testing.

Figure 1 represents a comparison of tardiness obtained using EDD, NEH and ENG techniques. Every dataset is processed using the ENG approach, with population sizes of 25, 50, 100, and 200 for 500 iterations. The population size is the total number of job combinations that are sequenced for every iteration. Every data instance is run through 500 iterations. Table 3 shows that the ENG technique yields the optimal sequence with the lowest tardiness in all the instances.

The population size had an impact on how long it required to execute ENG to find the best sequence for each data instance. Table 4 shows the amount of time (in minutes) needed to run the ENG approach on the datasets for 500 epochs. The

Table 3. Total Tardiness of selected instances using EDD, NEH and ENG technique.

			ENG			
Instance	EDD	NEH	Population size=25	Population size=50	Population size=100	Population size=200
Tai20_5_1	10463	**9506**	**9506**	**9506**	**9506**	**9506**
Tai20_5_2	12334	11397	11232	11161	**11051**	11076
Tai20_5_3	11585	10401	10002	10401	**9757**	**9757**
Tai20_5_4	12300	**10902**	**10902**	**10902**	**10902**	**10902**
Tai20_5_5	9888	**9329**	**9329**	**9329**	**9329**	**9329**
Tai20_10_1	14229	13004	13004	**12956**	**12956**	**12956**
Tai20_10_2	15833	14644	**14324**	14337	14337	14337
Tai20_10_3	**11803**	12177	**11803**	**11803**	**11803**	**11803**
Tai20_10_4	15935	**11307**	**11307**	**11307**	**11307**	**11307**
Tai20_10_5	12589	**10884**	**10884**	**10884**	**10884**	**10884**
Tai20_20_1	14617	13792	**13442**	13646	13646	13646
Tai20_20_2	17262	**14932**	**14932**	**14932**	**14932**	**14932**
Tai20_20_3	**15937**	17312	**15937**	**15937**	**15937**	**15937**
Tai20_20_4	16157	**13260**	**13260**	**13260**	**13260**	**13260**
Tai20_20_5	16478	16556	**16199**	**16199**	**16199**	**16199**

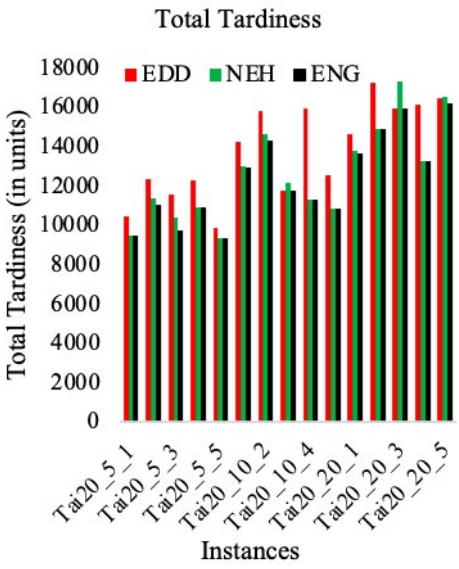

Figure 1. Comparison of total tardiness for EDD, NEH and ENG (Population size=200) techniques.

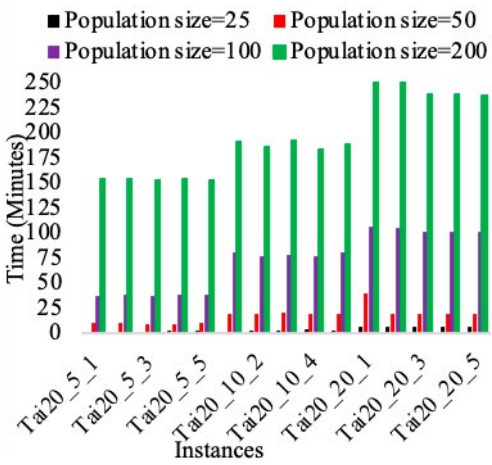

Figure 2. Time required to achieve the best sequence.

impact of population size on the execution of the ENG approach is shown in Figure 2. Based on the figure, it can be observed that the population size is growing and that the execution time required increases with the number of machines. It takes two to seven minutes when there are 25 sequences in the population, nine to forty minutes when there are 50 sequences in the population, 37 to 106 minutes when there are 100 sequences in the population and 153 to 252 minutes when there are 200 sequences in the population. Table 3 and 4 shows that, as opposed to population sizes 100 and 200, one can obtain the optimal sequence in less time with a population size of 25. A handful of instances showed performance improved as the population increased. It was shown by Bari et al. (2024) and Bari and Karande (2023) that a combinatorial technique converges faster than standard GA. In this study, ENG begins its search for the optimal sequence by using EDD and NEH as its best-based sequences for the first iteration, where the sequences are injected into the initial population. The sequence with lower tardiness is added to the population of the subsequent iteration after each

Table 4. Time required execute ENG technique on instances.

	Time (Minutes)			
Instances	Population size=25	Population size=50	Population size=100	Population size=200
Tai20_5_1	2.134	10.311	37.307	154.245
Tai20_5_2	1.959	10.146	39.187	153.778
Tai20_5_3	1.988	9.811	37.921	153.054
Tai20_5_4	2.426	9.810	39.372	154.429
Tai20_5_5	2.450	10.295	38.893	153.352
Tai20_10_1	2.074	19.727	81.577	190.972
Tai20_10_2	3.314	19.462	76.858	185.866
Tai20_10_3	3.482	20.334	77.954	192.065
Tai20_10_4	3.555	19.869	77.741	183.632
Tai20_10_5	3.491	20.158	81.059	189.168
Tai20_20_1	7.057	40.668	106.512	252.639
Tai20_20_2	7.028	19.869	105.165	251.850
Tai20_20_3	6.979	19.869	102.046	238.222
Tai20_20_4	7.031	19.869	102.146	239.144
Tai20_20_5	7.032	19.869	100.980	237.600

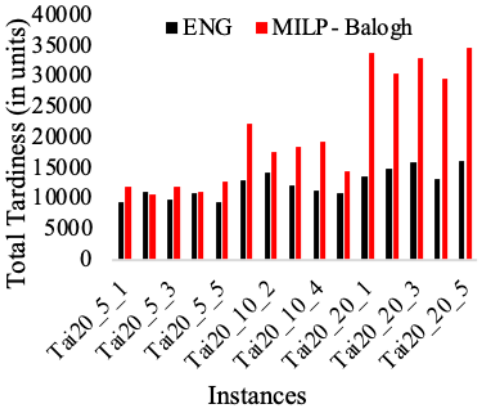

Figure 3. Comparison of ENG and MILP.

iteration. In this study, the algorithm uses 500 epochs as its termination criterion. It has been noted that even with a 25-sequence population, greater outcomes are obtained.

In these experiments, the proposed method is compared to MILP model methods (Balogh et al., 2022). Table 5 shows the lower bounds calculated by Balogh et al. (2022) with their MILP solver and the total tardiness computed using the ENG approach considering the due date with tightness factor = 1. The bold value indicates lesser tardiness among two values. It is observed that 14 of the 15 instances had less tardiness computed by ENG than the MILP solver. Figure 3 shows the graphical representation of tardiness computed by the proposed ENG and MILP solver. As can be seen from the figure, in instances with 20 jobs and 20 machines, the lower bound tardiness calculated by the MILP solver is about twice as high as the tardiness computed by ENG. 14 out of 15 tested Taillard instances of 20 machines and varied number of jobs, which could achieve less tardiness than the best-known solution provided by MILP- Balogh. Thus 93.33% of instances obtained the best solutions.

Table 5. Lower bounds with MILP and tardiness with ENG.

Instances	ENG	MILP- Balogh
Tai20_5_1	**9506**	11967
Tai20_5_2	11076	**10752**
Tai20_5_3	**9757**	12041
Tai20_5_4	**10902**	11163
Tai20_5_5	**9329**	12812
Tai20_10_1	**12956**	22317
Tai20_10_2	**14337**	17657
Tai20_10_3	**12177**	18568
Tai20_10_4	**11307**	19259
Tai20_10_5	**10884**	14414
Tai20_20_1	**13646**	33891
Tai20_20_2	**14932**	30518
Tai20_20_3	**15937**	32949
Tai20_20_4	**13260**	29603
Tai20_20_5	**16199**	34747

4.2. Statistical analysis – t-test

A Raincloud plot comparing total tardiness values for ENG and MILP-Balogh across 15 benchmark instances are presented in Figure 4. Each paired line shows the difference for the same instance, with nearly all lines sloping upward toward MILP-Balogh, indicating higher tardiness for that method. The boxplots and density distributions further illustrate this pattern: ENG exhibits a lower median tardiness and a much narrower spread, while MILP-Balogh shows both higher values and greater variability. These visual findings align with the statistical analysis, where the paired t-test confirmed a significant difference shown in Table 6.

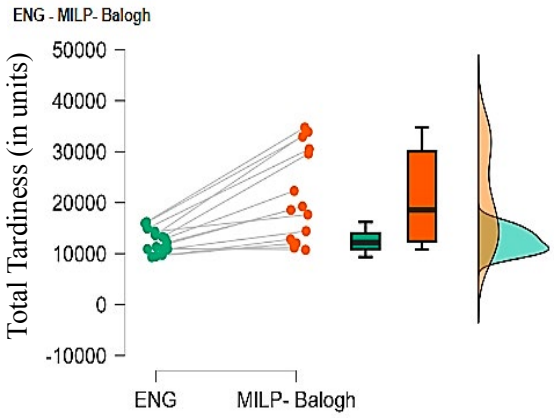

Figure 4. Raincloud plot.

The paired samples t-test (Table 6a) was conducted using Jeffreys's Amazing Statistics Program (JASP), a statistical software to compare the total tardiness values obtained by the proposed ENG algorithm and the MILP-Balogh approach across 15 Taillard benchmark instances. The results reveal a statistically significant difference between two methods ($t = -4.534$, $df = 14$, $p < .001$), indicating that ENG consistently outperforms MILP-Balogh. The negative t-value reflects that ENG achieves lower tardiness than MILP-Balogh. On average, ENG reduced total tardiness by 8430 units, which is a substantial improvement in scheduling performance.

e-ISSN: 2340-4876

Table 6. Statistical analysis (t-test) using JASP.

(a) Paired t-test

Measure 1	Measure 2	t	df	p	Mean Difference	SE Difference
ENG	MILP- Balogh	-4.534	14	< 0.001	-8430	1859

(b) Test of Normality (Shapiro-Wilk)

Measure 1	Measure 2	W	p
ENG	MILP- Balogh	0.881	0.049

(c) Descriptives

Measures	N	Mean	SD	SE	Coefficient of variation
ENG	15	12414	2268	585.7	0.183
MILP- Balogh	15	20844	9095	2348.3	0.436

where *t*- t statistic, *p* – probability value, *df* - degrees of freedom, *W* – normality test statistic, *N* – number of instances, *SD* - standard deviation, *SE* - standard error (of the Mean), *Coefficient of variation* - relative standard deviation

The standard error of the difference ($SE = 1{,}859$) suggests that this improvement is consistent across instances.

Although, Shapiro-Wilk test (Table 6b) ($W = 0.881$, $p = 0.049$) indicates a slight deviation from normality in the distribution of differences, the paired t-test is generally robust to mild non-normality, and the observed p-value ($p < 0.001$) provides strong evidence against the null hypothesis. Therefore, the conclusion that ENG significantly improves tardiness remains valid.

Descriptive statistics (Table 6c) further reinforce these findings. ENG achieved a mean tardiness of 12 414 with a standard deviation of 2268, whereas MILP-Balogh recorded a much higher mean tardiness of 20 844 and a standard deviation of 9,095. This indicates that ENG not only delivers better average performance but also exhibits greater stability and reliability, as shown by its lower coefficient of variation (0.183 compared to 0.436 for MILP-Balogh). In practical terms, this means ENG provides more predictable and consistent results, which is critical in real-world manufacturing environments where variability can lead to operational inefficiencies.

Overall, these statistical outcomes confirm the effectiveness of the proposed ENG algorithm in addressing the research gap. By integrating EDD and NEH heuristics with a GA, ENG achieves superior performance in minimising tardiness under no-idle constraints, offering both significant improvements and robust consistency compared to traditional MILP-based methods.

5. Application of ENG in the steel industry: a case study

The study was conducted in collaboration with a steel industry-specific manufacturing company to validate the technique and compare findings to a real-world scenario. Table 7 shows the company datasheet. The computational framework to simulate the ENG method is constructed by applying Python code, the main window of computational framework as illustrated in Figure 5. There are no additional software or hardware necessities. Table 7 shows a dataset with job numbers and machines, along with job processing times and due dates with a tightness factor of one. The "Browse File" option serves to pick an Excel file containing the dataset's job numbers, processing times on each machine, and job due dates. The number of iterations to give as terminating conditions for the method to be used as input to the model depicted in Figure 6. Once the data is entered, clicking the "Submit" button results in the best sequence with the tardiness, as illustrated in Figure 7.

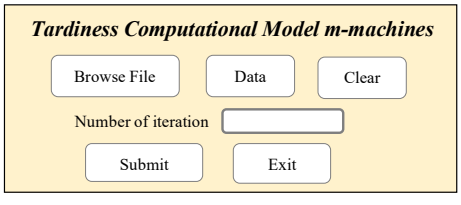

Figure 5. Computational framework.

Table 8 illustrates the tardiness of several approaches. The company's typical strategy is first come, first served (FCFS); Table 8 shows that the tardiness of FCFS is 600730, whereas the tardiness of the

Table 7. Company datasheet.

Job No.	Machines and Processing time						dd_j
	CNC- Zayer	VTL-1	CNC- Union	VTL-2	CNC- Doosan	Drilling	
1	1120	1088	1920	1472	2240	640	8480
2	960	1152	1984	1344	3040	704	9184
3	1024	1120	1920	1536	2400	640	8640
4	960	1024	2112	1440	3072	800	9408
5	100	120	200	192	120	80	812
6	120	128	216	208	140	72	884
7	112	120	200	180	140	88	840
8	140	140	240	220	160	80	980
9	80	100	160	152	140	64	696
10	100	112	180	168	168	80	808
11	92	100	160	160	180	60	752
12	112	120	200	180	224	72	908
13	200	600	560	200	800	160	2520
14	240	520	640	224	880	144	2648
15	224	592	608	192	760	176	2552
16	256	624	704	240	1000	200	3024
17	144	440	480	360	560	120	2104
18	192	360	544	400	512	144	2152
19	176	416	608	480	480	120	2280
20	224	480	672	520	576	128	2600
21	160	320	400	280	440	200	1800
22	200	360	560	320	800	240	2480
23	240	336	448	336	496	256	2112
24	280	400	640	360	880	240	2800
25	90	72	180	48	300	90	780
26	108	240	300	180	420	108	1356
27	96	90	150	60	336	84	816
28	132	270	372	210	480	120	1584
29	40	72	160	48	232	60	612
30	64	140	220	128	300	80	932
31	48	80	128	56	240	60	612
32	80	200	260	168	340	80	1128

Table 8. Tardiness comparison.

Approach	Traditional method used by company (FCFS)	NEH	EDD	ENG
Tardiness	600730	248610	140154	140154

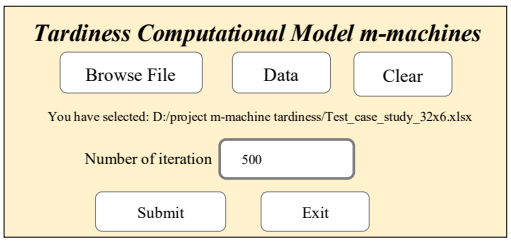

Figure 6. Input to the computational framework.

ENG algorithm is 140154, demonstrating that the proposed strategy outperforms FCFS. If the company implements the proposed ENG, it can save time as well as utilise it for several additional objectives.

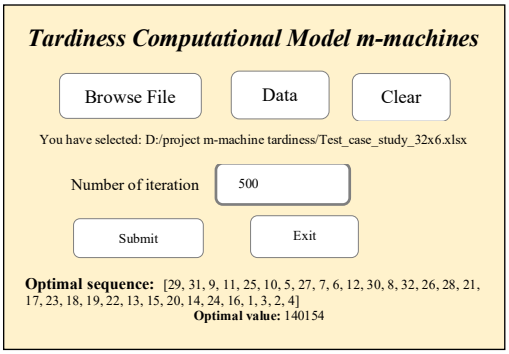

Figure 7. Output tardiness and optimal sequenc.

6. Conclusions

EDD and NEH alone are ineffective in sequencing the jobs to reduce tardiness, but when combined with the evolutionary approach, they produce good results. This research combines EDD, NEH and GA to create an optimal sequence with a lower tardiness performance measure for multiple machines. A total of 15 benchmark datasets were tested for 25, 50, 100 and 200 population size to compute the tardiness with ENG. The results were compared with EDD and NEH, revealing that the total tardiness with the ENG approach is less than EDD and NEH. In the ENG method, the best sequence of each iteration is inserted into the population of the next iteration, enhancing the fitness function of all sequences and promising its convergence. The total tardiness computed using the ENG approach considering the due date with tightness factor = 1. It is observed that 14 of the 15 instances had less tardiness computed by ENG than the MILP solver suggested by Balogh et al. (2022). The algorithm has been tested for a case study in a Steel manufacturing company, and it outperforms. The company's typical strategy is first come, first served. It is noticed that the tardiness of FCFS is 600730, whereas the tardiness of the ENG algorithm is 140154, demonstrating that the proposed strategy outperforms FCFS. If the company implements the proposed ENG, it can save time as well as utilise it for several additional objectives.

The study has a limitation which includes fixed parameters, like mutation rate, and crossover rate. Further, as future scope parameter adjustments can be incorporated. Combinatorial approaches can be used by researchers to improve the accuracy of scheduling issues when combined with other different metaheuristic techniques. Furthermore, changing the stand-alone design parameters for web apps allows the operator to use it remotely on any device.

References

Adiri, I., & Pohoryles, D. (1982). Flowshop/no-idle or no-wait scheduling to minimize the sum of completion times. *Naval Research Logistics Quarterly*, *29*(3), 495–504. https://doi.org/10.1002/nav.3800290311

Afsar, H. M., Lacomme, P., Ren, L., Prodhon, C., & Vigo, D. (2016). Resolution of a job-shop problem with transportation constraints: A master/slave approach. *IFAC-PapersOnLine*, *49*(12), 898–903. https://doi.org/10.1016/j.ifacol.2016.07.889

Balogh, A., Garraffa, M., O'Sullivan, B., & Salassa, F. (2022). MILP-based local search procedures for minimizing total tardiness in the no-idle permutation flowshop problem. *Computers and Operations Research*, *146*, 105862. https://doi.org/10.1016/j.cor.2022.105862

Baptiste, P. & Hguny, L. (1997). A branch and bound algorithm for the f/no- idle/cmax. In *Proceedings of the International Conference on Industrial Engineering and Production Management, IEPM*, 97, 429–438.

Bari, P., Karande, P. & Bag, V. (2024). Hybrid genetic algorithm to minimize scheduling cost with unequal and job dependent earliness tardiness cost. *International Journal of Production Management and Engineering*, *12*(1), 19-30. https://doi.org/10.4995/ijpme.2024.19277

Bari, P., & Karande, P. (2023). Optimal job scheduling to minimize total tardiness by dispatching rules and community evaluation chromosomes. *Decision Making: Applications in Management and Engineering*, *6*(2), 201–250. https://doi.org/10.31181/dmame060104052023b

Bianchi, L., Dorigo, M., Gambardella, L. M., & Gutjahr, W. J. (2009). A survey on metaheuristics for stochastic combinatorial optimization. *Natural Computing*, *8*(2), 239–287. https://doi.org/10.1007/s11047-008-9098-4

Choo, W. M., Wong, L. P., & Khader, A. T. (2016). A modified bee colony optimization with local search approach for job shop scheduling problems relevant to bottleneck machines. *International Journal of Advances in Soft Computing and Its Applications*, *8*(2), 52–78.

Framinan, J. M., Leisten, R., & Ruiz-Usano, R. (2002). Efficient heuristics for flowshop sequencing with the objectives of makespan and flowtime minimisation. *European Journal of Operational Research*, *141*(3), 559–569. https://doi.org/10.1016/S0377-2217(01)00278-8

Jourdan, L., Basseur, M., & Talbi, E. G. (2009). Hybridizing exact methods and metaheuristics: A taxonomy. *European Journal of Operational Research*, *199*(3), 620–629. https://doi.org/10.1016/j.ejor.2007.07.035

Kemmoé, S., Lamy, D., & Tchernev, N. (2015). A job-shop with an energy threshold issue considering operations with consumption peaks. *IFAC-PapersOnLine*, *28*(3), 788–793. https://doi.org/10.1016/j.ifacol.2015.06.179

Li, Y. Z., Meng, L. L., & Zhang, B. (2025). Two-stage iterated greedy algorithm for distributed flexible assembly permutation flowshop scheduling problems with sequence-dependent setup times. *AIMS Mathematics*, *10*(5), 11488–11513. https://doi.org/10.3934/math.2025523

Lunardi, W. T., Birgin, E. G., Ronconi, D. P., & Voos, H. (2021). Metaheuristics for the online printing shop scheduling problem. *European Journal of Operational Research*, *293*(2), 419–441. https://doi.org/10.1016/j.ejor.2020.12.021

Martı, R. & Reinelt, G. (2011) The linear ordering problem: Exact and heuristic methods in combinatorial optimization. *Applied Mathematical Sciences*. 175.

Mei, Y., Nguyen, S., Xue, B., & Zhang, M. (2017). An efficient feature selection algorithm for evolving job shop scheduling rules with genetic programming. *IEEE Transactions on Emerging Topics in Computational Intelligence*, *1*(5), 339–353. https://doi.org/10.1109/TETCI.2017.2743758

Mencía, R., Sierra, M. R., Mencía, C., & Varela, R. (2015). Memetic algorithms for the job shop scheduling problem with operators. *Applied Soft Computing Journal*, *34*, 94–105. https://doi.org/10.1016/j.asoc.2015.05.004

Mirmozaffari, M., Hejazi, S. M., Karamizadeh, N., & Montazeri, A. (2024). A mixed-integer non-linear no-wait open-shop scheduling model for minimizing makespan and total tardiness in manufacturing. *Decision Analytics Journal*, *10*, 100403. https://doi.org/10.1016/j.dajour.2024.100403

Mousighichi, K., & Avci, M. G. (2024). The distributed no-idle permutation flowshop scheduling problem with due windows. *Computational and Applied Mathematics*, *43*(4), 1–32. https://doi.org/10.1007/s40314-024-02702-w

Nip, K., Wang, Z., & Xing, W. (2016). A study on several combination problems of classic shop scheduling and shortest path. *Theoretical Computer Science*, *654*, 175–187. https://doi.org/10.1016/j.tcs.2015.12.027

Riahi, V., Chiong, R., & Zhang, Y. (2020). A new iterated greedy algorithm for no-idle permutation flowshop scheduling with the total tardiness criterion. *Computers and Operations Research*, *117*, 104839. https://doi.org/10.1016/j.cor.2019.104839

Ruiz, R., Vallada, E., & Fernández-Martínez, C. (2009). Scheduling in flowshops with no-idle machines. *Studies in Computational Intelligence*, *230*, 21–51. https://doi.org/10.1007/978-3-642-02836-6_2

Sawik, T. (2000). Mixed integer programming for scheduling flexible flow lines with limited intermediate buffers. *Mathematical and Computer Modelling*, *31*(13), 39–52. https://doi.org/10.1016/S0895-7177(00)00110-2

Shao, W., Pi, D., & Shao, Z. (2018). A hybrid discrete teaching-learning based meta-heuristic for solving no-idle flow shop scheduling problem with total tardiness criterion. *Computers and Operations Research*, *94*, 89–105. https://doi.org/10.1016/j.cor.2018.02.003

Shen, J. N., Wang, L., & Wang, S. Y. (2015). A bi-population EDA for solving the no-idle permutation flow-shop scheduling problem with the total tardiness criterion. *Knowledge-Based Systems*, *74*, 167–175. https://doi.org/10.1016/j.knosys.2014.11.016

Shi, F., Zhao, S., & Meng, Y. (2020). Hybrid algorithm based on improved extended shifting bottleneck procedure and GA for assembly job shop scheduling problem. *International Journal of Production Research*, *58*(9), 2604–2625. https://doi.org/10.1080/00207543.2019.1622052

Sobeyko, O., & Mönch, L. (2016). Heuristic approaches for scheduling jobs in large-scale flexible job shops. *Computers and Operations Research*, *68*, 97–109. https://doi.org/10.1016/j.cor.2015.11.004

Taillard, E. (1993). Benchmarks for basic scheduling problems. *European Journal of Operational Research*, *64*(2), 278–285. https://doi.org/10.1016/0377-2217(93)90182-M

Tasgetiren, F., Pan, Q. K., Suganthan, P. N., & Chen, A. H. L. (2010). A discrete artificial bee colony algorithm for the permutation flow shop scheduling problem with total flowtime criterion. *2010 IEEE World Congress on Computational Intelligence, WCCI 2010 - 2010 IEEE Congress on Evolutionary Computation, CEC 2010*. https://doi.org/10.1109/CEC.2010.5586300

Tasgetiren, F., Pan, Q. K., Suganthan, P. N., & Oner, A. (2013). A discrete artificial bee colony algorithm for the no-idle permutation flowshop scheduling problem with the total tardiness criterion. *Applied Mathematical Modelling*, *37*(10–11), 6758–6779. https://doi.org/10.1016/j.apm.2013.02.011

Utama, D. M., & Al Imron, C.N. (2024). A systematic literature review on no-idle flow shop scheduling problem. *Operations Research Forum, 5*. https://doi.org/10.1007/s43069-024-00304-0

Wang, J. B., & Xia, Z. Q. (2005). No-wait or no-idle permutation flowshop scheduling with dominating machines. *Journal of Applied Mathematics and Computing*, *17*(1–2), 419–432. https://doi.org/10.1007/BF02936066

Williamson, D. P., & Shmoys D. B. (2011). The design of approximation algorithms. Cambridge University Press.

Zhang, J., Ding, G., Zou, Y., Qin, S., & Fu, J. (2019). Review of job shop scheduling research and its new perspectives under Industry 4.0. *Journal of Intelligent Manufacturing*, *30*(4), 1809–1830. https://doi.org/10.1007/s10845-017-1350-2

Zhao, C., Wu, L., Zuo, C., & Zhang, H. (2024). An improved fruit fly optimization algorithm with Q-learning for solving distributed permutation flow shop scheduling problems. *Complex and Intelligent Systems*, *10*(5), 5965–5988. https://doi.org/10.1007/s40747-024-01482-4