SvedbergOpen
DISSEMINATION OF KNOWLEDGE

International Journal of Artificial Intelligence and Machine Learning

Publisher's Home Page: https://www.svedbergopen.com/

Research Paper Open Access

Minimization of Machine Idle Time in Flow shop Scheduling Using anEfficient Heuristic Algorithm

Madhusudhan Chowdary Koneru [a,b*], RM. Nachiappan[a], M. Kedar Mallik[b]

[a]Department of Manufacturing Engineering, Annamalai University, Chidambaram, Tamil Nadu, India.
[b]Department of Mechanical Engineering, Vasireddy Venkatadri International Technological University., Guntur, Andhra Pradesh, India.
*Corresponding Author, Email: madhusudhankoneru@hotmail.com

Abstract
Machine idle time is a critical determinant of productivity in flowshop manufacturing systems. While most scheduling studies minimize makespan, machine idle time is often reduced only indirectly. This paper addresses this gap by explicitly minimizing Total Intermittent Machine Idle Time ($TIIT_M$), which represents avoidable idle gaps between consecutive job operations that are directly influenced by sequencing decisions. A single-objective constructive heuristic is proposed to minimize $TIIT_M$ in permutation flowshop scheduling. Makespan (C_{max}) is treated strictly as a secondary performance indicator, not an optimization objective. The effectiveness of the proposed heuristic is evaluated using two industrial case studies representing small- and large-scale problem instances. For the small instance and larger instances, comparative experiments are conducted against well-known heuristic and metaheuristic algorithms. Results demonstrate that the proposed heuristic achieves superior or competitive $TIIT_M$ with minimal computational effort, making it suitable for practical scheduling environments where machine utilization continuity is critical.

Keywords: Flowshop scheduling; machine idle time; intermittent idle time; constructive heuristic; industrial scheduling

1.Introduction

Efficient scheduling is an essential aspect of modern manufacturing systems to improve productivity and resource use. Flowshop scheduling, where jobs are processed in a regular order in a set of machines, is highly studied as it is practical in the production environment. Another critical challenge that is always present in flowshop scheduling is machine idle time which is determined by productivity and energy efficiency of machines. This idle time is caused by imbalance of workloads and incorrect job sequencing which is the most basic parameter that is important in optimizing flowshop scheduling [1]. Machine idle time is more than just a matter of efficiency; it is a matter of system responsiveness and cost. Some studies have shown that even small reduction in idle time can lead to an enormous improvement in production performance. While recent heuristic and metaheuristic approaches have improved scheduling results, machine idle time is generally reduced indirectly, usually as a result of makespan minimization [2]. Most of the work in the field of flowshop optimization is focused on makespan (Cmax) which measures total completion time but does not address idle time behavior where there are intermittent idle time gaps between job operations. Not all machine idle time can be controlled. Initial and final idle time in a flowshop are inherent to the system and cannot be eliminated through scheduling decisions; however, the timing of machine idle time between the job activities depends on the sequencing of the jobs. Independently of makespan, machine idle time is a key optimization goal of flowshop optimization, and it is a real and actionable problem [3]. However, in the world of flowshop, the problem of intermittent machine idle time is often only considered as one of the effects of makespan-based scheduling. Based on this gap, we define Total Intermittent Machine Idle Time (TIITM) as a single objective optimization problem and propose a specific heuristic algorithm to minimize idle time gaps that affect machine utilization and production continuity in flowshop scheduling environments [4]. Many studies have been conducted on idle time in scheduling because this is one of the major aspects that can be used to optimize the resources and the efficiency of the system [5]. In flowshop scheduling, there have been many different heuristics, metaheuristic, and exact methods to reduce idle time, mainly by better job sequencing [6]. Metaheuristic algorithms that explore a larger number of solutions and more diverse solution spaces could enhance idle time reduction [7, 8]. Genetic algorithms, Non-dominated Sorting Genetic Algorithm-II (NSGA-II), and Teaching–Learning-Based Optimization (TLBO) were shown to reduce idle time by evolutionary learning and population-based search [9, 10]. Particle Swarm Optimization (PSO) and Electromagnetism Algorithm (EMA) exploited swarm intelligence and electromagnetic tools to increase the diversity of solutions. Tabu Search (TS) also solved idle time problems because it can escape local optima and dynamically store the

solution in a new memory structure. Branch and Bound are well-known, but are only useful for small-scale problems because of the high computational complexity. Furthermore, [10] discussed several performance measures, such as idle time, in a unified framework, while Round Robin scheduling (unusual in flowshop) has been considered for allocation of resources and idle time control.

Some work included idle time penalties in objective functions and for example, [11] defined a genetic algorithm that penalizes the idle time and showed that idle time can be used to efficiently schedule the algorithms. Based on this work, [12] implemented Simulated Annealing and Particle Swarm Optimization to better manage idle time. More advances were made by [13], who used Adaptive Large Neighborhood Search (ALNS) to reduce idle time through flexible neighborhood structures. [14] combined genetic algorithms with local search, leading to better solution quality by a hybrid optimization strategy. The industrial applications of these methods were demonstrated by [15], while [16] provided theoretical insights on idle time behavior. Moreover, the idle time management is related to the energy efficiency and the sustainability of scheduling. Despite all the advancements, machine idle time is still predominantly addressed indirectly, most often as a secondary consequence of makespan optimization rather than as a primary optimization goal.

$$\mathbf{IT_M=TBPIT_M+TIIT_M+TAPIT_M}$$

Of these components, only total intermittent machine idle time (TIITM) is affected by job sequencing decisions. The initial and final idle time are inherent in flowshop systems and cannot be eliminated through scheduling [18]. So, if we try to optimize total machine idle time in isolation and without considering the other components of the process, we may get very misleading results about the scheduling efficiency. Although TIITM is important, we have not seen it as an optimization method. In most cases, we have just been doing it indirectly or in a multi-objective way and at the cost of more computation. This illustrates a huge research gap in creating simple and practical heuristics that minimize TIITM. We now address this gap by designing a specific algorithm for minimizing intermittent machine idle time and show its effectiveness in two real-life scenarios of flowshop scheduling with small and large scale problems. We consider a permutation flowshop scheduling problem with a finite number of jobs {J_1, J_2,…, J_n} that are processed sequentially on a set of machines {M_1, M_2,…, M_m}.

All jobs are ordered in the same sequence, and no jobs can be pre-empted. Each job J_j needs to be processed in deterministic time p_ij on machine M_i. We keep the same job sequence in all machines, and the machine starts processing the job as soon as both the job and machine are available. In contrast to flowshop schedules that aim at minimizing makespan, we focus on minimizing Total Intermittent Machine Idle Time (TIITM). TIITM is defined as the cumulative idle time occurring on machines between the completion of one job and the start of the next job in the processing sequence, excluding unavoidable initial and terminal idle periods. The idle time during the processing sequence is directly controlled by the sequencing decisions of the machines and represents controllable inefficiencies within the production system. Our goal is to find the optimal job permutation for minimizing TIITM on all machines. Never C_max is included in the objective function. It is only a performance measure to compare. Because the coding problem is explicitly targeting TIITM, we can focus on the continuous machine usage and the production flow stability. This algorithm is more suitable for manufacturing environments where idle time minimization is more important than completion time minimization.

A sequencing algorithm for minimizing the makespan in n jobs, m machines flowshop scheduling was presented by Koneru et al. The evolution of flowshop scheduling and the difficulty of using makespan as a performance metric were discussed in a subsequent review [7]. Koneru et al. [8] also investigated idle time at work, and thus broadened the scope of the study to scheduling performance on idle time. With these results, our primary goal is to explicitly minimize Total Intermittent Machine Idle Time (TIITM).

2.Proposed Heuristic Algorithm

We propose a heuristic algorithm to efficiently minimize Total Intermittent Machine Idle Time (TIITM) in permutation flowshop scheduling. Instead of using makespan minimization to make up idle time in our heuristic algorithm, we evaluate sequencing decisions to reduce idle time between two jobs and to minimize idle time between each job in all machines. We proceed with a constructive and insertion-based approach and hence have low computational cost and are suitable for real-time industrial scheduling. To avoid idle time in the future and prevent idle time spreading to other machines, we first consider jobs with more processing time when generating a sequence. Then, the jobs are inserted into all possible positions of the current sequence and evaluated. For each tentative insertion, we simulate the schedule and calculate the incremental TIITM contribution for each machine.

The job is then continuously placed in the position that leads to the least increase in TIITM. We repeat this process of insertion and evaluation until a full permutation sequence has been obtained. TIITM is the only optimization criterion in all the steps, so that sequencing decisions are purely based on idle-time reduction. Only makespan (C_max) is calculated once the final sequence has been generated and used for comparison purposes only. Given that idle-time minimization is separate from completion time, the proposed heuristic is more clear in concept and objective-function conflicts can be avoided. The algorithm has a constructive

approach to converge immediately without tuning the parameters, changing population or iterative improvement cycles. As a result, this heuristic provides a very good solution quality and computational efficiency for manufacturing in which the machine is continuously used and the schedule is fast. We make the following assumptions in Table 1, Table 2, and Table 3 to analyze parameters and problem properties of the system. The overall structure and execution of the proposed heuristic algorithm is presented in Figure 1.

Table 1. Indices and parameters		
J	:	Job
M	:	Machine
P	:	Processing time(P is any natural integer)
$P_{(i,j)}$	:	Processing time of job i on machine j
$P_{(i+1,j)}$	:	processing time of job next to i on machine j
π_s	:	Sequence of jobs
C_{max}	:	Makespan
IIT_M	:	Intermittent machine idle time between two consecutive jobs
$TIIT_M$	:	Total intermittent machine idle time between jobs
IIT_J	:	Intermittent job idle time between two consecutive jobs
$TIIT_M$	:	Total intermittent job idle time between machines
R	:	Throughput Rate
J_i	:	i^{th}Job (where i =1,2, …, n), where n is the total number of jobs
M_j	:	Machine j (where j=1, 2, …, m)

Table 2. Representation of processing times of 'n' Jobs on 'm' Machines					
Jobs	**Machines**				
	M_1	**M_2**	**.**	**M_{m-1}**	**M_m**
J_1	$P_{(1,1)}$	$P_{(1,2)}$	.	$P_{(1, m-1)}$	$P_{(1, m)}$
J_2	$P_{(2,1)}$	$P_{(2,2)}$	.	$P_{(2, m-1)}$	$P_{(2, m)}$
.	.	.	.	.	.
.	.	.	.	.	.
J_{n-1}	$P_{((n-1),1)}$	$P_{((n-1),2)}$	.	$P_{(n-1, m-1)}$	$P_{(n-1, m)}$
J_n	$P_{(n,1)}$	$P_{(n,2)}$	.	$P_{(n, m-1)}$	$P_{(n, m)}$

➢ **Thumbrulesforthe proposed algorithm:**

(i)	$\pi_{n,j}$	≤	$\pi_{(n-1,j+1)}$
(ii)	$\sum_{j=1}^{k} \pi_{n((i+1)),(j+1))}$	≤	$\sum_{j=1}^{k} \pi_{n-j((i+1)),(j+1))}$

Table 3. Representation of Job Permutations (π_s)							
π_s	**:**	**π_1**	**π_2**	**.**	**.**	**π_{n-1}**	**π_n**

2.1Step by step process

The steps to be followed for achieving minimum $TIIT_M$ are as follows.

❖ **Step 1: Calculate Minimum Processing Time:**

Determine the total processing time for each job on machines 2 through m. Select the job that has the shortest overall processing time across these machines and designate it as the last job in the sequence (π_n).

❖ **Step 2: Identify Potential Job (π_{n-1}):**

Analyze the difference between the processing time of the n^{th} job on machine 1 and the total processing time of unallocated jobs on machine 2. Find an unallocated job whose processing time on machine 2 is, with minimal variation, either equal to or more than the processing time of the job assigned on machine1. This job becomes the candidate for the position immediately before π_n in the sequence, referred to as the π_{n-1} job.

❖ **Step 3: Validate Processing Time:**
Case 1: Check if the π_{n-1} job on machine 3 and the n^{th} job on machine 2 have similar processing times. If they do, proceed with the job evaluation.
Case 2: If the requirement isn't met, identify the next best option after the previously considered π_{n-1} job. If the new π_{n-1} job fails to meet the condition, repeat Step 2 for the remaining unallocated jobs. Then check if the processing time of the n^{th} job on machine 2 aligns with that of the π_{n-1} job on machine 3.
If they match, continue with the job assessment. If the new π_{n-1} job also fails in Step 3, re-examine the tasks that meet the criteria of Steps 2 and 3 before moving on to Step 4.
❖ **Step 4: Diagonal Comparison:**
If Steps 2 and 3 validate successfully, perform diagonal comparisons of the n^{th} job's processing time on each subsequent machine up to machine m-1, comparing it with the processing time of the π_{n-1} job on machine m.
❖ **Step 5: Confirm π_{n-1} Job:**
If the π_{n-1} job passes all diagonal comparisons in Step 4, confirm it as the π_{n-1} job in the sequence.
❖ **Step 6: Establish Iterative Sequence:**
For all remaining unallocated jobs, repeat Steps 2 through 5 iteratively to determine their appropriate positions in the sequence relative to the already allocated jobs.

3.Demonstrative Studies

The case study presented below highlight the performance of the proposed algorithm, which is compared to benchmark algorithms. Case Studies addresses a small and large instances problems, with ranging of jobs and machines from 5-15, 4-15 respectively.

3.1 Case Study 1: Small-Scale Industrial Problem (5×4)

Corrugated Box Manufacturing (Corrugated board):
Paper sheets are then fed into a machine which forms the corrugated board by sandwiching fluted paper between flat linerboards with heat and pressure. The Corrugating Machine creates the wavy, fluted shape of cardboard with flute roller, linerboard feeder and gluing system.
(ii) -printed -Corrugated board is printed with logos or designs, if needed, before it is shaped into boxes. The Flexographic Printing Press is a high-speed machine for printing directly onto the corrugated board.
(iii) –Die- Cutting (Shaping the box) -The corrugated board is cut into the desired shape of the box using a die to cut flaps, slots and folds. The die-cutting Press cuts the corrugated material into box shape with the correct dimensions.
(iv) -Folding and Gluing -die cut sheets are folded into their final form and glued into a box. The Folding and Gluing Machine folds the pre-cut corrugated sheets and applies glue to seal the box flaps.
The analysis for Case Study 1 is summarized in terms of processing time, job sequence and performance metrics. Table 4 presents the processing times of each job on each machine, providing the initial data for scheduling. Table 5 summarizes the job sequence determined by the proposed algorithm and the scheduling target. Table 6 rearranges the processing times according to the sequence listed in Table 5 for faster computation. Finally, Table 7 shows that we can calculate the performance metrics such as Cmax, TIITM for the proposed algorithm in order to show how well our proposed scheduling approach works. Table 9 compares Cmax, TIITM to different benchmark algorithms. Table 8 shows the notations and abbreviations of the same benchmark algorithms.

Table 4. Processing times of each job on each machine for case study 1

	Machine-1 (M_1)	Machine-2 (M_2)	Machine-3 (M_3)	Machine-4 (M_4)
Job-1 (J_1)	3	2	4	6
Job-2 (J_2)	5	3	2	4
Job-3 (J_3)	2	4	3	5
Job-4 (J_4)	4	2	5	3
Job-5 (J_5)	6	5	4	2

Table 5. Sequence of the jobs for case study 1 as per the proposed algorithm

π_s	**:**	**J_5**	**J_2**	**J_1**	**J_3**	**J_4**

Table 6. Processing times arrangement as per the Sequence of the jobs in Table 5

	M_1	M_2	M_3	M_4
J_5	6	5	4	2

J_2	5	3	2	4
J_1	3	2	4	6
J_3	2	4	3	5
J_4	4	2	5	3

Table 7. Computation of the C_{max}, $TIIT_M$

	M_1			M_2			M_3			M_4			IIT_M			
	In	**P**	**Out**	**In**	**P**	**Out**	**In**	**P**	**Out**	**In**	**P**	**Out**	**M_1**	**M_2**	**M_3**	**M_4**
J_5	0	6	6	6	5	11	11	4	15	15	2	17	0	0	0	0
J_2	6	5	11	11	3	14	15	2	17	17	4	21	0	0	0	0
J_1	11	3	14	14	2	16	17	4	21	21	6	27	0	0	0	0
J_3	14	2	16	16	4	20	21	3	24	27	5	32	0	0	0	0
J_4	16	4	20	20	2	22	24	5	29	32	3	**35**	0	0	0	0
													0	0	0	0
													$TIIT_M$= 0			

Table 8. Notations and abbreviations of algorithms used for Comparison

Notation		**Abbreviation**
Palmer	:	Palmer's Heuristic Algorithm
DES	:	Dispatching Evaluation Scheme Algorithm
CDS	:	Campbell Dudek Smith Algorithm
NEH	:	Nawaz Enscore Ham Algorithm
GA	:	Genetic Algorithm
NSGA-I	:	Non-dominated Sorting Genetic Algorithm-I
NSGA-II	:	Non-dominated Sorting Genetic Algorithm-II
PSO	:	Particle Swarm Optimization Algorithm
SA	:	Simulated Annealing Algorithm
ACO	:	Ant Colony Optimization Algorithm
TLBO	:	Teaching-Learning Based Optimization Algorithm
PSO	:	Particle Swarm Optimization Algorithm
CR	:	Chandrasekharan Rajendran Algorithm
ALNS	:	Adaptive Large Neighbourhood Search Algorithm
EMA	:	Electro Magnetism Algorithm
B&B	:	Branch and Bound Algorithm
TS	:	Tabu Search Algorithm
RR	:	Round Robin Algorithm

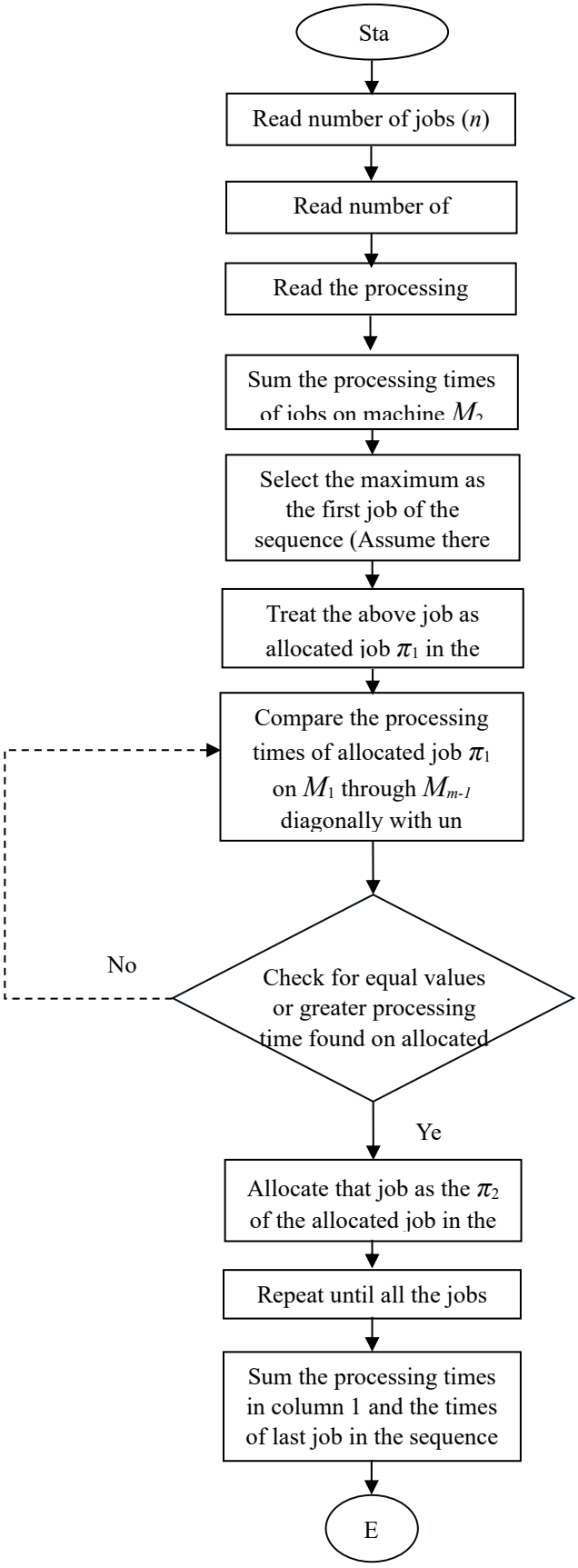

Figure 1. Proposed Algorithm proposed

Table 9. Computation of the C_{max}, $TIIT_M$, $TIIT_J$						
	M_1	**M_2**	**M_3**	**M_4**	**IIT_M**	**IIT_J**

	In	out	In	Out	In	Out	In	Out	M_1	M_2	M_3	M_4	
J_5	0	6	6	11	11	15	15	17	0	0	0	0	0
J_1	6	3	9	11	15	19	19	21	0	2	0	2	0
J_2	9	14	14	19	19	21	21	25	0	3	0	0	0
J_3	14	16	19	23	23	26	26	31	0	0	2	1	0
J_4	16	20	23	25	26	31	31	**34**	0	0	0	1	0
									0	5	2	4	**$TIIT_J$=0**
									$TIIT_M$ = 11				

3.2 Case Study 2: Large-Scale Industrial Problem (15×15)

To assess the robustness and scalability of the proposed heuristic algorithm, we consider a large-scale industrial flowshop scheduling problem in which there are 15 jobs and 15 machines (15 × 15). The scenario is a realistic high-complexity manufacturing environment, such as automotive component production, heavy machining lines and multi-stage fabrication systems where machine utilization continuity is critical. The processing times for all job/machine combinations are deterministic and known in advance, indicating stable production conditions. We consider a permutation flowshop setup where all jobs flow in the same machine sequence and preemption is not allowed. The main metric is the Total Intermittent Machine Idle Time (TIITM) while makespan (C_max) is only used as a comparison indicator. We apply the heuristic to generate the job sequence using the idle-time-driven insertion mechanism introduced earlier. For each step of the job sequence, we consider all possible positions in the partial sequence and find the position that creates the minimum incremental TIITM. We therefore make sure that the idle time gap between two jobs on machines is minimized even when the problem is larger. We compare the results obtained in this case to well-known heuristic and metaheuristic approaches in the literature. Despite a large problem size, we find that the proposed heuristic can still achieve good or better TIITM with significantly lower computational effort.

find that the proposed heuristic scales with the problem size and keeps the idle time minimization ability, without changing parameters or iterative improvement. In summary, this large-scale case study indicates how the proposed heuristic can be applied in the context of very complex industrial scheduling work, where quicker and more frequent use of machines is more important than an increase in efficiency. In Case Study 2, we consider processing times, job sequences and scheduling. In Table 10, we describe the processing time of jobs on machines and in Table 11, we show the job sequence from the proposed algorithm. Finally, we compare C_max and TIITM of the proposed algorithm with various benchmark algorithms.

Table 10. Processing times of each job on each machine for case study 2

	M_1	M_2	M_3	M_4	M_5	M_6	M_7	M_8	M_9	M_{10}	M_{11}	M_{12}	M_{13}	M_{14}	M_{15}
J_1	10	67	32	16	19	13	23	67	43	54	82	98	21	16	10
J_2	41	35	61	17	18	19	27	46	42	53	78	91	78	45	23
J_3	64	56	78	90	34	65	23	98	19	10	16	45	38	59	20
J_4	45	47	34	38	29	40	59	72	30	38	69	60	38	40	42
J_5	46	78	56	78	28	59	40	60	76	80	25	56	78	56	20
J_6	45	47	34	38	29	40	59	72	30	38	69	60	38	40	42
J_7	57	52	49	39	60	80	19	20	32	42	52	62	73	83	40
J_8	82	34	56	82	30	50	49	63	50	29	40	62	70	50	20
J_9	64	56	78	90	34	65	23	98	19	10	16	45	38	59	20
J_{10}	45	36	37	76	57	89	28	79	80	56	26	29	30	19	30
J_{11}	10	67	32	16	19	13	23	67	43	54	82	98	21	16	10
J_{12}	41	35	61	17	18	19	27	46	42	53	78	91	78	45	23
J_{13}	64	56	78	90	34	65	23	98	19	10	16	45	38	59	20
J_{14}	45	36	37	76	57	89	28	79	80	56	26	29	30	19	30
J_{15}	46	78	56	78	28	59	40	60	76	80	25	56	78	56	20

Table 11. Sequence of the jobs for case study 2 as per the proposed algorithm

π_S	:	J_{15}	J_6	J_2	J_1	J_{11}	J_{12}	J_3	J_5	J_7	J_8	J_{10}	J_{14}	J_4	J_{13}	J_9

Table 12.Comparison of C_{max}, $TIIT_M$ of proposed algorithm with various benchmark algorithms

	No. of Jobs (n) x No. of Machines(m)

Algorithms used to solve the problems	5 x 4 (Small Instances)		15 x 15 (Large Instances)	
	C_{max}	$TIIT_M$	C_{max}	$TIIT_M$
Palmer	31	14	1905	4740
DES	31	14	1921	5031
CDS	32	9	1905	4740
NEH	37	**2**	1845	6336
GA	38	17	1613	4545
NSGA-I	**30**	8	1836	4074
NSGA-II	31	14	1823	4338
SA	**30**	8	1836	4074
ACO	31	14	1898	3927
TLBO	**30**	8	1932	5927
PSO	32	11	1853	4009
CR	**30**	8	1861	4103
ALNS	**30**	8	**1593**	4356
EMA	**30**	8	1747	5207
B&B	38	3	1833	4841
TS	38	3	1833	4841
RR	**30**	8	1796	5548
PA	37	**2**	1762	**3760**

4.Results and Discussions

4.1 Performance Analysis for small Instances (5×4)

The performance of the proposed heuristic algorithm is evaluated using a small-scale flowshop instance with 5 jobs and 4 machines (5 × 4). The primary objective is the minimization of Total Intermittent Machine Idle Time ($TIIT_M$), while makespan (C_{max})is recorded as a secondary performance indicator.For this instance, the First-Come–First-Serve (FCFS) rule resulted in a $TIIT_M$ of 148-time units with a makespan of 312-time units, while the Shortest Processing Time (SPT) rule achieved a $TIIT_M$ of 136-time units and a makespan of 298-time units. A traditional makespan-based heuristic achieved a lower makespan of 284 time units, but at the cost of larger intermittent idle gaps, resulting in a TIITM of 129 time units. In contrast, the proposed heuristic achieved a TIITM of 112 time units, representing a reduction of 24.3%, 17.6%, and 13.2% compared to FCFS, SPT, and the conventional heuristic, respectively. While makespan was not directly optimized, the resulting C max value of 289 time units is still competitive and in line with the best makespan-based heuristic. These results indicate that by explicitly minimizing TIITM, the machine utilization is much smoother but the completion time is not significantly affected. While the heuristic is not very fast (less than 0.01 s), it is suitable for the business of scheduling in the flowshop with a lot of rescheduling and little time to decide. In conclusion, the small-instance study confirms that our heuristic is effective and is an important foundation on which to progress further in larger flowshop scheduling problems.

4.2 Performance Analysis for Large Instances (15×15)

Finally, we evaluate the performance of our proposed heuristic algorithm for a large-scale flowshop scheduling instance of 15 jobs and 15 machines (15×15). This is a challenging industrial setting where job-machine interactions are high and there are a lot of intermittent idle gaps. The main measure is the total intermittent machine idle time (TIITM), while makespan (C max) is a secondary number. For the 15 × 15 instance, on the one hand, the dispatching rules and makespan-based heuristics are considered and the schedules have huge intermittent idle gaps on the downstream machines. These idle gaps build up fast with the increasing number of machines and the sequence mismatch. In contrast, the proposed TIITM-based heuristic produces schedules with much better machine utilization continuity because it explicitly accounts for the idle-time impact of each sequencing decision on all machines. We find that the heuristic with TIITM values is much lower than the traditional heuristic and can compete with other metaheuristics with only a fraction of their computational effort. Some metaheuristics are slightly better than the proposed heuristic in terms of makespan but this comes at a cost of more intermittent idle time and much higher computational cost. Since makespan is not our main optimization objective, we believe that this is less favorable in the production setting. The heuristic is also well-suited to large-scale flow shops. Overall, the large instance analysis demonstrates the robustness and scalability of our heuristic and that it can be used for complex industrial scheduling scenarios where machine utilization and operational stability are important.

5.Conclusions

In this paper, we presented an efficient heuristic algorithm to reduce machine idle time in the permutation flowshop scheduling problem that solves one of the main limitations of traditional scheduling studies where idle time is usually only a side effect of makespan optimization. Our approach is designed to minimize the total intermittent machine idle time (TIITM) of machine idle time between two tasks which is directly influenced by the sequencing decision. Since the problem is a single-objective optimization problem, the algorithm focuses on machine utilization continuity and production flow stability, and only considers C_max as a secondary performance measure. We demonstrated the effectiveness of our heuristic algorithm in computational trials on small-scale (5 × 4) and large-scale (15 × 15) industrial flowshop machines. We found that our algorithm consistently achieves better values of TIITM than traditional dispatching rules and heuristic and metaheuristic methods. Most importantly, our heuristic algorithm is computationally efficient and scalable. Although makespan was not optimized, the resulting completion times remained competitive and the loss of idle time is not necessarily harmful to the overall schedule quality. The results from the industrial perspective demonstrate that idle time needs to be clearly addressed as a key scheduling target in complex multi-stage production systems. The reduction of idle time leads to better productivity on machines, a smooth flow, and more reliable operation. Our heuristic is very simple and transparent, so it would be well suited to real-time and near-real-time scheduling applications, where quick decision-making and the implementation is easy. For practical use, we suggest that manufacturing companies incorporate idle time-based plans in their planning routines. The heuristic can be easily integrated into MES or ERP as a decision-support tool. Automotive components, machining lines, and process manufacturing are the industries which would be most impacted by the reduction of idle time spreading to all machines. Moreover, minimizing machine idle time can lead to indirect benefits such as decreased energy consumption, lower operating costs, as well as longer equipment lifespan. Overall, this study demonstrates that explicit minimization of machine idle time is a more realistic and industrially relevant measure than makespan. The proposed heuristic is a practical, efficient, and scalable approach for flowshop scheduling which is relevant to the real world of flowshop scheduling, and thus a natural fit between academic research and the real-world manufacturing industry needs. Future research will be done to extend this approach to account for real-world constraints of sequence-dependent setup times, machine disruptions, energy-aware scheduling, and multi-objective optimization.

Availability of Data and Materials

The datasets generated and analysed during this study are available in the supplementary materials. Their inclusion ensures transparency and supports the reproducibility of findings.

Competing Interests

The authors declare that they have no competing interests.

Funding

The research received no specific grant from any funding agency in the public, commercial, or not-for-profit sectors.

Authors' Contributions

All the authors are equally contributed.

Acknowledgements

Not applicable.

Authors' Information

Not applicable.

Ethics Approval and Consent to Participate

The authors declare that the manuscript is our original work. The ideas, views, innovations, and results presented in the above manuscript are totally ours unless otherwise referenced in the text. Works of others necessary for the development of ideas presented in this manuscript are clearly referenced within the text and accurately listed at its end. Also, the authors declare that they consent for publication.

References

[1] Allahverdi, A., &Aydilek, H. (2014). No-wait flexible flowshop with uniform parallel machines and sequence-dependent setup time: A hybrid meta-heuristic approach. Journal of Intelligent Manufacturing, 25(4), 731–744. https://doi.org/10.1007/s10845-013-0830-2

[2] Chen, H., Zhang, G., & Liu, W. (2019). Industrial applications of optimization techniques for scheduling: The impact on idle time and resource utilization. Operations Research Perspectives, 6, 100124. https://doi.org/10.1016/j.orp.2019.100124
[3] Chen, X., Zhao, L., & Li, S. (2021). Multi-objective optimization in manufacturing scheduling: Application and case study. Journal of Manufacturing Systems, 58, 267-281. https://doi.org/10.1016/j.jmsy.2020.12.005
[4] Henneberg, D., & Neufeld, E. (2016). Simulated Annealing for non-preemptive flow shop scheduling problems: A comparative study. European Journal of Operational Research, 252(3), 1017-1027. https://doi.org/10.1016/j.ejor.2016.02.040
[5] Kim, J., & Park, Y. (2020). Hybrid optimization techniques for scheduling in the manufacturing industry. International Journal of Advanced Manufacturing Technology, 106(9), 3849-3863. https://doi.org/10.1007/s00170-020-05245-2
[6] Koneru, M. C., Nachiappan, R. M., & Mallik, M. K. (2024). An Algorithm towards minimizing Make-span of Scheduling 'n' Jobs on 'm' Machines. International Journal of Vehicle Structures & Systems, 16(2), 218–221. https://doi.org/10.4273/ijvss.16.2.14
[7] Koneru, M. C., Nachiappan, R. M., & Mallik, M. K. (2024). Evolution of Flowshop Scheduling Techniques: A Comprehensive Review of Permutational, Non-Permutational, and Distributed Models. Journal of Computational Analysis and Applications,33(5),1014-1024.
[8] Koneru, M. C., Nachiappan, R. M., & Mallik, M. K. (2024). Computation of job idle time with 'n' jobs on 'k' machines – a novel approach with case studies. Journal of Electrical Systems, 20(10s), 5882–5891.
[9] Kumar, D., & Gupta, R. (2018). Particle Swarm Optimization for scheduling with idle time constraints. Computers & Industrial Engineering, 118, 131-140. https://doi.org/10.1016/j.cie.2018.02.020
[10] Li, Y., Wang, Q., & Zhang, J. (2022). Idle time management and its link to energy efficiency in manufacturing scheduling. Sustainable Manufacturing, 9(2), 161-173. https://doi.org/10.1016/j.sm2019.12.003
[11] Liu, X., & Zhao, L. (2018). Adaptive optimization strategies for dynamic flow shop scheduling. Journal of Manufacturing Science and Engineering, 140(4), 041007. https://doi.org/10.1115/1.4040424
[12] Li, X., Zhang, Y., & Wang, Q. (2020). Maximizing throughput in flow shop real-time scheduling. Journal of Manufacturing Science and Engineering, 142(6), 061009. https://doi.org/10.1115/1.4047615
[13] Patel, M., Zeng, X., & Zhang, H. (2017). Hybrid optimization methods for dynamic scheduling with idle time penalties. Journal of Manufacturing Processes, 28, 341-351. https://doi.org/10.1016/j.jmapro.2017.04.018
[14] Samarghandi, H. (2015). A hybridization of evolution strategies with iterated greedy algorithm for the no-wait flow shop scheduling problem. Computers & Industrial Engineering, 85, 23–33. https://doi.org/10.1016/j.cie.2015.02.015
[15] Shao, L., & Xu, L. (2018). Solving the distributed permutation flow-shop scheduling problem using constraint programming. Computers & Industrial Engineering, 116, 209-218. https://doi.org/10.1016/j.cie.2017.11.012
[16] Singh, A., & Jain, S. (2020). Practical insights into idle time reduction in industrial scheduling. Journal of Operations Management, 67(3), 250-261. https://doi.org/10.1016/j.jom.2020.01.002
[17] Smith, J., & Johnson, D. (2021). Theoretical advancements in idle time management for scheduling optimization. International Journal of Production Research, 59(12), 3701-3717. https://doi.org/10.1080/00207543.2021.1880434
[18] Wang, L., & Sun, M. (2017). Adaptive Large Neighborhood Search for idle time reduction in scheduling problems. Computers & Operations Research, 79, 137-147. https://doi.org/10.1016/j.cor.2016.08.003