Solving Multi-stage Flexible Flow Shop Scheduling Problem with Cooling Times in the Steel Industry

Lotfi HIDRI

Industrial Engineering Department, College of Engineering, King Saud University, Riyadh, 11421, KSA
lhidri@ksu.edu.sa

Abstract: Steel manufacturing includes multiple production stages, such as steel-making, and rolling. Multi-stage flexible flow shops are used for modelling this process. Cooling is one of the operations encountered in the steel-making process, which is essential for controlling the microstructure of the steel, as it directly affects its mechanical properties. A proper cooling operation can enhance the strength, hardness, and durability of the final product. Moreover, it helps to prevent defects such as metal warping and cracking, thereby ensuring a better quality and usability of the end product. While preparing the production schedule, the cooling time is generally ignored or included in the processing time for a certain task. As the cooling time can be more significant in some cases, this study considers it separately from the processing time. This paper tackles the complex problem of multi-stage flexible flow shop scheduling with cooling times, which is known to be strongly NP-hard. In order to find near-optimal solutions for this problem, a two-phase heuristic optimisation algorithm is proposed, using the solution obtained for the parallel machine scheduling problems with specific constraints such as release dates, cooling times, and delivery times. This paper also presents a novel and efficient lower bound for the aforementioned scheduling problem, which is employed for assessing the performance of the heuristic algorithm by relaxing the capacity constraints in all stages, except for one of them. The comprehensive experiments which were carried out based on standard benchmark test problems validate the effectiveness of the proposed methods.

Keywords: Multi-stage flexible flow shop, Steel-making industry, Cooling time, Heuristic, Lower bound.

1. Introduction

Modern industrial processes are modelled based on multi-stage flexible flow shop scheduling problems (FFSPs) (Billaut et al., 1997), such as those in the field of electronics (Li et al., 2023) , in the textile industry (Meng et al., 2020), steelmaking (Pan et al., 2012), petrochemical industry (Zhang et al., 2020), and healthcare (Azaiez et al., 2022). A flexible flow shop environment consists of many sequential stages out of which at least one includes two or more parallel machines (Neufeld et al., 2023). Scheduling is more challenging in the FFSP environment and requires more complex algorithms than in a regular flow shop. Additionally, each task must follow the same processing path.

The FFSP and its variants have attracted the attention of many authors, and abundant literature was provided. Because of its powerful modelling capabilities, the FFSP and its variants prove to be an effective tool for modelling many different manufacturing processes. Sand casting industry FFSP problems with batches have been addressed in (Li et al., 2023) and an efficient heuristic was proposed. A FFSP problem is solved by considering human factors in (Liu et al., 2023). Reinforcement learning (RL) and genetic algorithms (GAs) are used to tackle the latter problem. A FFSP with two stages and maintenance requirements is considered in (Wang et al., 2023). To address this problem, both a genetic algorithm and an integer linear programming model are developed as potential solutions. Furthermore, a numerical example illustrates how the proposed models are effective and efficient.

Managing and scheduling steel production has become increasingly complex as the steel industry has advanced (Wegel et al., 2022; Zhao et al., 2020). In addition, today's firms have also been forced to optimise integrated operations instead of single stages due to fierce market competition. Since the steel industry has undergone intense consolidation over the past decade, this is particularly true today. This led to new research focusing on understanding how modern integrated steel production plants work and developing integrated production schedules rather than studying steelmaking operations separately. It was also in the midst of a steel industry consolidation that this shift in focus occurred. Consequently, integrated steel manufacturing processes are identified and considered. In (Jiang et al., 2023) authors studied a steelmaking-continuous casting problem which was modelled as a FFSP. An online scheduling algorithm and an integer linear program were proposed. An experimental study was carried out based on real data in order to compare the different proposed algorithms. Further on, more research papers for the FFSP in

https://doi.org/10.24846/v34i1y202504

the steel industry appeared (Neufeld et al., 2023, Colak and Keskin, 2021).

Cooling is one of the important operations in the steel-making industry (Bobba et al., 2023). There is a surprising lack of consideration for cooling time in the majority of FFSP scheduling in the steel industry, which refers to the duration of the cooling operation until a task is completed by a machine. The reason for this is that cooling usually takes less time than processing in several applications. Moreover, it is commonly believed that the duration of cooling does not significantly affect the production efficiency. Most research works on FFSP in the context of steel-making do not place as much emphasis on it. Moreover, it can be difficult to measure cooling time as compared to other parameters. To the best of the author's knowledge, there is no research on FFSP in the steel-making industry that takes into account cooling time.

This paper proposes a multi-stage flexible flow shop scheduling problem with cooling times (FFSPC), where the cooling time is considered independently of the processing time. In this scheduling problem, tasks are assigned to machines while accounting for the cooling operations. Optimisation procedures for solving this complex problem are proposed. These procedures enable an efficient scheduling in a flow shop environment.

Thus, this work provides a more robust solution to this problem and a more accurate representation of the production process. A classical FFSP is extended to solve the analysed scheduling problem. A distinct approach is needed for this extension since it differs from the classical FFSP approach.

It is worth mentioning that once a job enters the cooling phase in a machine, no additional jobs can be handled by that machine. In the steel industry, this is akin to the annealing process where a furnace must cool down after heating steel to a high temperature. During this cooling phase, the furnace cannot be used to heat another batch of steel, causing a halt in production until the process is complete. This is crucial for ensuring the quality and structural integrity of the final product (Kugelmeier et al., 2024).

Even with just two stages, FFSP features considerable theoretical difficulties because it is strongly NP-hard (Gupta, 1988). Moreover, when preemption is allowed, FFSP remains NP-hard (Gupta & Tunc, 1991). As a result, most FFSP variants are strongly NP-hard. FFSPC, the problem studied in this case, is NP-hard in a strong sense. An efficient heuristic based on two phases is proposed to deal with the latter problem. In this heuristic, the identical parallel machine scheduling problem is solved by considering release dates, cooling times, and delivery times. The first phase involves determining a feasible solution. In the second phase, the solution obtained in the first phase is improved. In addition, a family of lower bounds of the optimal solution's value is presented. These lower bounds are efficient in terms of closeness to the optimal solution as evidenced by the obtained numerical results. The proposed heuristic can be evaluated over a relative gap by using these lower bounds. A computational study is conducted based on benchmark test problems. The proposed lower bound along with the heuristic are both efficient and effective as a result of this computational study.

As a result of this study, the following contributions have been made:

The cooling time is fully taken into account during the scheduling and planning of tasks in the steel-making industry, for the first time to the best of one`s knowledge.

In this study, the problem's symmetry is highlighted as one of its key properties. The quality of the obtained solutions can be improved by utilising this property.

This research presents an effective two-phase heuristic that can produce optimal or nearly optimal solutions within a moderate computational time.

A new efficient lower bound was proposed, which makes it possible to evaluate the quality of the two-phase heuristic using the relative gap.

This research facilitates the modeling of real-world steel manufacturing systems.

The remainder of this paper is structured as follows. In Section 2, the problem is defined

and its symmetric property is presented. Section 3 refers to the lower bounds. In section 4, the proposed heuristic is described in detail. In Section 5, the results of an experimental study are presented. Finally, this paper is concluded and future research directions are outlined in Section 6.

2. Problem Definition

In this section, the studied problem was described in more detail. Several of its major characteristics are also explained in detail.

2.1 Problem Statement

The studied scheduling problem (FFSPC) is stated as follows. A shop composed of a set $ST = \{S_1, S_2, ..., S_K\}$ of K production stages in a sequence has to process a set $J = \{1, 2, ..., n\}$ of n tasks. Each stage S_i contains a set $M_i = \{M_{i,1}, M_{i,2}, ..., M_{i,m_i}\}$ of m_i parallel machines which are identical ($i = 1, 2, ..., K$). At least one stage contains more than two machines and $n > \max\{m_i, 1 \le i \le K\}$. Each task j has to be carried out from stage S_1 until stage S_K, following this order. This process is performed by only one machine in each stage. The processing duration for $j \in J$ in stage S_i is denoted by $pr_{i,j}$. The machine processing a task will not be accessible until it has fully cooled after processing that task and that task was removed following the completion of its processing. In stage S_i, the cooling time related to a task $j \in J$ is denoted by $cool_{i,j}$.

It is worth mentioning that there may be a delay related to the cooling of a machine and removing a certain task after it has been processed. In fact, the cooling operation is completely separate from the processing stage.

The schedule is generated under the following assumptions:

- Tasks cannot be preempted during the processing phase;
- At each stage, one machine processes a task at a time;
- A machine can only process one task at a time;
- Buffers have unlimited capacity between the two stages;
- Processing and cooling times are assumed to be positive integers;
- From time zero, all machines are available;
- Machines can be idle, waiting for jobs;
- From time zero, tasks are ready for processing.

The objective is finding a schedule σ^* that minimises the makespan $C_{\max}\left(\sigma^*\right) = \max_{j \in J}\left(C_{K,j}\left(\sigma^*\right)\right)$, where $C_{K,j}\left(\sigma^*\right)$ is the completion of the cooling task j in the last stage. Following the three-field notation (Brucker, 1998), the studied problem can be expressed as: $FH_K, \left(\left(PM^{(l)}\right)_{l=1}^{K}\right) \left| cool_{i,j} \right| C_{\max}$.

It should be recalled that in the three-field notation, the first field describes the machine environment. The second field is dedicated to the job characteristics. The objective function is presented in the third field. In this study, for the first field, *FH* stands for Hybrid Flowshop, K is the number of stages and *PM* indicates that in each stage there are parallel machines. For the second field, each task j is characterised by a cooling time $cool_{i,j}$ in stage S_i. In the third field, the objective function to be minimised is the maximum completion time denoted by $C_{\max}$. This problem is strongly NP-Hard since a particular case is NP-Hard in a strong sense (Gupta & Tunc, 1991).

It is worth mentioning that in most cases, an operator, who could be a human, a robot, or another type of equipment, performs the cooling task. The operator has the option to start the cooling operation just after the processing phase is completed, or he/it may choose to delay it without affecting the total duration of the process. This scheduling flexibility allows the operator to decide on the best timing to begin cooling. This enables him to perform other tasks in the meantime, thus enhancing machine productivity.

2.2 Symmetric Property

In this subsection, the studied problem is examined in terms of its symmetry. This leads to introducing a symmetric problem. In comparison with its symmetric counterpart, the studied problem has the same optimal solution. This led to a systematic exploration of the symmetric problem to improve the solution's quality.

Definition 1: The symmetric (or backward) problem is approached by scheduling in reverse order, starting from the final stage S_K and moving towards the initial one S_1. Scheduling from S_1 to S_K is the forward problem.

Definition 2: The notations for the symmetric problem are:

- The stages: $S_k^B = S_{K-k+1}$ and
- Number of machines: $m_k^B = m_{K-k+1}$.
- Machines: $M_{k,l}^B = M_{K-k+1,l}$.
- Processing time: $pr_{(k,j)}^B = cool_{K-k+1,j}$.
- Cooling time: $cool_{k,j}^B = pr_{K-k+1,j}$.
- Three-field notation:
$$FH_K, \left(\left(PM^{(l)}\right)_{l=K}^{1}\right) \left| cool_{k,j}^{B} \right| C_{\max}.$$

Below is an example of why it is important to investigate the symmetric problem:

Proposition 1: Each feasible schedule for the forward problem is automatically transformed into a feasible schedule for the symmetric problem. Moreover, both schedules have the same makespan. This is performed by keeping the same schedules for all stages and converting the time scale "t" for the forward problem into "t^B" for the symmetric problem, the formula $t^B = C_{\max} - t$ is used.

Proof. Let FS represent a schedule for the forward problem, and by keeping the same order of tasks for the machines, a schedule FS^B is created for the symmetric problem. Additionally, a time scale "t^B" is used for the symmetric problem, defined as $t^B = C_{\max} - t$, where "t" is the time scale for the forward problem. Clearly, FS and FS^B share the same critical path and, consequently, the same makespan. Moreover, by applying the previously mentioned method, a viable schedule for a symmetric problem can be adapted into a workable schedule for a forward problem.

The following example illustrates the symmetric problem:

Example 1: Consider $K = 2$, $n = 5$, and $m_1 = m_2 = 2$ The processing times $pr_{i,j}$, and $pr_{k,j}^B$ and the cooling times $cool_{i,j}$ ($i = 1, 2, ..., K$ and $j \in J$) $cool_{k,j}^B$ are presented in Table 1.

Table 1. Processing and cooling times for the forward problem and the symmetric problem

forward problem						symmetric problem					
j	1	2	3	4	5	j	1	2	3	4	5
$pr_{1,j}$	1	2	1	1	2	$pr_{1,j}^B$	1	1	2	2	2
$cool_{1,j}$	1	1	2	1	3	$cool_{1,j}^B$	2	1	2	3	1
$pr_{2,j}$	2	1	1	3	1	$pr_{2,j}^B$	1	1	2	1	3
$cool_{2,j}$	1	1	2	2	2	$cool_{2,j}^B$	1	2	1	1	2

The left side of Figure 1 illustrates a feasible schedule for the forward problem, while the right side shows the corresponding feasible schedule for the symmetric problem. As illustrated in Figure 1, even though the processing operation concludes at time 4, task 4 continues to block machine $M_{1,2}$. Time 7 marks the beginning of the cooling operation, and time 9 marks the completion $C_{1,4}$ of the task 4. During the time window [4;7], the machine is unavailable. From this, it can be concluded that the cooling operation

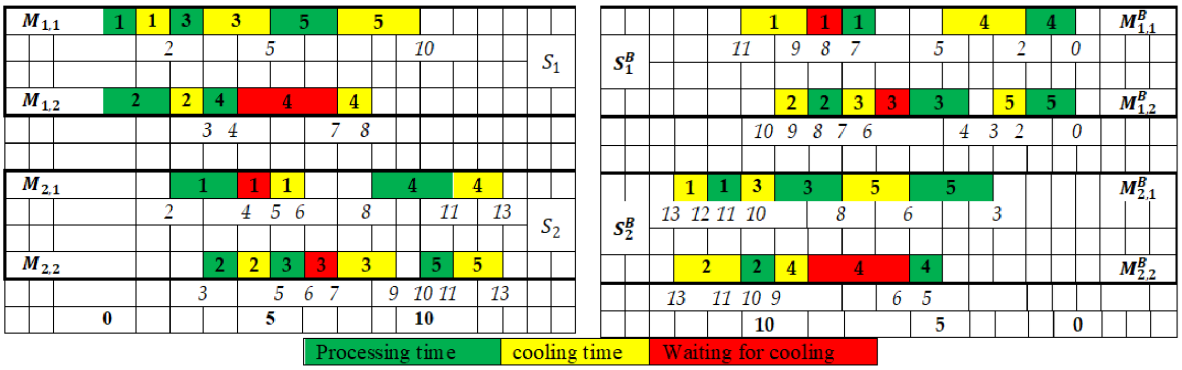

Figure 1. Gantt chart of a feasible schedule for the symmetric problem relative to example 1 (Hidri & Tlija, 2024)

does not depend on the processing time. In this feasible schedule σ, the cooling operation for job 4 is delayed for 3 units of time after finishing the processing operation.

The following relevant corollary follows from Proposition 1:

Corollary 1: The forward and symmetric problems have the same optimal makespan.

Proof. A direct result of Proposition 1.

Consequently, to enhance the quality of the obtained solution, the suggested procedures (the lower bounds and heuristic) inherently tackle the symmetric problem.

Useful notations and definitions for the rest of the paper are presented below.

For every stage S_k, a task $j \in J$ is associated with a release date $re_{k,j}$ and a delivery time $qe_{k,j}$, which are defined in the following manner:

$$\begin{cases} re_{k,j} = \sum_{i=1}^{k-1} \left(cool_{i,j} + pr_{i,j} \right) \text{ if } k > 1 \\ re_{k,j} = 0 \text{ if } k = 1 \end{cases} \quad (1)$$

$$\begin{cases} qe_{k,j} = \sum_{i=k+1}^{K} \left(cool_{i,j} + pr_{i,j} \right) \text{ if } k < K \\ qe_{k,j} = 0 \text{ if } k = K \end{cases} \quad (2)$$

3. Lower Bounds

In this section, a lower bound for the analysed problem is proposed. Lower bounds are also useful in assessing how effective the heuristic is based on measuring the relative gap between the upper and lower bounds. For all stages except for stage S_k, there is an unlimited number of machines available. This means relaxing the capacity constraints. Consequently, a task is handled immediately when it reaches any relaxed stage. In the unrelaxed stage S_k a parallel machine scheduling problem $P_m \mid re_{k,j}, cool_{k,j}, qe_{k,j} \mid C_{\max}$ is obtained. The parameters of this problem are the number of machines, denoted by $m = m_k$, the release dates, expressed by $re_{k,j}$, the cooling times, denoted by $cool_{k,j}$, and the delivery times, expressed by $qe_{k,j}$. If idle times are eliminated, the problem becomes a parallel machine scheduling problem which only includes release dates and delivery times, referred to as $P_m \mid re_{k,j}, qe_{k,j} \mid C_{\max}$. In this case, every task j has a processing time $pr_{k,j} + cool_{k,j}$.

(Carlier, 1987) introduced a lower bound for the problem $P_m \mid re_{k,j}, qe_{k,j} \mid C_{\max}$. This lower bound is calculated for each stage S_k and expressed by the following equation:

$$LB_k = \frac{1}{m_k} \left(\sum_{i=1}^{m_k} \overline{re_{k,j_{(i)}}} + \sum_{j=1}^{n} \left(cool_{k,j} + pr_{k,j} \right) + \sum_{i=1}^{m_k} \overline{qe_{k,j_{(i)}}} \right) \quad (3)$$

where $\overline{re_{k,j_{(i)}}}$ ($\overline{qe_{k,j_{(i)}}}$, respectively) is the i^{th} value in the list of $re_{k,j}$'s ($qe_{k,j}$'s, respectively) ($j = 1, ..., n$), which is sorted in ascending order.

Obviously, one obtains:

Corollary 2: The studied problem has the following valid lower bound:

$$LB = \max_{1 \le k \le K} \left\{ LB_k \right\} \quad (4)$$

$O(Kn)$ is the time complexity of this lower bound.

Proof. LB_k is a lower bound for the studied problem at stage S_k. Therefore,

$LB = \max_{1 \le k \le K} \left\{ LB_k \right\}$ is a valid lower bound for the addressed problem. $O(n)$ is the time complexity of LB_k. Repeating LB_k for K times, yields a time complexity of the form $O(Kn)$.

4. Heuristic Algorithm (TH)

The studied problem can be solved in a near-optimal manner using this heuristic. The initial feasible solutions will be obtained in the first phase (P1) of the heuristic. Further improvements will be made to the feasible solutions during the second phase (P2). In both phases, parallel machine scheduling problems ($P_m \mid r_j, cool_j, q_j \mid C_{\max}$) need to be solved. For each job there is a release date r_j, a cooling time $cool_j$, and a delivery time q_j. A variant of ($P_m \mid r_j, cool_j \mid L_{\max}$) that minimises maximum lateness must also be solved.

The Approximate Decomposition Algorithm with Cooling (ADAC) is a heuristic algorithm used in this study. Using ADAC, a near-optimal solution can be derived for the $P_m \mid r_j, cool_j, q_j \mid C_{\max}$ problem. NP-hardness is a well-known property of this problem. Notably, ADAC extends the Approximate Decomposition Algorithm (ADA), which was originally created to find approximate solutions for the problem $P_m \mid r_j, q_j \mid C_{\max}$ (Gharbi & Haouari, 2007). Taking cooling times into account is an important aspect of ADAC. During every iteration of the ADAC, the machines

with the shortest and longest completion times are selected. Further on, the parallel machine scheduling problem described in the previous step is solved. Iterations are halted when a stopping condition is reached. The experiments carried out indicate that, in most cases, the ADAC algorithm is fast and achieves optimal scheduling. Both phases (P1 and P2) are described in detail below.

4.1 Constructive Heuristic (P1)

First, a starting stage S_i $(1 \le i \le K)$ is selected and the associated parallel machine scheduling problem $(P_{m_i} \mid re_{i,j},\ cool_j,\ qe_{i,j} \mid C_{\max})$ is solved using the ADAC heuristic. Let $CC_{i,j}$ be the end time of the cooling operation of task j and the release date in the next stage S_{i+1} is set as $r_j = CC_{i,j}$. A parallel machine scheduling problem expressed as $(P_{m_{i+1}} \mid r_j,\ cool_j,\ q_j \mid C_{\max})$ is obtained and solved using the ADAC. The same procedure is repeated iteratively from stage S_{i+2} to stage S_K. Consequently, a partial schedule is obtained in stages $S_i, S_{i+1}, ..., S_K$. For stage S_{i-1} a due time for each task $j \in J$ is set as its starting processing time in stage S_i. A parallel machine scheduling problem $(P_{m_{i-1}} \mid re_{i-1},\ cool_j,\ d_j \mid L_{\max})$ is obtained and solved with ADAC. After solving the latter scheduling problem, the optimal maximum lateness in stage S_{i-1}, $(L^{i-1}_{\max})$ is either strictly positive $(L^{i-1}_{\max} > 0)$ or negative $(L^{i-1}_{\max} \le 0)$. The actions to be taken in these two cases are as follows:

1. If $L^{i-1}_{\max} > 0$, then the starting time of the processing operation must be right-shifted by $L^{i-1}_{\max}$ units of time in all stages $S_i, ..., S_K$.
2. If $L^{i-1}_{\max} \le 0$, then the starting time of the processing operation must be left-shifted by $L^{i-1}_{\max}$ units of time in all stages $S_i, ..., S_K$.

The same procedure is reiterated consecutively from stage S_{i-2} to stage S_1. The result is a feasible schedule γ_i with the makespan (upper bound) UB_i. When varying the starting stage S_i $(1 \le i \le K)$, K feasible schedules are obtained $(\gamma_1, ..., \gamma_K)$. The schedule with the minimum makespan value γ is selected to be the initial solution.

A self-contained figure (Figure 2) presents an example of the procedure with five stages and a starting stage S_4.

4.2 The Improvement Phase (P2)

A starting stage S_h is selected firstly. Based on the feasible schedule γ derived in Phase 1, an iterative sequence of improvements is developed. A convergence criterion must be met in order to stop the improvement phase. During the improvement phase, schedules are fixed in all stages except for one, that is (S_h). As part of the rescheduling process, a parallel machine scheduling problem $(P_{m_h} \mid r_j,\ cool_j,\ d_j \mid L_{\max})$ must be solved. The parameters of the latter problem are r_j, which denotes the release dates and is expressed by $r_j = CC_{h-1,j}$, where $CC_{h-1,j}$ is the final cooling time of task j in stage S_{h-1} and d_j, which denotes the due times $d_j = T_{h+1,j}$, where $T_{h+1,j}$ is the starting time for the processing of the task j in stage S_{h+1}. In the stage S_{K-1} the due times are $d_j = UB$. Two cases must be considered once ADAC has solved the problem:

- $L^h_{\max} < 0$: An improved solution has been found in this particular case at the stage S_h with the makespan $+L^h_{\max}$. As a result, a new schedule is generated by left-shifting all the tasks in the stages from S_h down to S_K, by $|L^h_{\max}|$ units of time;
- $L^h_{\max} \ge 0$: As a result, no improvement has been observed in this case.

Remark 1: $L^h_{\max} \le 0$ since γ is a feasible schedule.

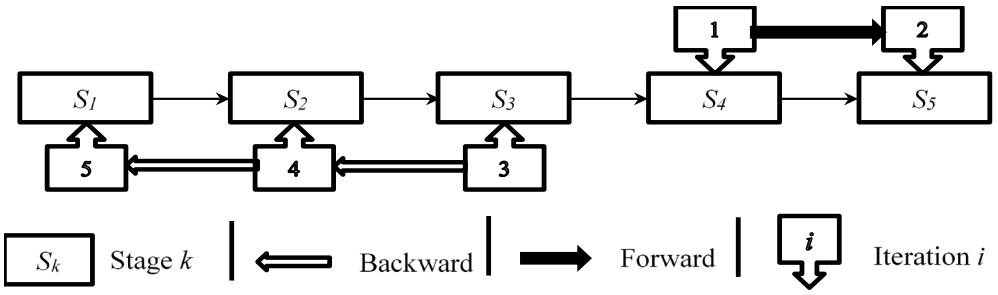

Figure 2. Illustration of the first phase (Phase 1)

Choosing a starting stage S_h and setting up a problem $P_{m_h} \mid r_j, cool_j, d_j \mid L_{\max}$ is the first step. Using the ADAC heuristic, this scheduling problem is solved. A rescheduling of stage S_h is performed if improvements are detected. The same procedure as for S_h is performed consecutively for the upstream stages $S_{h-1}, S_{h-2}, ..., S_1$. If an improvment is detected the respective stage is rescheduled and the improvement is propogated to the downstream stage(s). When reaching the first stage S_1, the improvement procedure is applied to the downstream stages $S_2, S_3, ..., S_K$ consecutively. The stopping condition for the improvement phase is set to $2K-2$ visited stages (problems) without improvement. The obtained schedule at the end of the latter procedure is denoted by Δ_h ($h = 1, ..., K$). Therefore, K schedules are proposed and the best self-contained one is kept and denoted by γ.

Remark 2: The symmetric problem (given in Definition 1) is solved using the proposed two-phase heuristic. By doing so it is possible to find better solutions.

5. Experimental Results

The test problems are generated as in (Vandevelde et al., 2005). Therefore, $K \in \{2,4,6,8,10\}$ and $n \in \{10,20,40,80\}$. The uniform distribution within [20,30] is used to generate the uniform processing times $P_{k,j}$. The stage-machine configurations are detailed in (Vandevelde et al., 2005). Cooling times are uniformly generated within [1,10] (Type 1), [20,40] (Type 2), and [20,60] (Type 3), respectively. It is crucial to evaluate the cooling times relative to the processing times for each type above. In the case of Type 1, the cooling time is comparatively less important than the processing time.

Five instances are generated for each combination of n, cooling Type, and stage-machine configurations (S_k;m_k), leading to a total number of 1800 instances. The testbed encompasses a wide variety of problem sizes, machine distribution patterns, as well as different processing and cooling times. Consequently, it offers a high level of diversity, enabling an impartial assessment of the proposed procedures.

5.1 The Performance of the Two-phase Heuristic

The proposed two-phase heuristic was developed and tested using the 1800 instances mentioned previously. Table 2 includes the global results obtained for the three cooling Types by employing the proposed heuristic algorithm (TH). Each instance also has a relative gap rg calculated as $rg = 100 \times \frac{UB - LB}{LB}$, where LB and UB represent respectively the lower bound (presented in section 3) and the makespan obtained by the heuristic **TH**. Furthermore, for each class of instances the average relative gap is calculated. Three metrics are used to measure the performance of the employed heuristic, namely **MG –** the average relative gap for a class of instances, **MaxG –** the maximal relative gap within a class of instances, and **MT** - the average computational time (CPU) in seconds (s) for a class of instances.

Table 2. Global results obtained for the three cooling Types by employing the proposed heuristic algorithm (TH)

	MT	MG	MaxG
Type 1	8.88	1.16	9.76
Type 2	14.86	0.77	8.17
Type 3	12.97	1.27	10.05
All types	12.24	1.07	10.05

By employing the proposed two-phase heuristic, effective solutions are generated, as evidenced by the results included in Table 2, which demonstrates its effectiveness. In fact, the average required CPU time is 12.24 seconds and the average gap is 1.07%. For Types 1 and 3, the average relative gaps are 1.16% and 1.27%, respectively. Type 2 is the easiest of the three types of test problems since the average relative gap is just 0.77%. In terms of test difficulty, Type 3 is the most challenging. Accordingly, Type 3 features the highest average relative gap, that is 1.27% and the longest average CPU time, that is 12.97s. For Type 3, the processing time is less important than the cooling time.

5.1.1 The Effect of Implementing the Second Phase (P2)

To gain a clearer understanding of the achievements related to Phase 2, a comparison

between Phase 2 and Phase 1 is carried out. Table 3 includes the obtained results. According to them, it is evident that in Phase 2 the quality of the obtained solution was improved. For example, if Type 3 is considered, in phase 2 the *MG* and *MaxG* were reduced from 1.38% and 10.23%, respectively to 1.27% and 10.05%, respectively, within just 6 seconds.

Table 3. A comparison between P1 and P2 based on the employed cooling Types

		MT	MG	MaxG
Type 1	P1	8.23	1.23	9.76
	P2	8.88	1.16	9.76
Type 2	P1	14.22	0.82	8.17
	P2	14.86	0.77	8.17
Type 3	P1	12.14	1.38	10.23
	P2	12.97	1.27	10.05
All types	P1	11.53	1.14	10.23
All types	P2	12.24	1.07	10.05

A second pairwise comparison is made for PH1 and PH2 by using the three above-mentioned metrics and the following cases are obtained:

- ($PH2 < PH1$): the percentage of time when PH2 dominates PH1.
- ($PH2 = PH1$): the percentage of time when PH2 and PH1 do not differ.

The results of this pairwise comparison are included in Table 4. This table shows that PH2 strictly dominates PH1 in 18.72% of instances. In addition, there is a slight advantage for Type 3 in comparison with Types 1 and 2 with a percentage of 20.17%.

Table 4. Pairwise comparison between PH1 and PH2 based on the employed cooling Types

	PH2 < PH1	PH2 = PH1
Type 1	17.83	82.17
Type 2	18.17	81.83
Type 3	20.17	79.83
All types	18.72	81.28

5.1.2 The effect of Implementing the Symmetric Problem

By contrasting the forward problem of Phase 2 (PH2D) with the symmetric problem (PH2S), the effect of implementing the symmetric problem, as outlined in Definition 1, is evaluated. Tables 5 and 6 display the overall findings. The implementation of the symmetric problem leads to a notable decrease in both the average and maximum relative gaps, as illustrated in Table 5.

Table 5. A comparison between PH2D and PH2S based on the employed cooling Types

		MT	MG	MaxG
Type 1	PH2D	4.68	1.45	10.00
	PH2S	4.21	1.16	9.76
Type 2	PH2D	7.47	0.95	8.17
	PH2S	7.39	0.77	8.17
Type 3	PH2D	6.82	1.58	10.40
	PH2S	6.14	1.27	10.05

Table 6 includes the following three cases:

- ($PH2D < PH2S$): the percentage of time when PH2D strictly dominates PH2S.
- ($PH2D = PH2S$): the percentage of time when PH2D = PH2S.
- ($PH2D > PH2S$): the percentage of time when PH2S strictly dominates PH2D.

Table 6. Pairwise comparison between PH2D and PH2S based on the employed cooling Types

	PH2D > PH2S	PH2D = PH2S	PH2D < PH2S
Type 1	39.00	40.00	21.00
Type 2	38.17	39.50	22.33
Type 3	38.83	37.50	23.67
All types	38.67	39.00	22.33

Table 6 shows that the implementation of the symmetric problem improves the quality of the obtained solution in 38.67% of all cases. In this context, it can be noticed that the solutions for the colling Types 2 and 3 after implementing the symmetric procedure are improved by 38.17% and 38.83%, respectively. Therefore, this study illustrates the benefits of exploring symmetric problems.

6. Conclusion

In this work, the multi-stage flexible flow shop scheduling problem, particularly concerning the cooling operations in the steel industry, is addressed. The cooling time is generally ignored

in the existing literature. Also, the tackled problem is NP-Hard in a strong sense. Thus, a heuristic algorithm is introduced, alongside a lower bound. This lower bound involves relaxing the capacity constraints at all stages except for one. The proposed two-phase heuristic provided a near-optimal solution for the studied problem. Further on, an iterative process of solving a parallel machine scheduling problem is required for implementing this heuristic. So, the parallel machine scheduling problem is solved based on a newly adapted algorithm referred to as ADAC.

An extensive experimental study is conducted based on benchmark test problems to evaluate the performance of the proposed lower bound and heuristic. It can be concluded that the proposed procedures are effective based on the obtained results. The problem becomes more difficult to solve when the cooling time is more important than the processing time (cooling Type 3). According to the obtained computational results, the proposed two-phase heuristic consistently provides high-quality production schedules.

In future research, another approach, such as metaheuristics, may be explored to provide a better near-optimal solution within a reasonable CPU time. To that, based on the already proposed procedures, several variants of the studied problem can be explored as including release dates in the first stage and delivery times in the last stage for different tasks.

Acknowledgement

The author extends his appreciation to King Saud University for funding this work through Researchers supporting project number (RSPD2025R687), King Saud University, Riyadh, Saudi Arabia.

REFERENCES

Azaiez, M.-N., Gharbi, A., Kacem, I., Makhlouf, Y. & Masmoudi, M. (2022) Two-stage no-wait hybrid flow shop with inter-stage flexibility for operating room scheduling. *Computers & Industrial Engineering*. 168, 108040. https://doi.org/10.1016/j.cie.2022.108040.

Billaut, J.-C., Tacquard C. & Martineau, P. (1997) Modeling FMS Scheduling Problems As Hybrid Flowshop Scheduling Problems. *Studies in Informatics and Control*. 6, 25-30.

Bobba, S, Babu, B. H., Rao, M. S. & Leman, Z. (2023) The consequences of hot deformation and control cooling on the microstructure of medium carbon microalloyed steel − 38MnSiVS5 grade. *Materials Today: Proceedings.* https://doi.org/10.1016/j.matpr.2023.04.290.

Brucker, P. (1998) Some Problems in Combinatorial Optimization. In: *Scheduling Algorithms*. Berlin, Heidelberg, Germany, Springer, pp. 11-36.

Carlier, J. (1987) Scheduling jobs with release dates and tails on identical machines to minimize the makespan. *European Journal of Operational Research*. 29(3), 298-306. https://doi.org/10.1016/0377-2217(87)90243-8.

Colak, M. & Keskin, G. A. (2022) An extensive and systematic literature review for hybrid flowshop scheduling problems. *International Journal of Industrial Engineering Computations*. 13(2), 185-222.

Gharbi, A & Haouari, M. (2007) An approximate decomposition algorithm for scheduling on parallel machines with heads and tails. *Computers & Operations Research*. 34(3), 868-883. https://doi.org/10.1016/j.cor.2005.05.012.

Gupta, J. N. D. (1988) Two-Stage, Hybrid Flowshop Scheduling Problem. *Journal of the Operational Research Society*. 39(4), 359-364. https://doi.org/10.2307/2582115.

Gupta, J. N. D. & Tunc, E. A. (1991) Schedules for a two-stage hybrid flowshop with parallel machines at the second stage. *International Journal of Production Research*. 29(7), 1489-1502. https://doi.org/10.1080/00207549108948025.

Hidri, L. & Tlija, M. (2024) Multi-stage hybrid flow shop scheduling problem with lag, unloading, and transportation times. *PeerJ Computer Science*. 10, e2168. https://doi.org/10.7717/peerj-cs.2168.

Jiang, S.-L., Xu, C., Zhang, L. & Ma, Y. (2023) A decomposition-based two-stage online scheduling approach and its integrated system in the hybrid flow shop of steel industry. *Expert Systems with Applications*. 213(C), 119200. https://doi.org/10.1016/j.eswa.2022.119200.

Kugelmeier, C. L., Silva, R., Filho, A. A. M., Magnabosco, R., Kliauga, A. M. & Rovere, C. A. D. (2024) Gleeble® simulation of the heat-affected zones in lean duplex stainless steel 2404: Understanding the effect of cooling rates on microstructure evolution and pitting corrosion resistance. *Journal of Materials Research and Technology*. 33, 7518-7533. https://doi.org/10.1016/j.jmrt.2024.11.118.

Li, X., Guo, X., Tang, H., Wu, R. & Liu, J. (2023) An improved cuckoo search algorithm for the hybrid flow-shop scheduling problem in sand casting enterprises considering batch processing. *Computers & Industrial Engineering*. 176, 108921. https://doi.org/10.1016/j.cie.2022.108921.

Liu, Y., Shen, W., Zhang, C. & Sun, X. (2023) Agent-based simulation and optimization of hybrid flow shop considering multi-skilled workers and fatigue factors. *Robotics and Computer-Integrated Manufacturing*. 80, 102478. https://doi.org/10.1016/j.rcim.2022.102478.

Meng, L., Zhang, C., Shao, X., Zhang, B., Ren, Y. & Lin, W. (2020) More MILP models for hybrid flow shop scheduling problem and its extended problems. *International Journal of Production Research*. 58(13), 3905-3930. https://doi.org/10.1080/00207543.2019.1636324.

Neufeld, J. S., Schulz, S. & Buscher, U. (2023) A systematic review of multi-objective hybrid flow shop scheduling. *European Journal of Operational Research*. 309(1), 1-23.

Pan, Q.-K., Wang, L., Mao, K, Zhao, J.-H., & Zhang, M. (2012) An Effective Artificial Bee Colony Algorithm for a Real-World Hybrid Flowshop Problem in Steelmaking Process. *IEEE Transactions on Automation Science and Engineering*. 10(2), 307-322. https://doi.org/10.1109/TASE.2012.2204874.

Vandevelde, A., Hoogeveen, H., Hurkens, C. A. J. & Lenstra, J. K. (2005) Lower Bounds for the Head-Body-Tail Problem on Parallel Machines: A Computational Study of the Multiprocessor Flow Shop. *INFORMS Journal on Computing*. 17(3), 305-320. https://doi.org/10.1287/ijoc.1040.0082.

Wang, Z., Deng, Q., Zhang, L., Li, H. & Li, F. (2023) Joint Optimization of Integrated Mixed Maintenance and Distributed Two-Stage Hybrid Flow-Shop Production for Multi-Site Maintenance Requirements. *Expert Systems with Applications*. 215, 119422. https://doi.org/10.1016/j.eswa.2022.119422.

Wegel, S., Volling, T. & Sahling, F. (2022) Developing a Matheuristic for the Integrated Planning of a Cold Rolling Steel Plant. *IFAC-PapersOnLine*. 55(10), 1231-1236. https://doi.org/10.1016/j.ifacol.2022.09.558.

Zhang, B., Pan, Q.-K., Gao, L., Meng, L.-L., Li, X.-Y. & Peng, K.-K. (2020) A Three-Stage Multiobjective Approach Based on Decomposition for an Energy-Efficient Hybrid Flow Shop Scheduling Problem. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*. 50(12), 4984-4999. https://doi.org/10.1109/TSMC.2019.2916088

Zhao, Z., Zhou, M., Liu, S., Guo, X. & Liu, H. (2020) A Lexicographic Bi-objective Scheduling Problem From Steel Production Systems. *IFAC-PapersOnLine*. 53(5), 158-163. https://doi.org/10.1016/j.ifacol.2021.04.140.