AIMS Mathematics, 11(7): 19545–19566.
DOI: 10.3934/math.2026794
Received: 01 March 2026
Revised: 23 June 2026
Accepted: 24 June 2026
Published: 02 July 2026

https://www.aimspress.com/journal/Math

Research article

An ODE-augmented matheuristic for the permutation flow shop scheduling problem with thermally driven processing times

Neus Garrido[1], Sandra Oltra-Crespo[2], Lucía Agud-Albesa[3], Daniel López-Rodríguez[3] and Ángel A. Juan[4,5,*]

[1] Instituto de Matemática Multidisciplinar, Universitat Politècnica de València, 46022 Valencia, Spain

[2] Technological Institute of Informatics, Universitat Politècnica de València, 03801 Alcoy, Spain

[3] Dept. of Applied Mathematics, Universitat Politècnica de València, 03801 Alcoy, Spain

[4] CIGIP – ValgrAI, Universitat Politècnica de València, 03801 Alcoy, Spain

[5] UNIE Universidad, Av. Monforte de Lemos 28, 28029 Madrid, Spain

* **Correspondence:** Email: ajuanp@upv.es.

Abstract: This work analyzes a dynamic extension of the permutation flow shop scheduling problem (PFSP) in which processing times depend on the evolution of machine states. The proposed matheuristic combines a constructive heuristic, a biased randomization strategy, and a system of ordinary differential equations (ODEs) that models the thermal behavior of machines. This integration allows scheduling decisions to interact directly with time-varying machine conditions, producing schedules that adapt to operational degradation effects. The proposed ODE-enhanced matheuristic is evaluated on classical benchmark instances ranging from 20 to 100 jobs and up to 20 machines. Processing times are modified through a state-dependent thermal model that introduces performance degradation as machine load increases. The computational study compares static schedules, the dynamic evaluation of those schedules, and the solutions obtained by the proposed method. The results show that ignoring thermal effects leads to a substantial increase in makespan. In contrast, the proposed matheuristic consistently mitigates this degradation, achieving average improvements of 36.2%, 26.4%, and 25.9% with respect to the dynamically evaluated static solutions. Furthermore, the proposed approach reduces the dispersion of makespan values across instances, indicating improved robustness under thermal fluctuations.

Keywords: permutation flow shop problem; dynamic processing times; matheuristics; first-order differential equations; thermodynamics; thermal stress
Mathematics Subject Classification: 68W20, 90-10

1. Introduction

The flow shop scheduling problem (FSP) is a classical problem in operations research and industrial engineering. In its standard form, the objective is to determine the optimal order of n jobs processed on m machines arranged in a series, typically minimizing the makespan C_{max}. A widely studied variant is the permutation flow shop scheduling problem (PFSP), where the same job sequence is imposed on all machines. This restriction reduces the search space from $(n!)^m$ to $n!$, while preserving the combinatorial complexity and practical relevance of the problem. The PFSP has been extensively studied due to its applicability in manufacturing systems and its role as a benchmark problem in scheduling research [1, 2].

Most classical PFSP formulations assume that processing times are fixed and known in advance. This assumption is often unrealistic in modern production environments. In practice, machine performance may change during execution due to heat accumulation, wear, or workload intensity. These effects are documented in deteriorating and time-dependent scheduling models, where processing times evolve over time [3]. In high-intensity systems such as automated machining, additive manufacturing, or continuous production lines, such variability can lead to significant deviations between planned and realized schedules [4]. As a result, schedules that are optimal under static assumptions may lose efficiency when executed under real operating conditions. Several extensions of the PFSP attempt to address this limitation. Deteriorating and time-dependent models define processing times as explicit functions of the time or job position in the sequence [5, 6]. Stochastic and simheuristic approaches incorporate uncertainty through probability distributions and simulation-based evaluation [7]. However, in most of these approaches, variability is exogenous. Processing times evolve independently of the scheduling decisions, and therefore do not capture feedback effects between machine usage and system condition. In many real systems, this assumption is restrictive. Machine states evolve as a consequence of the executed schedule, and this evolution directly affects future processing times. This type of endogenous dynamic behavior is not explicitly addressed in standard PFSP formulations.

This paper proposes a dynamic PFSP formulation in which processing times depend on the internal state of each machine. The state is represented by a variable $s_i(t)$, interpreted as the thermal condition of machine i, whose evolution is governed by an ordinary differential equation (ODE). Under this framework, the effective processing time of job j on machine i is defined as:

$$p_{ij}^t = g_i(s_i, t)p_{ij}^0, \tag{1.1}$$

where p_{ij}^0 denotes the nominal processing time and $g_i(s_i, t)$ is a state-dependent function. In contrast to classical time-dependent models, processing times are not prescribed as explicit functions of time. Instead, they are determined by the trajectory of the machine state, which depends on workload, heat dissipation, and operational history. This introduces a feedback mechanism in which scheduling decisions influence machine conditions, and these conditions in turn affect subsequent processing times (Figure 1). To solve this problem, we propose an ODE-augmented matheuristic that integrates continuous system dynamics with discrete scheduling decisions. The approach combines a Nawaz–Enscore–Ham (NEH)-based constructive heuristic [8] with a biased randomization mechanism [9] and an ODE-based evaluation of machine states. The method evaluates candidate schedules by dynamically updating processing times according to the solution of the differential

equations, allowing the search process to account for thermal effects.

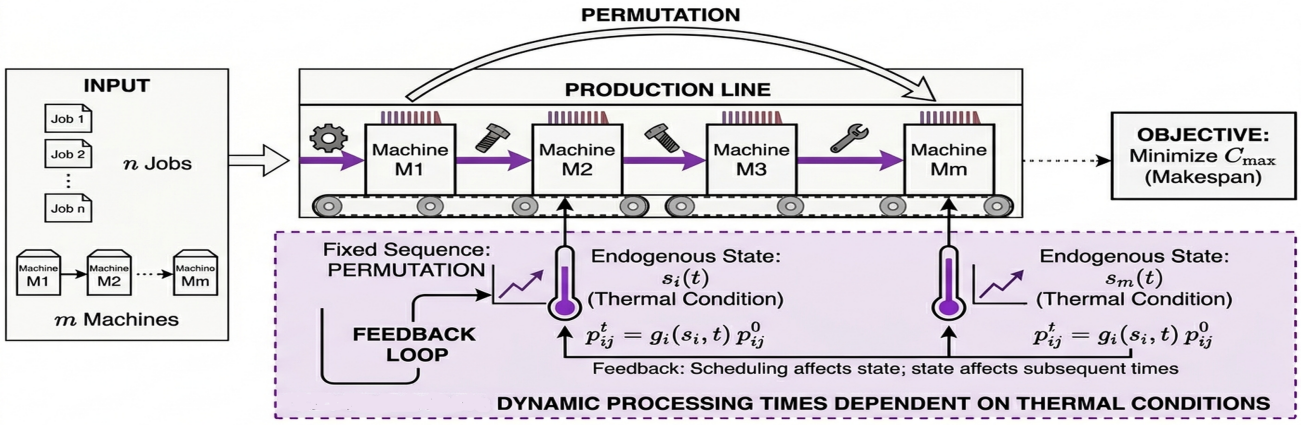

Figure 1. Schema of the PFSP with thermally driven processing times.

The main contributions of this work are as follows: *(i)* a state-dependent PFSP formulation is introduced, where processing times are determined by an endogenous dynamic model rather than by exogenous assumptions; *(ii)* an ODE-augmented matheuristic is developed to efficiently explore the solution space under these conditions; and *(iii)* a validation framework based on Taillard benchmark instances [10] is proposed, where static solutions are used to define reference bounds and to quantify the impact of dynamic effects. In particular, a baseline sequence S_0 obtained with a biased randomization algorithm (BRA) defines a soft lower bound $C_{max}(S_0, P^0)$, while its dynamic evaluation $C_{max}(S_0, P^t)$ provides a soft reference for the efficiency loss due to thermal effects. The proposed ODE-augmented matheuristic generates solutions S^* such that:

$$C_{max}(S_0, P^0) \leq C_{max}(S^*, P^t) \leq C_{max}(S_0, P^t), \tag{1.2}$$

which shows its ability to improve schedule quality under dynamic conditions and contributes to its validation process. Note that P^0 means the static processing times and P^t the dynamic processing times.

The remainder of the paper is organized as follows. Section 2 reviews the related literature. Section 3 presents the problem formulation and notation. Section 4 describes the proposed ODE-augmented matheuristic. Section 5 reports the computational experiments and results, which are then analyzed in Section 6. Finally, Section 7 concludes the paper and outlines future research directions.

2. Related work

Research on the PFSP had traditionally assumed deterministic and static processing times. Under this setting, a large body of work focused on the design of efficient heuristics and metaheuristics to minimize makespan, including constructive heuristics, iterated greedy methods, and hybrid metaheuristics [11]. These approaches provided high-quality solutions, but they relied on the assumption that processing times remained constant throughout execution. This assumption limited their applicability in environments where machine performance evolved during operation. To address this limitation, several studies introduced dynamic extensions of the PFSP in which processing times

changed during the scheduling horizon. One of the earliest and most studied approaches considered deteriorating jobs, where processing times increased as a function of job position or starting time. Different deterioration patterns, including linear, polynomial, and exponential forms, were discussed in [3]. In flow shop settings, position-dependent learning effects were analyzed in [12], while combined learning and deterioration effects under no-idle constraints were studied in [13]. More recent work incorporated energy efficiency and learning mechanisms, where processing times decreased with accumulated experience or depended on job sequencing decisions [14]. Although these models introduced temporal variability, they shared a common limitation: Processing times were defined as deterministic functions of time or sequence position. As a result, their evolution was exogenous and did not depend on the actual system state induced by the schedule.

Another research stream considered stochastic PFSP formulations, where uncertainty in processing times was modeled through probability distributions. In this context, simheuristics combined metaheuristic search with simulation-based evaluation. Candidate solutions were typically assessed over multiple simulation runs to estimate their performance under uncertainty [15]. Extensions incorporated additional objectives beyond makespan, improving robustness and decision support [7, 16]. Recently, fuzzy simheuristics have emerged as a hybrid methodology capable of simultaneously managing both stochastic and fuzzy uncertainty [17]. Despite these advances, variability remained externally imposed through probabilistic assumptions, and processing times did not depend on the system state generated by scheduling decisions. Related approaches included robust and reactive scheduling. Robust models aimed to identify solutions that performed well across different scenarios [18], while reactive methods adjusted schedules in response to disruptions such as machine breakdowns or delays [19, 20]. Maintenance-aware scheduling also incorporated reliability considerations, but system degradation was usually represented through discrete events or probabilistic failures rather than continuous state variables driven by workload.

A closer line of research was found in state-dependent and degradation-aware scheduling models, where processing conditions evolved as a function of system usage. Early contributions introduced deterioration effects linked to job processing, where operation times increased as a function of job position or processing order [21]. More general condition-based and degradation-aware models considered systems whose state evolved continuously and affected operational performance [22]. These approaches captured the dependence between system usage and performance, but they were mainly developed in reliability and maintenance contexts rather than in permutation flow shop scheduling. In addition, degradation was often modeled through simplified functions or stochastic processes, without explicitly embedding continuous-time physical dynamics into the scheduling evaluation.

Hybrid and simulation-based approaches provided another way to represent complex production environments. Discrete-event simulation (DES) has been combined with optimization techniques to evaluate scheduling policies under realistic conditions, including stochastic processing times and resource interactions. For instance, tabu search has been integrated with simulation to account for system variability during optimization [23]. More recent work has combined mixed-integer linear programming with DES through decomposition techniques, allowing detailed modeling of operational constraints [24]. In dynamic settings, simulation has also been coupled with machine learning to incorporate predictive maintenance into scheduling decisions [25]. These methods provided realistic performance estimates, but system dynamics were typically handled during the evaluation phase

rather than embedded directly into the optimization model. Notice that these research directions share a common limitation: Processing time variability is either prescribed as a function of time or position, modeled through stochastic distributions, or handled through simulation. In all cases, the evolution of processing times is largely decoupled from scheduling decisions. The system does not exhibit endogenous dynamics where machine conditions evolve as a consequence of workload and, in turn, influence future processing times. In contrast, real manufacturing systems often display strong feedback effects between machine usage and performance. Thermal accumulation, energy consumption, and wear depend on operational history and directly affect processing efficiency.

Besides, PFSP research has also been driven by the growing relevance of distributed manufacturing systems. In these environments, the geographical dispersion of production facilities significantly increases problem complexity by incorporating decisions concerning the allocation of production orders among factories alongside traditional scheduling considerations, thereby requiring more integrated coordination mechanisms across multiple production sites [26, 27]. Metaheuristic approaches remain the primary solvers for these NP-hard configurations, with evolutionary algorithms and iterated greedy algorithms [26]. Other established methodologies include teaching-learning-based optimization [28], genetic algorithms [29], and memetic algorithms utilizing meta-Lamarckian learning [30]. Efficiency in these frameworks is typically achieved through local search operators such as swap and insert, ruin-and-recreate strategies within adaptive large neighborhood search frameworks, and the integration of reinforcement learning, specifically Q-learning, to dynamically select search strategies [26, 27]. Regarding operational dynamism, existing models address uncertainties like machine breakdowns and variable processing times through stochastic programming, fuzzy logic to handle linguistic imprecision [31,32], or scenario-based robust optimization [27,33]. It is worth noting that these approaches are developed for distributed manufacturing.

In contrast to these distributed paradigms, our study addresses the fundamental single-factory PFSP configuration, allowing the analysis of processing time evolution. Furthermore, while the current literature relies on discrete or fuzzy modeling, this paper introduces a novel methodology by characterizing dynamic processing times through ODEs. This continuous mathematical modeling provides a precise representation of time-dependent variations, filling a critical research gap in dynamic scheduling identified in recent systematic reviews [26]. In this framework, the processing time of each operation depends on the instantaneous machine state, which evolves according to a continuous dynamic model driven by workload and dissipation effects. Processing times are not defined exogenously but emerge from the interaction between scheduling decisions and system dynamics. Table 1 summarizes the key characteristics of existing PFSP variants. The comparison highlights how different approaches incorporate variability in processing times, whether such variability is treated as endogenous or exogenous, and the corresponding modeling frameworks.

Table 1. Summary of related work on PFSP variants and positioning of this study.

Category	Works	Variability	Endogenous	Modeling approach
Static PFSP	[11]	None	No	Deterministic times
Deteriorating / learning	[3, 12–14]	Time / position	No	Prescribed functions
Stochastic PFSP	[15–17]	Random	No	Simulation + metaheuristics
Fuzzy PFSP	[17]	Vagueness / non-probabilistic	No	Fuzzy sets + metaheuristics
Robust / reactive	[18–20]	Disruptions	Partially	Scenario-based / rescheduling
Degradation-aware	[21, 22]	Usage-dependent	Limited	Analytical / reliability models
Simulation-based	[23–25]	Complex / hybrid	No	DES + optimization
This work	—	State-dependent	Yes	ODE + matheuristic

3. Problem formulation

The classical PFSP is defined by the requirement to determine an optimal sequence for a set J of n jobs, $J = \{1, 2, \ldots, n\}$, which must be processed on a set M of m machines, $M = \{1, 2, \ldots, m\}$. The operational flow is governed by a strict technological precedence where each job follows a unidirectional path, starting on the first machine and continuing sequentially until the final operation on machine m is completed. In a PFSP, all machines follow the same job permutation. The feasibility of the schedule is constrained by the synchronization between machine and job availability: an operation of job $j \in J$ on machine $i \in M$ can only start after the machine has completed the previous operation and the job has finished processing on machine $i - 1$. The classical PFSP assumes that machines are continuously available, failures do not occur, jobs are available at time zero, interruptions are not allowed once processing starts, and intermediate buffers have unlimited capacity [34].

Let $p_{i,j}$ denote the processing time of job j on machine i. The following binary variable defines the position of each job in the permutation:

$$X_{k,j} = \begin{cases} 1, & \text{if job } j \text{ is assigned to position } k \\ 0, & \text{otherwise} \end{cases} \qquad \forall k, j \in \{1, \ldots, n\}, \tag{3.1}$$

where k represents the position in the permutation and not a specific job. Let $C_{i,k}$ denote the completion time on machine i of the job assigned to position k, with $i \in \{1, \ldots, m\}$ and $k \in \{1, \ldots, n\}$. The variable C_{max} represents the makespan of the complete schedule. The position-based mathematical formulation of the PFSP is therefore:

$$\text{Minimize: } Z = C_{max}, \tag{3.2}$$

subject to:

$$\sum_{j=1}^{n} X_{k,j} = 1 \qquad k = \{1, \ldots, n\}, \tag{3.3}$$

$$\sum_{k=1}^{n} X_{k,j} = 1 \qquad j = \{1, \ldots, n\}, \tag{3.4}$$

$$C_{1,1} = \sum_{j=1}^{n} X_{1,j} p_{1,j}, \tag{3.5}$$

$$C_{1,k} \geq C_{1,k-1} + \sum_{j=1}^{n} X_{k,j} p_{1,j} \qquad k = \{2, \ldots, n\}, \tag{3.6}$$

$$C_{i,k} \geq C_{i-1,k} + \sum_{j=1}^{n} X_{k,j} p_{i,j} \qquad i = \{2, \ldots, m\},\ k = \{1, \ldots, n\}, \tag{3.7}$$

$$C_{i,k} \geq C_{i,k-1} + \sum_{j=1}^{n} X_{k,j} p_{i,j} \qquad i = \{1, \ldots, m\},\ k = \{2, \ldots, n\}, \tag{3.8}$$

$$C_{max} \geq C_{m,k} \qquad k = \{1, \ldots, n\}, \tag{3.9}$$

$$C_{i,k} \geq 0 \qquad i = \{1, \ldots, m\},\ k = \{1, \ldots, n\}, \tag{3.10}$$

$$X_{k,j} \in \{0, 1\} \qquad j, k = \{1, \ldots, n\}. \tag{3.11}$$

Equations (3.3) and (3.4) guarantee that every job is assigned to exactly one position and that every position contains exactly one job, generating a feasible permutation schedule. Constraint (3.5) initializes the schedule because the first job in the permutation starts at time zero on the first machine. Constraint (3.6) imposes the processing order of the permutation on the first machine. Constraints (3.7) impose the technological precedence between consecutive machines, ensuring that each job can only start on machine i after completing its operation on machine $i - 1$. Constraints (3.8) enforce the machine capacity restriction by preventing overlap between consecutive positions on the same machine. Constraint (3.9) links the makespan variable with the completion times on the last machine. The remaining constraints define the domains of the decision variables. This position-based formulation represents the sequence through the assignment variables $X_{k,j}$. Therefore, the order between jobs is implicitly defined by their assigned positions in the permutation. Consequently, the pairwise disjunctive constraints commonly modeled through big-M formulations are replaced by position-based recursive completion-time constraints. Notice that the completion times are recursively computed according to the selected permutation. Since the objective minimizes C_{max}, the inequalities defining completion times become tight at optimality. The dynamic extension proposed in this work evaluates each feasible permutation by incorporating the evolution of the machine states into the processing times.

The formulation above describes the static PFSP structure. In the proposed dynamic extension, the fixed processing time $p_{i,j}$ is replaced by the nominal processing time $p^0_{i,j}$, while the state-dependent processing time $p^t_{i,j}$ is obtained dynamically from the thermal state of each machine. Let $s_i(t)$ denote the state of machine i at time t. The evolution of this state is described by:

$$\frac{ds_i(t)}{dt} = f(s_i(t), \alpha_i, \beta_i, u_i(t)), \tag{3.12}$$

where α_i and β_i are machine-specific parameters and $u_i(t)$ represents the operational state of machine i. Consequently, the processing time becomes state-dependent and is denoted as $p^t_{i,j}$. In this work,

the machine state is represented by temperature such that $s_i(t) = T_i(t)$. Therefore, the processing time depends on the thermal condition at the beginning of each operation:

$$p^t_{i,j} = g(T_i(t_{start}))p^0_{i,j}, \tag{3.13}$$

where $p^0_{i,j}$ is the nominal processing time, $g(\cdot)$ is the thermal penalty function, and $T_i(t_{start})$ is the machine temperature when the operation begins. The binary activation function is defined as

$$u_i(t) = \begin{cases} 1, & \text{machine } i \text{ is processing a job,} \\ 0, & \text{machine } i \text{ is idle.} \end{cases} \tag{3.14}$$

The thermal evolution of machine i follows a first-order differential equation based on Newton's law of cooling:

$$\frac{dT_i}{dt} = \alpha_i(T_{env} - T_i) + \beta_i u_i, \tag{3.15}$$

where T_{env} is the environmental temperature, α_i represents the heat dissipation capacity, and β_i represents the heat generated during active processing. As the machine temperature increases, the processing time increases through the thermal penalty function:

$$p^t_{i,j} = (1 + 0.01 \max\{0, T_i(t_{start}) - T_{env}\})\, p^0_{i,j}. \tag{3.16}$$

The coefficient 0.01 controls the sensitivity of processing times to temperature variations. It provides a moderate coupling between machine temperature and processing duration while preserving the monotonic effect of thermal accumulation. Since the processing time depends on the thermal state, the evaluation of a candidate sequence requires integrating Eq (3.15) throughout the complete schedule. Consequently, the scheduling decisions and machine-state evolution are coupled: Longer processing periods increase temperature, and increased temperature further extends future processing times. The mathematical formulation above defines the feasible permutation structure.

4. Solution methodology

The methodology developed in this study addresses the PFSP with thermally driven processing times by coupling metaheuristic optimization with the physical state dynamics of the production system. The hybrid metaheuristic presented in this study integrates the well-known NEH constructive heuristic within a multi-start framework, enhanced by a BRA procedure. The initial solution is generated using the NEH heuristic, in which jobs are first ordered in decreasing total processing time, augmented by a penalty for high variance to prioritize more stable jobs in the initial sequence. The jobs are subsequently inserted into the position that minimizes the makespan. Although the search space corresponds to all possible job permutations, the NEH procedure explores only a restricted subset since at each iteration, the partial sequence is extended by inserting a single job into the existing order, thus limiting the number of candidate solutions. To enhance diversification, the BRA is subsequently applied: A geometric distribution is used to select jobs from the ordered list, enabling a broader exploration of the solution space. The only required parameter involved in the BRA is the p-value of the geometric distribution, which controls the intensity of the randomization during the construction of the solutions. In accordance with standard practices, we tested the parameter within

the range $[0.1, 0.3]$. The integration of these components allows for the exploration of the solution space while accounting for the non-linear dependencies between scheduling decisions and machine efficiency. In this context, the search process relies on the classical insertion operator underlying the NEH heuristic, complemented by the use of biased randomization mechanisms to effectively explore promising regions of the solution space. The core of the proposed approach lies in the evaluation of the makespan under time-varying conditions. To provide a robust assessment, the methodology distinguishes between three specific temporal metrics: the soft lower bound, the soft upper bound, and the dynamic makespan. The first one assumes constant processing times, while the soft upper bound is obtained by evaluating the static sequence under dynamic processing times, capturing the performance degradation induced by ignoring machine state effects during the construction phase. The last one is the primary objective of our research. The dynamism is computed by solving an ODE that characterizes the machine state during both active and idle intervals.

As depicted in the procedural architecture (Figure 2), the NEH heuristic is first applied under static processing times to generate an initial sequence, which serves as a reference solution. This sequence is then evaluated under dynamic conditions to obtain the corresponding dynamic makespan (D_{mspan}). This sequence, as well as the soft lower bound, are subsequently refined through the BRA, which introduces controlled stochasticity into the job selection process [35]. The search explores the neighborhood of the most promising solutions, effectively balancing the exploitation of the heuristic knowledge with the exploration of the permutation space to avoid local optima. As shown in Figure 2, soft upper bound values always come from applying dynamic times to the corresponding soft lower bound sequence.

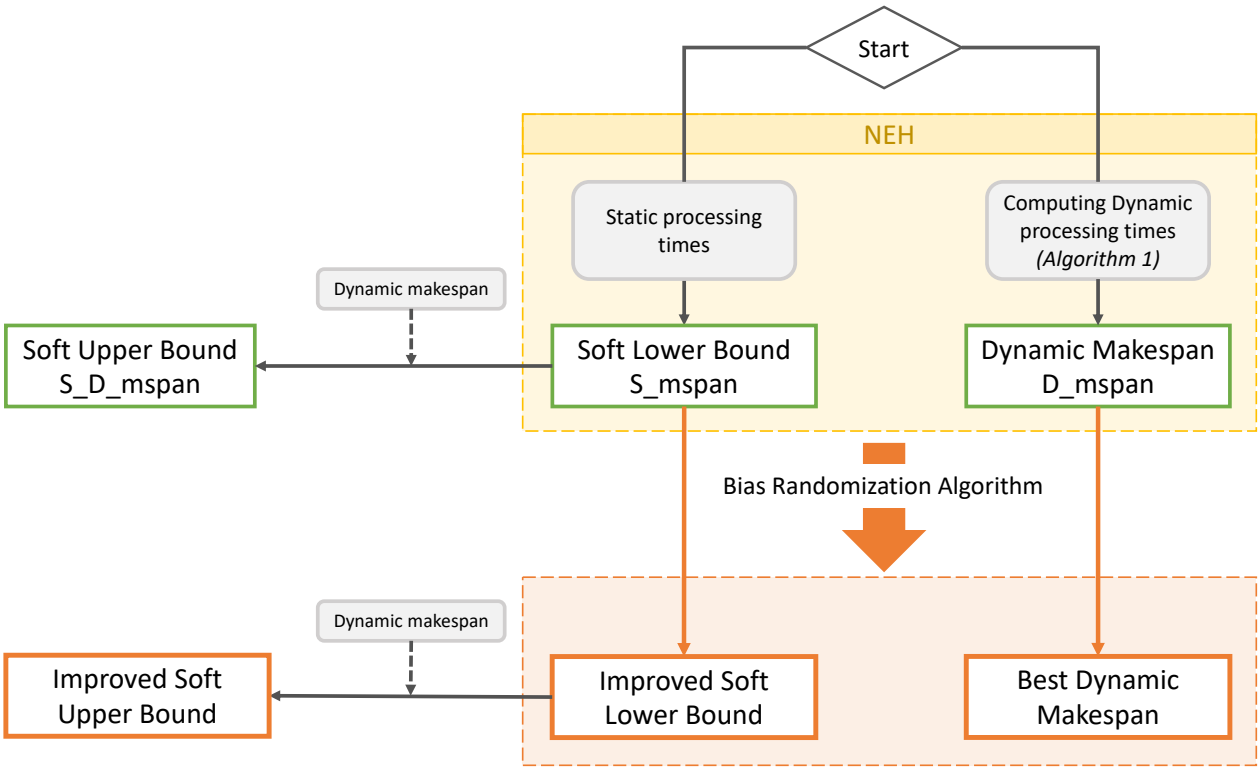

Figure 2. Scheme of the algorithm.

The physical interaction within the flow shop is modeled as a continuous feedback loop where the idle time between operations facilitates a cooling effect, thereby resetting or reducing the machine state variable. Conversely, active processing periods generate heat that increases the state variable, leading to a direct impact on the dynamic processing time of subsequent tasks (Algorithm 1). The iterative execution of this randomized framework progressively improves the dynamic makespan, leading to high-quality solutions that account for both scheduling and thermal constraints.

Algorithm 1 Computing dynamic processing times

Input: Sequence of jobs σ, List of machines Λ, p_{ij}^0, T_{env}
Output: Dynamic Makespan C_{max}

1: Initialize $M_{ready} \leftarrow 0_{1\times m}$, $J_{ready} \leftarrow 0_{1\times n}$
2: $T[i] \leftarrow T_{env} \quad \forall i \in \Lambda$
3: **for each** $j \in \sigma$ **do**
4: **for each** $i \in \Lambda$ **do**
5: $t_{start} \leftarrow \max(M_{ready}[i], J_{ready}[j])$
6: $t_{idle} \leftarrow t_{start} - M_{ready}[i]$
7: **if** $t_{idle} > 0$ **then**
8: $T[i] \leftarrow \text{SolveODE}(T[i], t_{start}, t_{idle})$ ▷ Machine Recovery (Cooling)
9: **end if**
10: $p_{ij}^t \leftarrow p_{i,j}^0 \times g(T[i], T_{env})$
11: $T[i] \leftarrow \text{SolveODE}(T[i], t_{start}, p_{ij}^t)$ ▷ Machine Working (Heating)
12: $M_{ready}[i] \leftarrow t_{start} + p_{ij}^t$
13: $J_{ready}[j] \leftarrow M_{ready}[i]$
14: **end for**
15: **end for**
16: $C_{max} \leftarrow \max(M_{ready})$
17: **return** C_{max}

To evaluate the efficiency of the generated schedules, a dynamic makespan calculation procedure is developed (Algorithm 1). Unlike traditional scheduling where processing times are fixed, our approach integrates the machine's physical state (T_i) into the temporal constraints. The procedure iterates through the job sequence, tracking two synchronized timelines: machine availability (M_{ready}) and job availability (J_{ready}). For each operation, the algorithm identifies the idle time. During this interval, the machine's state recovery (cooling) is simulated by solving the governing ODE with a null workload (line 8). Once the operation begins, the nominal processing time p_{ij}^t is adjusted by a state-dependent efficiency factor g (line 10), reflecting the degradation at the start time. Finally, the machine state is updated by solving the ODE under an active workload for the duration of the real processing time (line 11 in Algorithm 1). This integration ensures that the resulting makespan C_{max} accounts for both the technological constraints of the flow shop and the thermal/physical limitations of the production system.

5. Computational experiments

This section describes the experimental setup and the results obtained with the proposed ODE-augmented matheuristic described in Algorithm 1. The experiments are conducted on the well-known Taillard benchmark instances for the PFSP [10]. Each instance is defined by a pair (n, m), where n is the number of jobs and m is the number of machines, with sizes ranging from 5×20 to 20×500. For each configuration, 10 independent problems are generated using different random seeds. Processing times p^0_{ij} are integer values drawn from a uniform distribution over $[1, 99]$. For each instance, the best known solution (BKS) is available in the literature and is used as a reference for the static scenario.

To introduce state-dependent processing times, the thermal model defined in Eq (3.15) is used. For each machine i, the parameters are randomly generated within the ranges $\alpha_i \in [0.02, 0.08]$ and $\beta_i \in [2.0, 8.0]$. The ambient temperature is fixed to $T_{env} = 25°$. These parameters remain constant during the evaluation of each instance. The proposed methodology is implemented in Python 3.11 and executed on a workstation with an Intel Core i5-11400 processor and 32 GB of RAM, running Windows 11 Pro. The same computational environment is used for all experiments. The primary computational effort arises from the evaluation of candidate schedules during the biased-randomized search process. The dynamic extension introduces additional computational overload due to the simulation of machine-state evolution, as the thermal state of each machine must be continuously updated throughout schedule execution. In this approach, the average execution times for the Taillard instances with 20 jobs are 58, 169, and 501 seconds for 5, 10, and 20 machines, respectively. When the number of jobs increases to 50 and 100, the values are 622, 2351, and 5161 seconds, respectively, and 3305, 9996, and 16340 seconds, respectively. Consequently, the overall computational cost depends on both the number of candidate solutions explored and the numerical integration required to update the thermal states. Three makespan values are considered for each instance: *(i)* $C_{max}(S_0, P^0)$, the makespan of the solution generated by the BRA under static processing times (denoted as S_BRA), which acts as a reference lower bound within the proposed framework; *(ii)* $C_{max}(S_0, P^t)$, the dynamic evaluation of the same sequence (denoted as S_BRA_D), which provides a reference upper bound capturing the degradation caused by thermal effects; and *(iii)* $C_{max}(S^*, P^t)$, the dynamic makespan of the best solution found by the proposed method (denoted as D_BRA). In accordance with the validation framework introduced in Section 1, these values should satisfy

$$C_{max}(S_0, P^0) \leq C_{max}(S^*, P^t) \leq C_{max}(S_0, P^t). \tag{5.1}$$

To quantify the effect of machine dynamics, the thermal impact of a sequence S is measured as:

$$\Delta E(S) = \left(\frac{C_{max}(S, P^t) - C_{max}(S, P^0)}{C_{max}(S, P^t)} \right) \times 100, \tag{5.2}$$

which represents the relative increase in makespan when static processing times are replaced by state-dependent ones.

Tables 2–4 report the results for instances with $n = 20$, $n = 50$, and $n = 100$ jobs. For each problem, the tables include the BKS, the static solution obtained with the BRA (S_BRA), its dynamic evaluation (S_BRA_D), and the best dynamic solution found (D_BRA). The corresponding optimality gaps with respect to the BKS and the reference bounds are also reported. The results show a consistent pattern across all instance sizes. First, the gap between BKS and S_BRA remains small, indicating

that the constructive phase provides competitive solutions in the static setting. Second, the dynamic evaluation of these static sequences (S_BRA_D) leads to a substantial increase in makespan, reflecting the impact of thermal effects when they are ignored during optimization. Third, the proposed method (D_BRA) systematically improves upon this dynamic baseline, producing solutions that remain strictly below $C_{max}(S_{BRA}, P^t)$. The magnitude of the thermal impact ΔE confirms that the static assumption is inadequate in the presence of state-dependent dynamics. Across all instances, the efficiency loss is significant, with average values exceeding 50% and reaching higher levels for larger problem sizes. This effect becomes more pronounced as the number of jobs and machines increases, due to the accumulation of thermal load over longer schedules. In contrast, the proposed ODE-augmented matheuristic mitigates this degradation by adapting job sequences to the evolving machine states.

Table 2. Computational results for $n = 20$ and $m \in \{5, 10, 20\}$.

Name	Problem	BKS	S_BRA (LB)	GAP BKS vs. S_BRA	D_BRA	S_BRA_D (UB)	GAP D_BRA vs. LB	GAP D_BRA vs. UB	ΔE (%)
t_j20_m5	P0	1278	1278	0.0%	1982.61	2855.46	35.5%	-44.0%	55.2%
	P1	1359	1359	0.0%	1781.82	2206.67	23.7%	-23.8%	38.4%
	P2	1081	1098	1.6%	1512.78	2483.24	27.4%	-64.2%	55.8%
	P3	1293	1301	0.6%	1947.48	3477.28	33.2%	-78.6%	62.6%
	P4	1235	1244	0.7%	1879.03	3444.36	33.8%	-83.3%	63.9%
	P5	1195	1200	0.4%	1762.3	2320.66	31.9%	-31.7%	48.3%
	P6	1234	1239	0.4%	1696.06	2454.7	26.9%	-44.7%	49.5%
	P7	1206	1206	0.0%	2058.29	2402.51	41.4%	-16.7%	49.8%
	P8	1230	1240	0.8%	1818.74	2808.16	31.8%	-54.4%	55.8%
	P9	1108	1117	0.8%	1506.6	1935.27	25.9%	-28.5%	42.3%
t_j20_m10	P0	1582	1599	1.1%	2277.91	3622.71	29.8%	-59.0%	55.9%
	P1	1659	1673	0.8%	2389.85	3181.76	30.0%	-33.1%	47.4%
	P2	1496	1518	1.5%	2193.03	2838.25	30.8%	-29.4%	46.5%
	P3	1377	1394	1.2%	2563.48	2962.71	45.6%	-15.6%	52.9%
	P4	1419	1426	0.5%	1925.22	2683.18	25.9%	-39.4%	46.9%
	P5	1397	1413	1.1%	2030.19	2498.44	30.4%	-23.1%	43.4%
	P6	1484	1486	0.1%	2123.48	2915.35	30.0%	-37.3%	49.0%
	P7	1538	1554	1.0%	3594.71	4045.54	56.8%	-12.5%	61.6%
	P8	1593	1610	1.1%	2440.48	2934.57	34.0%	-20.2%	45.1%
	P9	1591	1608	1.1%	2402.85	2663.67	33.1%	-10.9%	39.6%
t_j20_m20	P0	2297	2323	1.1%	2961.22	4874.31	21.6%	-64.6%	52.3%
	P1	2099	2114	0.7%	2978.45	4524.6	29.0%	-51.9%	53.3%
	P2	2326	2356	1.3%	3234.19	3558.51	27.2%	-10.0%	33.8%
	P3	2223	2233	0.4%	2944.26	3876.24	24.2%	-31.7%	42.4%
	P4	2291	2321	1.3%	3127.07	3595.97	25.8%	-15.0%	35.5%
	P5	2226	2254	1.3%	2997.13	3604.8	24.8%	-20.3%	37.5%
	P6	2273	2304	1.4%	3091.15	4895.49	25.5%	-58.4%	52.9%
	P7	2200	2213	0.6%	2986.35	3962.84	25.9%	-32.7%	44.2%
	P8	2237	2250	0.6%	3158.15	3726.46	28.8%	-18.0%	39.6%
	P9	2178	2180	0.1%	3473.77	4595.92	37.2%	-32.3%	52.6%
	Average	1656.83	1670.37	0.8%	2427.96	3264.99	30.9%	-36.2%	48.5%

Table 3. Computational results for $n = 50$ and $m \in \{5, 10, 20\}$.

Name	Problem	BKS	S_BRA (LB)	GAP BKS vs. S_BRA	D_BRA	S_BRA_D (UB)	GAP D_BRA vs. LB	GAP D_BRA vs. UB	ΔE (%)
t_j50_m5	P0	2724	2724	0.0%	4517.53	5727.95	65.8%	-21.1%	52.4%
	P1	2834	2838	0.1%	8386.03	10205.68	195.5%	-17.8%	72.2%
	P2	2621	2621	0.0%	3822.85	4725.28	45.9%	-19.1%	44.5%
	P3	2751	2753	0.1%	4931.7	6396.25	79.1%	-22.9%	57.0%
	P4	2863	2863	0.0%	5156.29	7684.45	80.1%	-32.9%	62.7%
	P5	2829	2829	0.0%	4806.41	5409.22	69.9%	-11.1%	47.7%
	P6	2725	2732	0.3%	5580.81	8475.26	104.3%	-34.2%	67.8%
	P7	2683	2683	0.0%	8126.41	9050.54	202.9%	-10.2%	70.4%
	P8	2552	2558	0.2%	6458.24	6764.08	152.5%	-4.5%	62.2%
	P9	2782	2782	0.0%	4908.21	5307.89	76.4%	-7.5%	47.6%
t_j50_m10	P0	2991	3084	3.1%	5233.57	9330.8	69.7%	-43.9%	66.9%
	P1	2867	2960	3.2%	4991.78	6994.07	68.6%	-28.6%	57.7%
	P2	2839	2924	3.0%	4911.28	6467.74	68.0%	-24.1%	54.8%
	P3	3063	3107	1.4%	5289.8	7518.19	70.3%	-29.6%	58.7%
	P4	2976	3065	3.0%	5179.55	10008.26	69.0%	-48.2%	69.4%
	P5	3006	3084	2.6%	5078.14	6251.07	64.7%	-18.8%	50.7%
	P6	3093	3164	2.3%	5057.35	7815.67	59.8%	-35.3%	59.5%
	P7	3037	3093	1.8%	5738.98	7754.13	85.5%	-26.0%	60.1%
	P8	2897	2975	2.7%	5249.08	6695.4	76.4%	-21.6%	55.6%
	P9	3065	3158	3.0%	5262.08	9088.48	66.6%	-42.1%	65.3%
t_j50_m20	P0	3846	3985	3.6%	8675.4	9791.47	117.7%	-11.4%	59.3%
	P1	3699	3853	4.2%	6088.09	10372.78	58.0%	-41.3%	62.9%
	P2	3640	3806	4.6%	6164.82	9950.87	62.0%	-38.0%	61.8%
	P3	3719	3876	4.2%	6011.23	6931.19	55.1%	-13.3%	44.1%
	P4	3610	3739	3.6%	6373.44	8870.5	70.5%	-28.2%	57.8%
	P5	3679	3815	3.7%	6476.31	8493.16	69.8%	-23.7%	55.1%
	P6	3704	3858	4.2%	5709.32	11530.83	48.0%	-50.5%	66.5%
	P7	3691	3858	4.5%	6092.87	8325.46	57.9%	-26.8%	53.7%
	P8	3741	3865	3.3%	6031.9	9000.27	56.1%	-33.0%	57.1%
	P9	3755	3911	4.2%	6484.15	8736.62	65.8%	-25.8%	55.2%
	Average	3142.7	3218.8	2.2%	5759.8	7989.1	81.1%	-26.4%	58.5%

Table 4. Computational results for $n = 100$ and $m \in \{5, 10, 20\}$.

Name	Problem	BKS	S_BRA (LB)	GAP BKS vs. S_BRA	D_BRA	S_BRA_D (UB)	GAP D_BRA vs. LB	GAP D_BRA vs. UB	ΔE (%)
t_j100_m5	P0	5493	5493	0.0%	8827.06	12622.31	60.7%	-30.1%	56.5%
	P1	5268	5284	0.3%	8473.42	11496.02	60.4%	-26.3%	54.0%
	P2	5175	5196	0.4%	9592.4	11057.2	84.6%	-13.2%	53.0%
	P3	5014	5020	0.1%	10174.68	13029.95	102.7%	-21.9%	61.5%
	P4	5250	5252	0.0%	8069.6	18411.48	53.6%	-56.2%	71.5%
	P5	5135	5137	0.0%	7292.8	11413.57	42.0%	-36.1%	55.0%
	P6	5246	5250	0.1%	8885.42	19174.98	69.2%	-53.7%	72.6%
	P7	5094	5100	0.1%	14548.98	15978.95	185.3%	-8.9%	68.1%
	P8	5448	5452	0.1%	8887.1	17784.93	63.0%	-50.0%	69.3%
	P9	5322	5337	0.3%	10847.19	11588.24	103.2%	-6.4%	53.9%
t_j100_m10	P0	5770	5799	0.5%	16466.51	19471.85	184.0%	-15.4%	70.2%
	P1	5349	5386	0.7%	10019.22	14036.8	86.0%	-28.6%	61.6%
	P2	5676	5699	0.4%	11834.2	13132.03	107.7%	-9.9%	56.6%
	P3	5781	5891	1.9%	10312.56	16562.02	75.1%	-37.7%	64.4%
	P4	5467	5575	2.0%	10264.2	13499.23	84.1%	-24.0%	58.7%
	P5	5303	5339	0.7%	8442.19	12346.92	58.1%	-31.6%	56.8%
	P6	5595	5648	0.9%	11310.98	13190.96	100.3%	-14.3%	57.2%
	P7	5617	5695	1.4%	11295.06	14180.17	98.3%	-20.3%	59.8%
	P8	5871	5959	1.5%	10330.52	11461.38	73.4%	-9.9%	48.0%
	P9	5845	5881	0.6%	13091.11	22909.56	122.6%	-42.9%	74.3%
t_j100_m20	P0	6173	6470	4.8%	11407.97	16279.18	76.3%	-29.9%	60.3%
	P1	6183	6585	6.5%	11591.96	14295.07	76.0%	-18.9%	53.9%
	P2	6252	6423	2.7%	14937.69	17787.52	132.6%	-16.0%	63.9%
	P3	6254	6472	3.5%	10713.03	13562.19	65.5%	-21.0%	52.3%
	P4	6385	6555	2.7%	20969.76	22236.5	219.9%	-5.7%	70.5%
	P5	6331	6633	4.8%	11541.21	15288.63	74.0%	-24.5%	56.6%
	P6	6223	6515	4.7%	12095.13	18308.47	85.7%	-33.9%	64.4%
	P7	6372	6678	4.8%	11374.19	15917.42	70.3%	-28.5%	58.0%
	P8	6247	6534	4.6%	10820.26	14193.4	65.6%	-23.8%	54.0%
	P9	6404	6622	3.4%	11735.71	18787.4	77.2%	-37.5%	64.8%
	Average	5718.1	5829.3	1.8%	11205.1	15333.5	91.9%	-25.9%	60.7%

6. Analysis of results

The relationship between the operational schedule and the thermodynamic behavior of the system is illustrated through the synchronization of the Gantt chart and the thermal evolution profiles. We focus this discussion on our contribution, both when BRA is not applied in the execution (Figures 3 and 4) and when it is applied (Figures 5 and 6). As observed in Figures 3 and 4, and similarly in Figures 5 and 6, each block in the Gantt diagram corresponds to a period of active heat generation on a specific machine, where the duration of each task directly influences the slope of the temperature increase. The health state trajectories exhibit significant fluctuations and sustained peaks, particularly on machines with higher workload density or unfavorable dissipation parameters. These thermal spikes coincide with segments in the Gantt chart where idle times are insufficient to allow adequate cooling according to the relaxation term of the differential equation. Consequently, the accumulated thermal stress shown in Figures 4 and 6 explains the expansion of the individual processing times represented in Figures 3 and 5, respectively, as the state-dependent penalty function increases the effective duration of tasks when the machine operates.

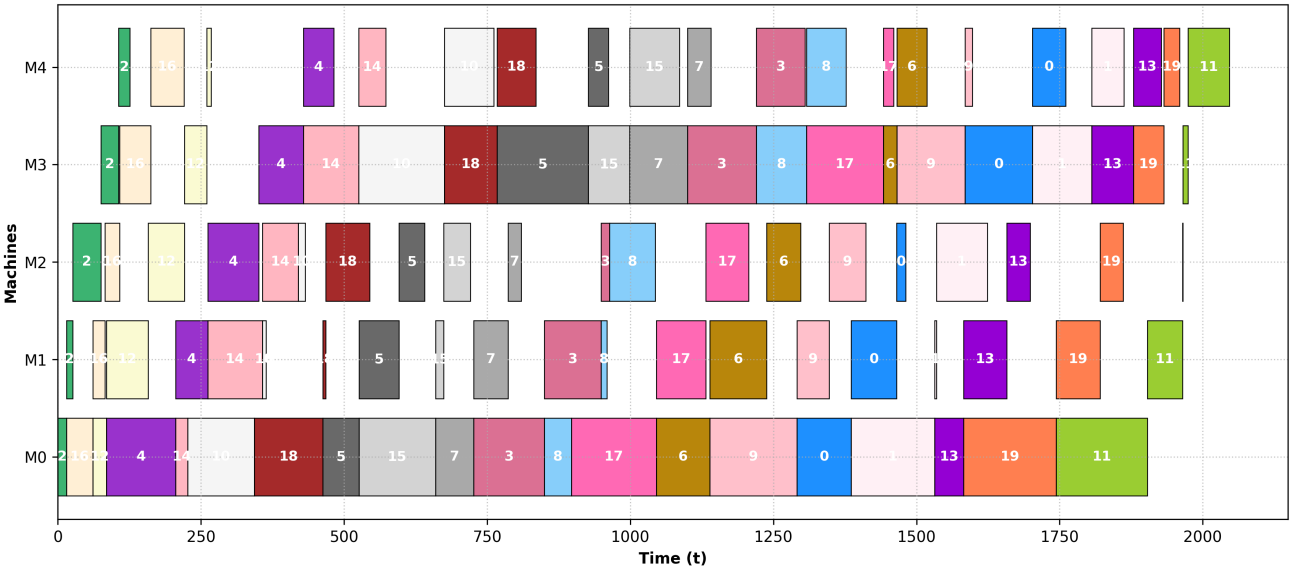

Figure 3. Gantt diagram for problem t_j20_m5_P0 with no_BRA execution. Makespan: 2046.49.

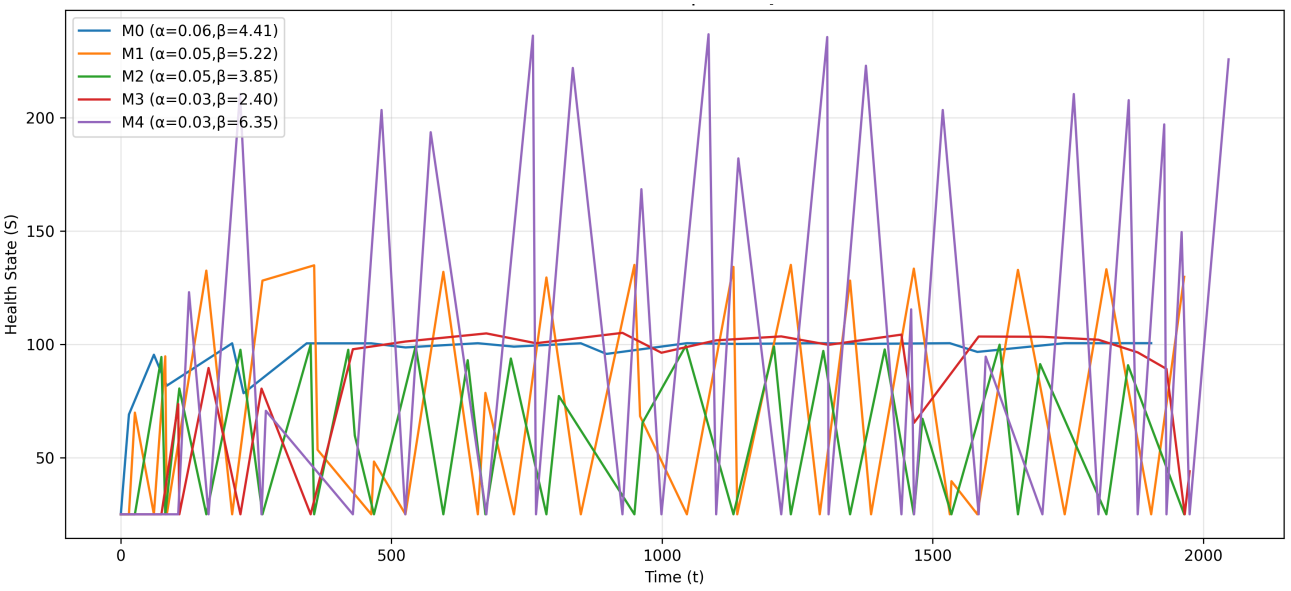

Figure 4. Health evolution diagram of the machines in the instance t_j20_m5_P0 with no_BRA execution.

Figures 3 and 5 illustrate the comparative performance of the scheduling sequences for the instance *t_j20_m5* under the P0-problem configuration. Figure 3 displays the baseline execution without the BRA mechanism, resulting in a total makespan of 2046.49 time units. In contrast, Figure 5 shows the impact of the BRA execution, which optimizes the workload distribution across the five machines and reduces the makespan to 1982.61 time units. This comparative analysis reveals that the integration of the proposed heuristic achieves a more compact schedule by effectively reordering jobs to minimize idle times. The visual distribution of tasks in the second scenario suggests a more balanced use of

machine resources, validating that the BRA approach enhances operational flow and improves overall temporal efficiency within the dynamic constraints of the flow shop problem. These results indicate that accounting for thermal profiles allows for a reduction in the total completion time while maintaining a more consistent processing density across the system.

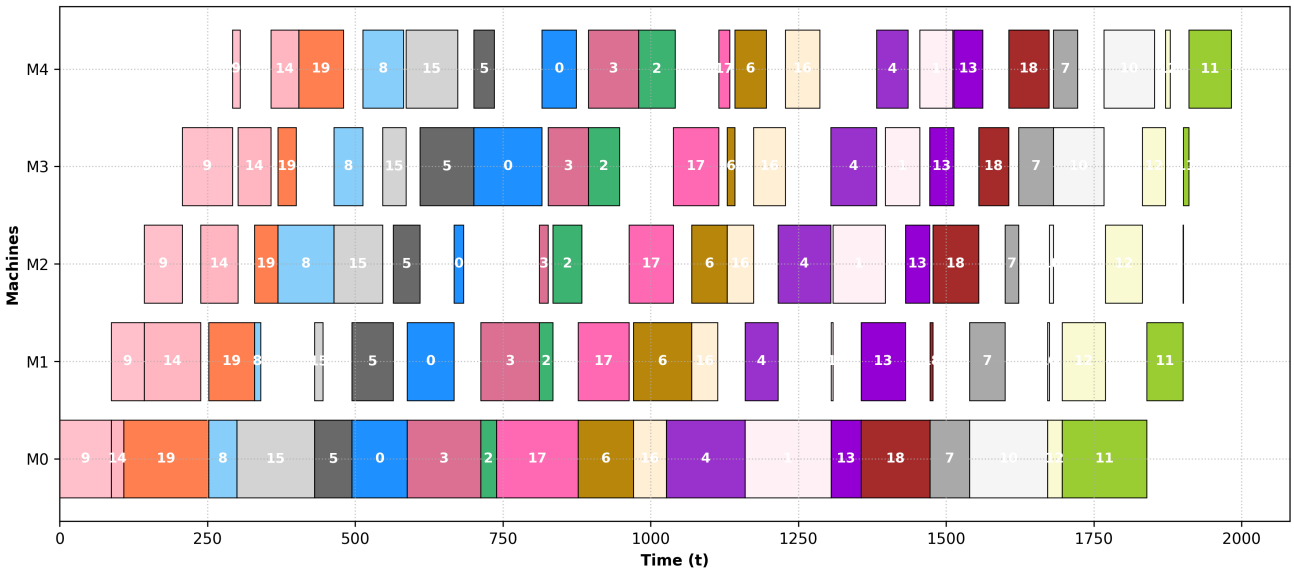

Figure 5. Gantt diagram for problem t_j20_m5_P0 with BRA execution. Makespan: 1982.61.

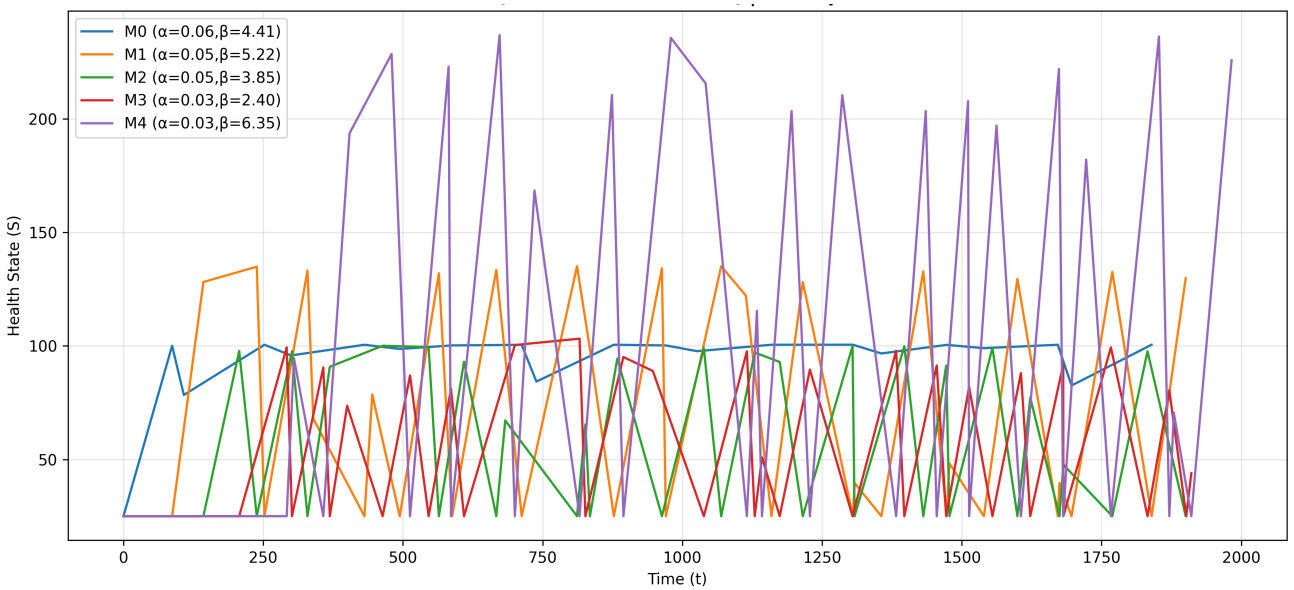

Figure 6. Health evolution diagram of the machines in the instance t_j20_m5_P0 with BRA execution.

The reliability of the BRA is evaluated by analyzing the distribution of makespan values across different operational environments. Figures 7–9 illustrate the performance dispersion for instances of varying scales under three distinct modeling approaches: static, dynamic and the evaluation of static

sequences under dynamic conditions (static-dynamic). In addition, for each approach, the dispersion of solutions is represented when the BRA is not applied and when it is applied. This statistical characterization is essential to confirm that the observed improvements are not incidental but result from a robust optimization strategy.

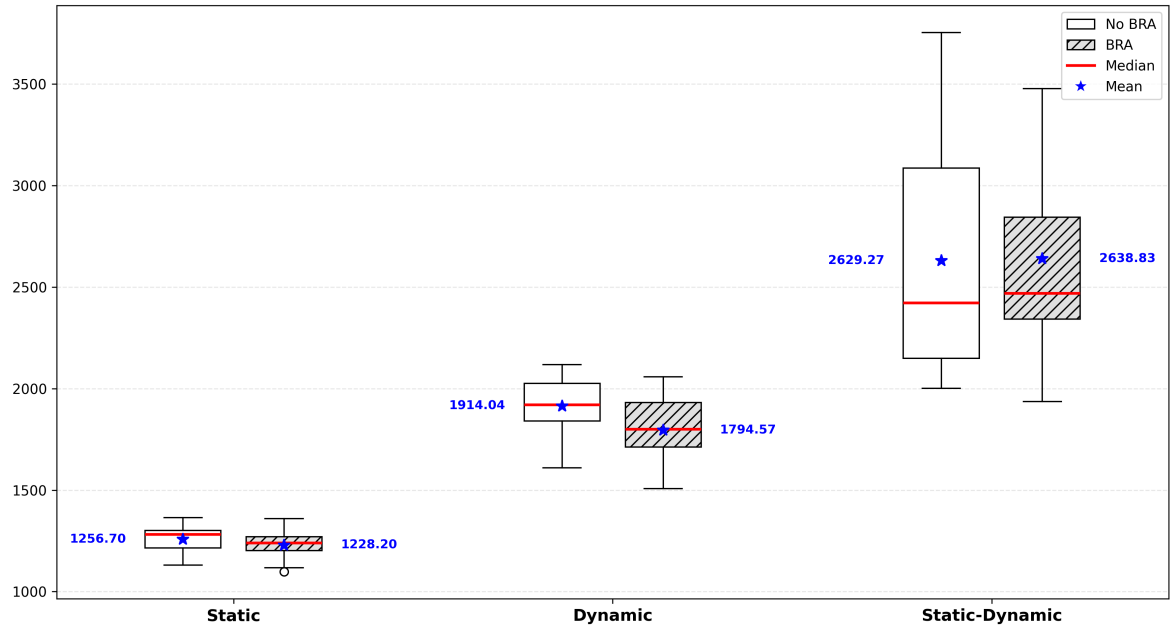

Figure 7. Boxplot comparison among the modeling approaches in the instance t_j20_m5.

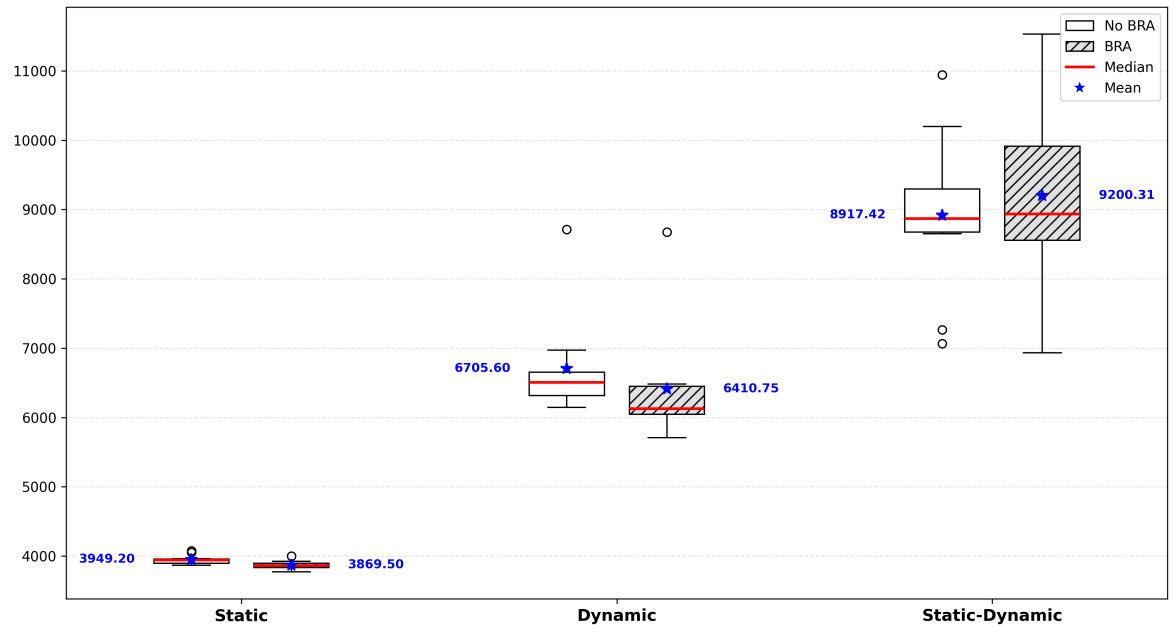

Figure 8. Boxplot comparison among the modeling approaches in the instance t_j50_m20.

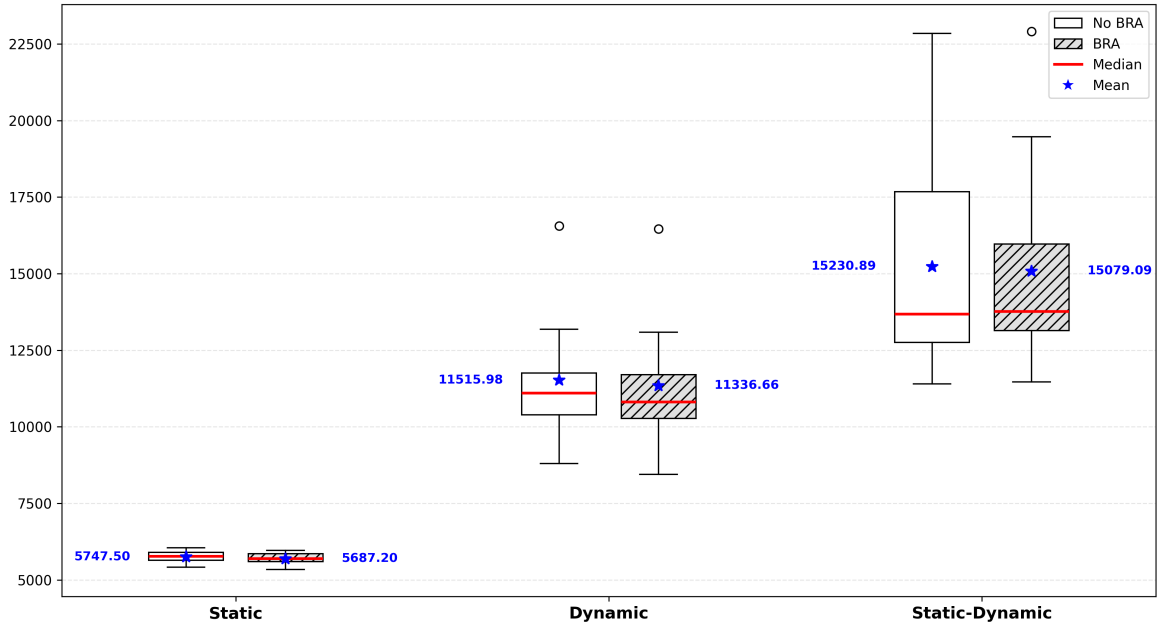

Figure 9. Boxplot comparison among the modeling approaches in the instance t_j100_m10.

In the static scenarios, the system exhibits minimal variability, as processing times remain constant and independent of the thermal state. However, the introduction of thermal dynamics in the model leads to a significant increase in both the average makespan and its dispersion. Across all tested instances, the BRA approach consistently shifts the distribution toward lower values compared to the standard NEH. For example, in the t_j50_m20 instance (Figure 8), the mean makespan decreases from 6705.60 to 6410.75 when BRA is applied on the dynamic scenario. Furthermore, the contraction of the interquartile range in the BRA distributions suggests that the randomized search effectively identifies job sequences that are less susceptible to heat-induced temporal dilation. The static-dynamic scenario provides a critical benchmark by applying dynamic makespan calculations to sequences originally optimized under static assumptions. This approach reveals a substantial performance gap, as the sequences that appear optimal in a static context suffer the most from thermal penalties when executed. As shown in Figure 9 for the t_j100_m10 instance, the static-dynamic distributions show the highest makespan values and the greatest dispersion, highlighting how traditional scheduling fails to account for the cumulative effect of heat. In contrast, the dynamic model with the BRA maintains a significantly more stable and efficient performance.

7. Conclusions

This work presented a dynamic extension of the PFSP in which processing times depend on the evolution of machine states. The proposed matheuristic method combines a NEH-based constructive heuristic with a biased randomization mechanism and an ODE model that describes the thermal behavior of each machine. This combination enables scheduling decisions to explicitly account for feedback between machine usage and processing efficiency. The experimental results on the Taillard benchmark instances show a consistent impact of thermal dynamics on schedule quality. When static sequences are evaluated under dynamic conditions, the makespan increases substantially across all

instance classes. On average, the gap between static and dynamic evaluations reaches about 48.5% for 20-job instances, 58.5% for 50-job instances, and 60.7% for 100-job instances. This confirms that classical static solutions are not robust under thermal effects. In contrast, the proposed ODE-augmented matheuristic approach consistently reduces this degradation, improving over the dynamically evaluated static baseline while remaining within the bounds defined by static and dynamic reference solutions.

Across all tested instances, the proposed matheuristic achieves stable improvements in the dynamic setting. For example, in 20-job instances, the average gap between the proposed solution and the dynamic upper bound is about -36.2%, while for 50-job and 100-job instances, this value is approximately -26.4% and -25.9%, respectively. At the same time, the deviation from the static lower bound remains positive, indicating feasible intermediate solutions within the expected theoretical range. The method also maintains consistent behavior across problem sizes, with average improvements in makespan that become more relevant as machine and job counts increase, particularly due to accumulated thermal effects. A further observation is the reduction in solution dispersion under the proposed strategy. While static solutions show low variability, the introduction of thermal dynamics significantly increases makespan variance, especially in larger instances. The proposed method mitigates this effect, producing more stable performance profiles and reducing sensitivity to thermal fluctuations. In addition, the thermal impact indicator confirms efficiency losses above 50% in many instances, reaching values above 70% in some large cases.

A key contribution of this work is the ODE-augmented matheuristic that couples discrete scheduling decisions with continuous machine-state evolution. Unlike classical approaches that treat processing times as fixed, the proposed framework embeds an ODE model directly into the evaluation of candidate solutions. This allows machine health states, driven by heat generation and dissipation, to influence processing times during the search process. The computational results confirm that this coupling leads to more robust schedules and reductions in dynamic makespan relative to static-based sequences.

Future work will focus on extending the model to other scheduling environments and on developing more detailed analytical tools for the underlying ODE-driven dynamics. In particular, the interaction between discrete scheduling decisions and continuous machine state evolution remains an open direction for further investigation.

Author contributions

N. Garrido: Software, formal analysis, writing—original draft preparation; S. Oltra-Crespo: Methodology, software, formal analysis, writing—review; L. Agud-Albesa: Software, formal analysis, writing—original draft preparation; D. López-Rodríguez: Methodology, writing—review, editing; A. A. Juan: Conceptualization, formal analysis, editing. All authors have read and approved the final version of the manuscript for publication.

Use of Generative-AI tools declaration

During the preparation of this manuscript, the authors used OpenAI ChatGPT (GPT-5.5 Pro) for language editing and Python support programming. The authors reviewed and edited the output and take full responsibility for the content of this publication.

Acknowledgments

This work has been partially supported by the UPV (PAID-06-25), the Spanish Ministry of Science, Innovation and Universities MICIU/AEI/10.13039/501100011033 (PID2022-138860NB-I00, AIA2025-163553-C44), and the Generalitat Valenciana (2024-CIAICO-117).

Conflict of interest

Prof. Ángel A. Juan is an editorial board member for AIMS Mathematics and was not involved in the editorial review or the decision to publish this article.

The authors declare that they have no conflicts of interest.

References

1. J. M. Framinan, J. N. D. Gupta, R. Leisten, A review and classification of heuristics for permutation flow-shop scheduling with makespan objective, *J. Oper. Res. Soc.*, **55** (2004), 1243–1255. https://doi.org/10.1057/palgrave.jors.2601784
2. R. Ruiz, C. Maroto, A comprehensive review and evaluation of permutation flowshop heuristics, *Eur. J. Oper. Res.*, **165** (2005), 479–494. https://doi.org/10.1016/j.ejor.2004.04.017
3. T. C. E. Cheng, Q. Ding, B. M. Lin, A concise survey of scheduling with time-dependent processing times, *Eur. J. Oper. Res.*, **152** (2004), 1–13. https://doi.org/10.1016/S0377-2217(02)00909-8
4. S. Chorghe, R. Kumar, M. S. Kulkarni, V. Pandhare, B. K. Lad, Smart scheduling for next generation manufacturing systems: a systematic literature review, *J. Intell. Manuf.*, **36** (2025), 4447–4476. https://doi.org/10.1007/s10845-024-02484-2
5. W. C. Lee, C. C. Wu, C. C. Wen, Y. H. Chung, A two-machine flowshop makespan scheduling problem with deteriorating jobs, *Comput. Ind. Eng.*, **54** (2008), 737–749. https://doi.org/10.1016/j.cie.2007.10.010
6. Z. W. Sun, D. Y. Lv, C. M. Wei, J. B. Wang, Flow shop scheduling with shortening jobs for makespan minimization, *Mathematics*, **13** (2025), 363. https://doi.org/10.3390/math13030363
7. P. A. Villarinho, J. Panadero, L. S. Pessoa, A. A. Juan, F. L. Cyrino Oliveira, A simheuristic algorithm for the stochastic permutation flow-shop problem with delivery dates and cumulative payoffs, *Int. Trans. Oper. Res.*, **28** (2021), 716–737. https://doi.org/10.1111/itor.12862
8. M. Nawaz, E. E. Enscore, I. Ham, A heuristic algorithm for the m-machine, n-job flow-shop sequencing problem, *Omega*, **11** (1983), 91–95. https://doi.org/10.1016/0305-0483(83)90088-9
9. J. Belloso, A. A. Juan, J. Faulin, An iterative biased-randomized heuristic for the fleet size and mix vehicle-routing problem with backhauls, *Int. Trans. Oper. Res.*, **26** (2019), 289–301. https://doi.org/10.1111/itor.12379
10. É. Taillard, Benchmarks for basic scheduling problems, *Eur. J. Oper. Res.*, **64** (1993), 278–285. https://doi.org/10.1016/0377-2217(93)90182-M
11. A. N. H. Zaied, M. M. Ismail, S. S. Mohamed, Permutation flow shop scheduling problem with makespan criterion: Literature review, *J. Theor. Appl. Inf. Technol.*, **99** (2021), 830–848.

12. L. H. Sun, K. Cui, J. H. Chen, J. Wang, X. C. He, Research on permutation flow shop scheduling problems with general position-dependent learning effects, *Ann. Oper. Res.*, **211** (2013), 473–480. https://doi.org/10.1007/s10479-013-1481-6

13. Y. Y. Lu, Research on no-idle permutation flowshop scheduling with time-dependent learning effect and deteriorating jobs, *Appl. Math. Model.*, **40** (2016), 3447–3450. https://doi.org/10.1016/j.apm.2015.09.081

14. X. Xin, Q. Jiang, C. Li, S. Li, K. Chen, Permutation flow shop energy-efficient scheduling with a position-based learning effect, *Int. J. Prod. Res.*, **61** (2023), 382–409. https://doi.org/10.1080/00207543.2021.2008041

15. A. A. Juan, B. B. Barrios, E. Vallada, D. Riera, J. Jorba, A simheuristic algorithm for solving the permutation flow shop problem with stochastic processing times, *Simul. Model. Pract. Theory*, **46** (2014), 101–117. https://doi.org/10.1016/j.simpat.2014.02.005

16. E. M. Gonzalez-Neira, J. R. Montoya-Torres, J. F. Jimenez, A multicriteria simheuristic approach for solving a stochastic permutation flow shop scheduling problem, *Algorithms*, **14** (2021), 210. https://doi.org/10.3390/a14070210

17. J. Castaneda, X. A. Martin, M. Ammouriova, J. Panadero, A. A. Juan, A fuzzy simheuristic for the permutation flow shop problem under stochastic and fuzzy uncertainty, *Mathematics*, **10** (2022), 1760. https://doi.org/10.3390/math10101760

18. E. M. González-Neira, A. M. Urrego-Torres, A. M. Cruz-Riveros, C. Henao-García, J. R. Montoya-Torres, L. P. Molina-Sánchez, et al., Robust solutions in multi-objective stochastic permutation flow shop problem, *Comput. Ind. Eng.*, **137** (2019), 106026. https://doi.org/10.1016/j.cie.2019.106026

19. K. Katragjini, E. Vallada, R. Ruiz, Flow shop rescheduling under different types of disruption, *Int. J. Prod. Res.*, **51** (2013), 780–797. https://doi.org/10.1080/00207543.2012.666856

20. P. Valledor, A. Gomez, P. Priore, J. Puente, Solving multi-objective rescheduling problems in dynamic permutation flow shop environments with disruptions, *Int. J. Prod. Res.*, **56** (2018), 6363–6377. https://doi.org/10.1080/00207543.2018.1468095

21. G. Mosheiov, Scheduling jobs under simple linear deterioration, *Comput. Oper. Res.*, **21** (1994), 653–659. https://doi.org/10.1016/0305-0548(94)90080-9

22. S. Alaswad, Y. Xiang, A review on condition-based maintenance optimization models for stochastically deteriorating system, *Reliab. Eng. Syst. Saf.*, **157** (2017), 54–63. https://doi.org/10.1016/j.ress.2016.08.009

23. T. Yang, Y. Kuo, I. Chang, Tabu-search simulation optimization approach for flow-shop scheduling with multiple processors—a case study, *Int. J. Prod. Res.*, **42** (2004), 4015–4030. https://doi.org/10.1080/00207540410001699381

24. R. Wallrath, M. B. Franke, Integration of MILP and discrete-event simulation for flow shop scheduling using Benders cuts, *Comput. Chem. Eng.*, **189** (2024), 108809. https://doi.org/10.1016/j.compchemeng.2024.108809

25. E. Azab, M. Nafea, L. A. Shihata, M. Mashaly, A machine-learning-assisted simulation approach for incorporating predictive maintenance in dynamic flow-shop scheduling, *Appl. Sci.*, **11** (2021), 11725. https://doi.org/10.3390/app112411725

26. Y. Fu, Z. Li, K. Gao, A. M. Fathollahi-Fard, Z. Zhang, Integrated scheduling of distributed manufacturing with assembly and distribution: State of the art, challenges, and future directions, *Int. J. Prod. Res.*, **64** (2026), 4669–4710. https://doi.org/10.1080/00207543.2025.2602898

27. A. M. Fathollahi-Fard, L. Woodward, O. Akhrif, A scenario-based robust optimization model for the sustainable distributed permutation flow-shop scheduling problem, *Ann. Oper. Res.*, 2024. https://doi.org/10.1007/s10479-024-05940-7

28. D. M. Lei, B. Su, M. Li, Cooperated teaching-learning-based optimisation for distributed two-stage assembly flow shop scheduling, *Int. J. Prod. Res.*, **59** (2021), 7232–7245. https://doi.org/10.1080/00207543.2020.1836422

29. M. Torkashvand, F. Ahmadizar, H. Farughi, Distributed production assembly scheduling with hybrid flowshop in assembly stage, *Int. J. Eng.*, **35** (2022), 1037–1055. https://doi.org/10.5829/IJE.2022.35.05B.19

30. G. H. Zhang, K. Y. Xing, G. Y. Zhang, Z. X. He, Memetic algorithm with meta-lamarckian learning and simplex search for distributed flexible assembly permutation flowshop scheduling problem, *IEEE Access*, **8** (2020), 96115–96128. https://doi.org/10.1109/ACCESS.2020.2996305

31. Y. D. Zuo, P. Wang, Z. Fan, M. Li, X. H. Guo, S. J. Gao, Minimizing fuzzy makespan in a distributed assembly flow shop by using an efficient artificial bee colony algorithm, *J. Intell. Fuzzy Syst.*, **45** (2023), 7025–7046. https://doi.org/10.3233/JIFS-230592

32. L. Cheng, L. Wang, J. C. Cai, A regional biogeography-based optimization algorithm for the distributed assembly permutation flow-shop scheduling problem with fuzzy processing time, *J. Intell. Fuzzy Syst.*, **46** (2024), 3827–3841. https://doi.org/10.3233/JIFS-235854

33. X. L. Jing, Q. K. Pan, L. Gao, Local search-based metaheuristics for the robust distributed permutation flowshop problem, *Appl. Soft Comput.*, **105** (2021), 107247. https://doi.org/10.1016/j.asoc.2021.107247

34. K. R. Baker, *Introduction to sequencing and scheduling*, Wiley, 1974.

35. C. L. Quintero-Araujo, J. P. Caballero-Villalobos, A. A. Juan, J. R. Montoya-Torres, A biased-randomized metaheuristic for the capacitated location routing problem, *Int. Trans. Oper. Res.*, **24** (2017), 1079–1098. https://doi.org/10.1111/itor.12322

AIMS Press