Journal of Artificial Intelligence and Autonomous Intelligence Received 26-05-2025; Accepted 26-07-2025; Published 02-08-2025

Tuning Metaheuristics with Tree-Structured Parzen Estimator: A Case Study on Scheduling

Angelo Foggetti angelo.foggetti@studenti.unisalento.it
Department of Innovation Engineering,
University of Salento,
Lecce, Italy.

Francesco Nucci francesco.nucci@unisalento.it
Department of Innovation Engineering,
University of Salento,
Lecce, Italy.

Gabriele Papadia gabriele.papadia@unisalento.it
Department of Innovation Engineering,
University of Salento,
Lecce, Italy.

Corresponding Author: Francesco Nucci

Abstract

Optimizing production schedules with sequence-dependent setup times (SDSTs) represents a critical challenge in manufacturing operations. This study proposes an intelligent scheduling framework that combines a Genetic Algorithm (GA) with automated parameter tuning using the Tree-Structured Parzen Estimator (TPE). The TPE learns from previous optimization runs to systematically adjust GA parameters, eliminating manual trial-and-error approaches. We evaluate the method on parallel machine scheduling problems with SDSTs, comparing performance against exact algorithms under equivalent computational time constraints. Experimental results demonstrate significant total completion time reductions of up to 32.12% in best cases and 14.87% on average compared to exact methods. When TPE is applied to Simulated Annealing instead of GA, performance consistently degrades by 1.10% to 13.23%, confirming the superiority of the GA-based approach. The proposed framework offers a scalable, adaptive solution for manufacturing scheduling optimization with direct implications for production efficiency and cost reduction.

Keywords: Metaheuristics, Tree-Structured parzen estimator, Scheduling optimization.

1. INTRODUCTION

Modern manufacturing environments face unprecedented complexity in production scheduling, driven by Industry 4.0 digitalization and increasingly dynamic operational requirements [1]. Traditional

Citation: Angelo Foggetti, et al. Tuning Metaheuristics with Tree-Structured Parzen Estimator: A Case Study on Scheduling. Journal of Artificial Intelligence and Autonomous Intelligence. Research. 2025;2(2):19.
Pages:293-321

scheduling methods often prove inadequate for addressing the highly constrained nature of real-world production systems, where factors such as sequence-dependent setup times (SDSTs) significantly impact operational efficiency.

The theoretical foundation of scheduling problems reveals their inherent computational complexity. Most scheduling problems are NP-hard [2], making exact algorithms computationally impractical for large-scale applications despite their optimality guarantees. While metaheuristics offer efficient alternatives for finding high-quality solutions in complex scenarios [3], their effectiveness heavily depends on parameter configuration—a process traditionally requiring extensive manual tuning and domain expertise [4].

Contemporary research emphasizes the need for adaptive, flexible scheduling approaches that can accommodate real-world constraints and variability. The integration of human-centered considerations [5], dynamic demand patterns [6], and ergonomic factors [7], further complicates scheduling optimization, highlighting the limitations of simplified mathematical models that fail to scale effectively in practice.

This study addresses these challenges by proposing an intelligent scheduling framework that combines evolutionary computation with machine learning-based parameter optimization. Specifically, we develop a Genetic Algorithm (GA) tailored for parallel machine scheduling with SDSTs, enhanced by automatic parameter tuning using the Tree-Structured Parzen Estimator (TPE). The TPE employs Bayesian optimization principles to learn from previous optimization runs, systematically adjusting GA parameters to improve performance across varying problem instances.

The research objectives are threefold: (1) formulate a realistic parallel machine scheduling problem incorporating SDSTs while maintaining computational tractability; (2) design and implement a GA-based metaheuristic specifically tailored to the problem structure; and (3) enhance solution quality through intelligent, automated parameter tuning. Our problem formulation considers identical parallel machines processing independent jobs with type-dependent processing and setup requirements, preserving essential industrial characteristics while maintaining strong NP-hardness [8].

Experimental evaluation demonstrates that the proposed method consistently outperforms exact algorithms when both approaches operate under equivalent computational time constraints, achieving makespan (total completion time) reductions up to 32.12% in optimal cases. The framework provides a scalable, interpretable decision-support tool applicable across industries where scheduling efficiency directly impacts throughput, cost control, and delivery performance. By bridging intelligent optimization with practical scheduling requirements, this work advances the development of adaptive, AI-supported scheduling systems in operations research.

2. BACKGROUND

2.1 Foundations and Advances in Metaheuristic Scheduling

Scheduling problems form a core class of combinatorial optimization challenges with broad applications in manufacturing and service systems [9]. The Job Shop Scheduling Problem (JSSP) with Sequence-Dependent Setup Times (SDSTs) is particularly challenging due to its NP-hard nature

[8]. Three main solution paradigms are employed: exact methods offering guaranteed optimality but poor scalability [3]; heuristics providing computational efficiency but lacking quality guarantees [10, 11]; and metaheuristics bridging this gap through stochastic exploration mechanisms [3].

Metaheuristics can be classified by inspiration source (nature-inspired vs. non-nature), search structure (single-solution vs. population-based), behavior (deterministic vs. stochastic), and foundational principles (evolutionary, swarm intelligence, physics-based, human-behavior) [4].

Recent research increasingly addresses Distributed Flow Shop Scheduling Problems (DFSSPs) with SDSTs, with notable contributions including Huang et al.'s [12], restarted Iterative Greedy algorithm, Meng et al.'s [13], MILP-guided Artificial Bee Colony, and Li et al.'s [14], destruction-construction IG for batch scheduling [8]. These studies illustrate a trend toward hybrid, problem-specific metaheuristics balancing theoretical rigor and practical applicability [15, 16].

Genetic Algorithms (GAs) have demonstrated strong performance in scheduling contexts [17]. Lee et al. [18], showed their effectiveness across job-shop, flow shop, and distributed systems, while Cheung et al. [19], and Vallada and Ruiz [20], enhanced them for SDST-specific problems. However, GA performance depends heavily on parameter tuning, which is often manual and suboptimal.

Tree-structured Parzen Estimator (TPE) addresses this limitation through Bayesian optimization for automatic hyperparameter tuning [21]. TPE partitions historical evaluations into high- and low-performing groups, models their distributions, and samples new configurations from promising regions. Its effectiveness extends beyond scheduling, achieving 99.86% accuracy in cancer classification [22], outperforming other methods in satellite imagery [23], and reaching $R^2 > 91\%$ in N_2O mitigation prediction [24].

2.2 Recent Hybrid Metaheuristics for SDST Scheduling

Recent advances confirm that model-based techniques like TPE can significantly enhance metaheuristic schedulers. A clear trend in 2022-2025 publications is the development of hybrid metaheuristics balancing global exploration with local exploitation.

Sayah et al. [25], introduced improved bio-inspired methods combining global metaheuristics with advanced local search: ABC with Path Relinking (ABC-PR), Migrating Birds Optimization with Variable Neighborhood Search (MBO-VNS), and GA with Individual Improvement Scheme (GA-IIS). Xu et al. [26], proposed HGQPSO-AGA, combining Gaussian Quantum Particle Swarm Optimization and Adaptive Genetic Algorithm for large-scale Flexible Job Shop Scheduling Problems.

Learning-based adaptivity gains traction through Q-learning for operator selection. Liu et al. [27], and Zhao et al. [28], embed Q-learning in ABC frameworks to dynamically select local search strategies, improving convergence in distributed and no-wait flow shop problems with SDSTs.

Mathematical programming continues complementing metaheuristics. Zhang et al. [29], proposed Matheuristic Self-learning Iterated Greedy (MSIG) for distributed heterogeneous assembly flow shops, while Xiong et al. [30], designed enhanced Logic-Based Benders Decomposition (UL-LBBD-SSR) achieving near-optimal solutions on large-scale instances. Deenen et al. [31], demonstrate Constraint Programming can outperform MIP in semiconductor scheduling.

Table 1: Comparative Overview of Recent Scheduling Methods

Ref. Year	Method / Algorithm	Model Type	Learning / Adaptivity	SDST Handling	Scalability	Validation Type
[25] 2025	ABC-PR, MBO-VNS, GA-IIS (hybrid)	Meta-heuristic	Local Search (PR, VNS, IIS)	Yes	Not specified, but effective on various instances	Simulation
[26] 2025	HGQPSO-AGA	Hybrid	Adaptive (crossover / mutation), Chaotic encoding, Multi-subpopulation	Not specified	Effective for large-scale (benchmarks & industrial case)	Benchmark, Case Study
[27] 2025	QABC (ABC with Q-learning)	Meta-heuristic	Q-learning for local search selection	Yes	Effective on 72 instances	Statistical, Experimental
[28] 2023	QABC (ABC with Q-learning)	Meta-heuristic	Q-learning for neighborhood selection	Yes	High quality in reasonable time	Experimental
[29] 2025	MSIG	Hybrid (Matheuristic)	Self-learning, knowledge-based operators	Not specified	Superior to 9 heuristics and 6 meta-heuristics	Comparative
[30] 2025	UL-LBBD-SSR	Decomposition (CP, MILP)	Not applicable	Not specified	Optimal for small instances, near-optimal for large-scale	Experimental, Comparative
[31] 2023	CP	CP	Not applicable	Yes	Outperforms MILP, high quality for real-world instances	Real-world Case Study
[32] 2023	Improved Memetic Algorithm	Meta-heuristic	Neighborhood search strategies	Yes	Not specified	Numerical Experiments
[33] 2025	MILP (and variants)	MILP	Not applicable	Yes	Scalability challenges with dense eligibility matrices	Experimental
[34] 2022	Learning-based Grey Wolf Optimizer	Meta-heuristic	Reinforcement Learning, Optimal Computing Budget Allocation	Yes	Effective and efficient in stochastic environments	Benchmark, Random, Theoretical

Multi-objective optimization and domain-specific constraints remain central. Liu et al. [27], and Wu et al. [32], use enhanced metaheuristics to minimize makespan and energy consumption, while Otamendi et al. [33], generalize Unrelated Parallel Machine problems with MILP. Uncertainty-aware scheduling emerges as Lin et al. [34], address stochastic FJSPs using learning-based Grey Wolf Optimizer.

TABLE 1 provides a comparative overview evaluating each method by model type, learning integration, SDST support, scalability, and industrial validation.

2.3 Deterministic AI vs. Learning-Based Metaheuristics

In SDST scheduling optimization, two paradigms compete: Deterministic Artificial Intelligence (DAI) and learning-enhanced metaheuristics. DAI, including exact algorithms and mathematical programming, performs well when problems exhibit strong structural properties. As demonstrated

Table 2: Comparison between DAI and the Proposed TPE + GA Method

Feature	DAI	TPE-GA
Approach	Exact methods; explicit mathematical formulations; seeks optimality.	Meta-heuristic (GA) enhanced by Machine Learning (TPE); automatic parameter optimization; balances exploration/exploitation.
Strengths	• Optimality/near-optimality: Optimal or near-optimal solutions for well-defined problems. • Predictability: Behavior governed by mathematical models.	• Efficiency/Scalability: High-quality solutions with low computational effort on large-scale problems. • Adaptability: Flexible to changing requirements and complex problems. • Auto-tuning: Automatic parameter optimization via TPE.
Limitations	• Limited scalability: Exponential computation time on large instances; high memory usage. • Specificity: Hard to adapt to small changes in the problem; requires expertise.	No guaranteed optimality: Produces high-quality solutions but may not find the certified optimum.
Applicability	Problems with clear mathematical structure and manageable instance sizes.	Complex, large, and dynamic NP-hard problems where efficiency and flexibility are crucial.

by [30], and [31], CP and MILP models yield optimal or near-optimal results under strict modeling assumptions [3, 10].

DAI's effectiveness extends to control-theoretic applications where precise mathematical models are essential. Zhai et al. [35], showed feed-forward control outperformed physics-informed neural networks, Xu et al. [36], demonstrated Pontryagin-based planners' superiority in DC motor control, and Wang et al. [37], proved continuous-time DAI retained accuracy under coarse discretization in underwater vehicle control.

However, DAI's precision often sacrifices flexibility and scalability. Exact algorithms face exponential time complexity and high memory demands, while problem-specific tailoring limits general applicability [3, 10].

The TPE-guided metaheuristic proposed here emphasizes adaptive learning over exhaustive modeling. Combining Genetic Algorithms with Tree-Structured Parzen Estimator tuning, our approach dynamically adapts to complex scheduling challenges. While lacking optimality guarantees, metaheuristics offer robustness, generality, and scalability with minimal structural assumptions [38].

TABLE 2 compares deterministic methods and the proposed adaptive metaheuristic across key dimensions.

2.4 Open Challenges and Future Directions

While TPE-enhanced GAs show promise, several challenges remain. Scaling to dynamic environments with thousands of jobs and machines, incorporating uncertainty through robust models, and

extending to multi-objective optimization are key research directions. Further improvements could integrate domain heuristics, develop interpretable models, and establish standardized benchmarks for fair comparisons.

Methods like TPE offer a principled approach to automate parameter tuning, improving metaheuristic performance without costly manual effort, making them particularly suitable for complex SDST scheduling where exact methods falter due to computational limitations.

3. MATERIALS AND METHODS

3.1 Research Problem

This study addresses a production scheduling problem where a set of jobs must be executed on identical parallel machines within a defined planning horizon. Each job belongs to a specific job type, which determines both its processing time and the setup time required when scheduled after a job of a different type on the same machine.

A sequence-dependent setup time matrix defines the duration of these transitions between job types. Setup times are incurred only when two consecutive jobs of different types are assigned to the same machine, significantly influencing both machine utilization and overall schedule feasibility.

The objective is to determine the job assignments and sequencing that minimize the makespan—that is, the completion time of the final job—taking into account both processing and setup times.

3.2 Problem Formulation and Mathematical Model

The studied problem is a parallel machine scheduling problem denoted as $Pm|s_{\tau(j)\tau(k)}|C_{\max}$, involving m machines and n jobs. Each job comprises a single operation with sequence-dependent setup times $s^i_{\tau(j)\tau(k)}$ required between consecutive jobs of different types on the same machine. The objective is to minimize the makespan $C_{\max}$. Key notation is reported in TABLE 3. The complete mathematical formulation is provided in Appendix A

3.2.1 Solution method

To tackle the $Pm|s_{\tau(j)\tau(k)}|C_{\max}$ scheduling problem, a modular, three-layered optimization architecture was developed, inspired by the hierarchical parameter tuning paradigm and depicted in FIGURE 1. The entire framework was implemented in Python, utilizing its scientific ecosystem to ensure flexibility, scalability, and seamless integration of automated hyperparameter tuning.

At the bottom layer (Application Layer) lies the core scheduling problem. This component encapsulates the definition of jobs, machines, processing and setup times, and the objective of minimizing the makespan. A problem-specific fitness function evaluates each candidate solution by computing the schedule's makespan, accounting for both processing and sequence-dependent setup times.

Table 3: Mathematical Notation: Parameters and Decision Variables

Notation	Description	Type
$\mathcal{N}$	Job set $\{1, 2, \ldots, n\}$	Parameter
$\mathcal{M}$	Machine set $\{1, 2, \ldots, m\}$	Parameter
T	Set of job types	Parameter
$\tau(j) \in T$	Type of job j	Parameter
$s^i_{\tau(j)\tau(k)}$	Setup time on machine M_i for type transition $\tau(j) \to \tau(k)$	Parameter
p_{ij}	Processing time of job J_j on machine M_i	Parameter
t_{ij}	Starting time of job J_j on machine M_i	Decision Variable
C_j	Completion time of job J_j, equal to $t_{ij} + p_{ij}$	Decision Variable
$C_{\max}$	Makespan (maximum completion time)	Decision Variable
x^i_{jk}	Job Order: 1 if job J_k follows job J_j on machine M_i, 0 otherwise	Decision Variable

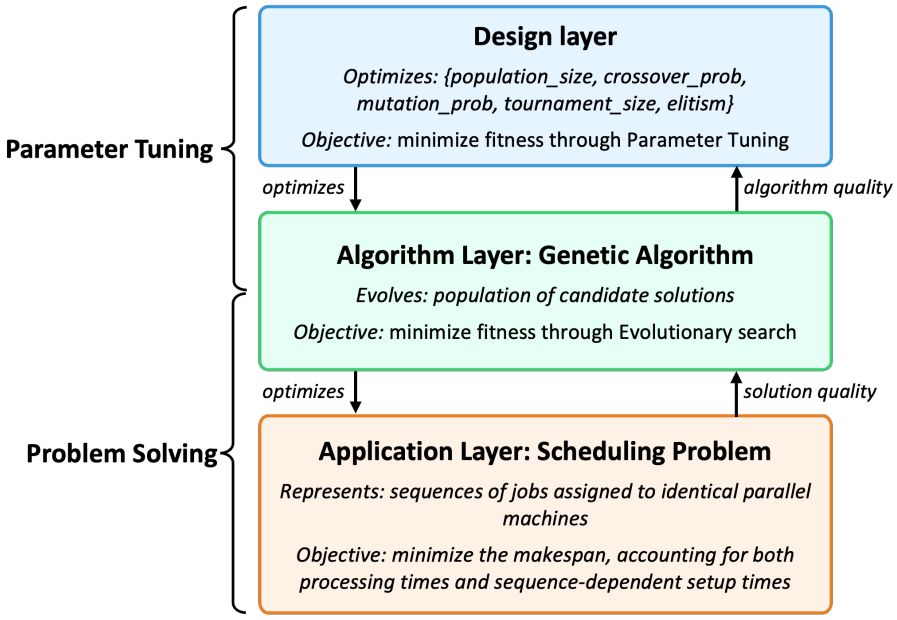

Figure 1: Three-Layer TPE-GA Architecture

The Design Layer, positioned at the top of the optimization architecture, is responsible for tuning the control parameters of the GA applied to the $Pm|s_{\tau(j)\tau(k)}|C_{\max}$ scheduling problem. This component is implemented using Optuna, a state-of-the-art framework for automated hyperparameter optimization. Within this framework, the TPE acts as a Bayesian sampler, iteratively suggesting GA configurations based on the performance of previously evaluated trials.

Design Layer (TPE Optimization): Implements Tree-structured Parzen Estimator via Optuna for automated hyperparameter tuning. TPE conducts 40 trials: 15 random exploration trials followed by 25 model-guided trials using kernel density estimation. Key enhancements include:

- Dynamic bound adjustment: Parameter ranges narrow around promising values during trials 16-40

- Convergence monitoring: Adaptation based on coefficient of variation and improvement rates
- Stagnation recovery: Range expansion when performance plateaus

The set of parameters subject to optimization includes those that govern the core dynamics of the Genetic Algorithm, namely: population size, crossover probability, mutation probability, number of elite individuals, and tournament size.

To enhance adaptability and responsiveness, the standard TPE sampler was extended with dynamic mechanisms designed to improve the efficiency of the search process, including convergence-aware search, dimension-wise adaptation, dynamic exploration-exploitation balancing, and stagnation recovery mechanisms.

Algorithm Layer (GA-TPR): runs a genetic algorithm with time-constrained parallel runs using configurations from TPE. Solutions are encoded as job sequences per machine - for example $[[3, 1], [0, 2]]$ means Machine 0 processes jobs 3 then 1, and Machine 1 processes jobs 0 then 2.

The Algorithm Layer serves as the computational core of the architecture, where a GA is executed to solve the $Pm|s_{\tau(j)\tau(k)}|C_{\max}$ scheduling problem. For each trial, the GA receives a configuration of parameters proposed by the TPE within the Design Layer.

This encoding representation captures both job-to-machine assignment and intra-machine sequencing, where the position of a job within each sublist determines its execution order. This approach is well suited for modeling scheduling problems with sequence-dependent setup times.

The GA follows a time-constrained multi-run parallelization strategy, referred to as Genetic Algorithm with Time-constrained Parallel Runs (GA-TPR). Its key features include:

- Time-constrained execution: each run is constrained by a duration that varies depending on the size of the problem instance, replacing the traditional fixed number of generations
- Parallelism: four independent runs are executed in parallel using the same parameter configuration but different random seeds
- Multi-run evaluation: the best solution among the four runs is selected as the output of the trial

This design enhances both the diversity and robustness of the search, enabling the algorithm to explore multiple regions of the solution space while efficiently utilizing multicore resources.

For each run, the genetic algorithm begins by generating an initial population of candidate schedules. Each individual is constructed using a hybrid strategy: jobs are first shuffled randomly to introduce diversity, then sorted according to the Longest Processing Time (LPT) heuristic, and finally assigned to the machines with the lowest current workload. GA-TPR features are:

- Hybrid initialization: random shuffling + LPT heuristic + load balancing
- Tournament selection with elitism preservation

Table 4: GA Parameter Configuration

Parameter	Description	TPE Optimized	Range/Value
pop_size	Population size	Yes	[30, 150]
cxpb	Crossover probability	Yes	[0.5, 0.95]
mutpb	Mutation probability	Yes	[0.05, 0.4]
elitism	Elite individuals	Yes	[1, 10]
tournament_size	Tournament size	Yes	[2, 15]
max_time	Execution time per run	No	Problem-dependent
num_runs	Parallel runs	No	4

- Machine-aware crossover preserving job sequences
- Dual mutation: inter-machine swapping + intra-machine reordering

The evolutionary process follows a classical yet customized structure. At each generation, elitism is applied to preserve a subset of the best-performing individuals. Parent selection is performed using tournament selection. The crossover operator is specifically designed for scheduling: for each machine, the job sequence is inherited from one of the two parents, selected at random, and any remaining unassigned jobs are allocated to the machine with the lowest workload.

Mutation is implemented through a hybrid operator that enhances diversity by swapping jobs between different machines and by randomly reordering jobs within a machine. The evolutionary cycle continues until a predefined time limit is reached.

Application Layer: Evaluates candidate solutions by computing makespan including setup times.

3.2.2 Parameters of the genetic algorithm

The main configuration parameters of the Genetic Algorithm are summarized in TABLE 4. This table is essential for understanding the optimization process and interpreting the experimental results.

Parameters marked with 'Yes' in the 'Optimized with TPE' column are subject to optimization using the Tree-structured Parzen Estimator algorithm implemented through Optuna. The indicated ranges represent the initial values but may be dynamically narrowed or expanded during adaptive optimization based on the best values found and the current exploration factor. Non-optimized parameters remain fixed throughout all trials.

3.2.3 Optimization workflow

FIGURE 2 illustrates the interaction between the TPE and the GA during the hyperparameter tuning process. The TPE proposes GA configurations, which are evaluated on a scheduling case study. The resulting makespan values are fed back to the TPE to iteratively refine future configurations.

FIGURE 3 presents the complete optimization workflow, showing the iterative interaction between TPE parameter suggestion and GA evaluation. The process begins with the definition of

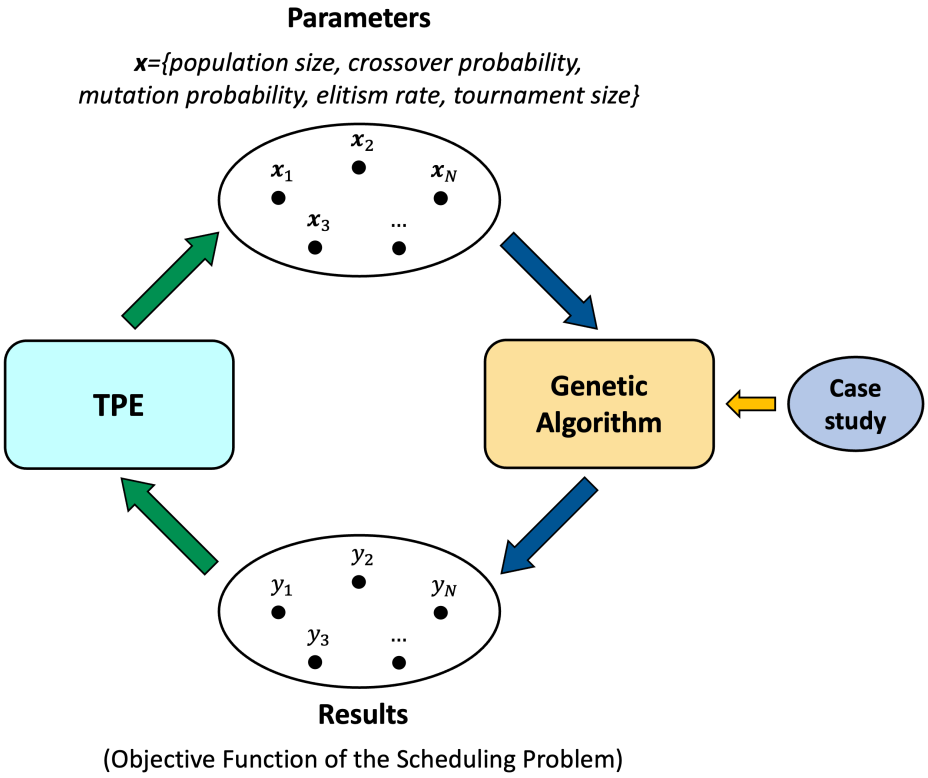

Figure 2: Interaction Between TPE and Genetic Algorithm in the Optimization Loop

the scheduling problem and the initialization of the GA, followed by the creation of an Optuna study and the iterative evaluation of parameter configurations. At each iteration, TPE suggests a new configuration, which is evaluated by executing the GA. The performance is then reported back to Optuna until the study concludes, at which point the best-performing parameters are selected.

The framework balances exploration and exploitation through adaptive mechanisms, enabling effective parameter tuning for complex scheduling instances with sequence-dependent setup times. This integrated approach addresses the nonlinear parameter interactions characteristic of combinatorial optimization problems.

For comprehensive theoretical details of the mathematical model constraints, setup time logic, solution representation theory, and detailed implementation specifics, please refer to Appendix A.

4. RESULTS

This section presents the experimental evaluation of the proposed TPE-GA methodology for solving parallel machine scheduling problems with sequence-dependent setup times. The key findings demonstrate that our approach consistently outperforms both exact optimization methods and alter-

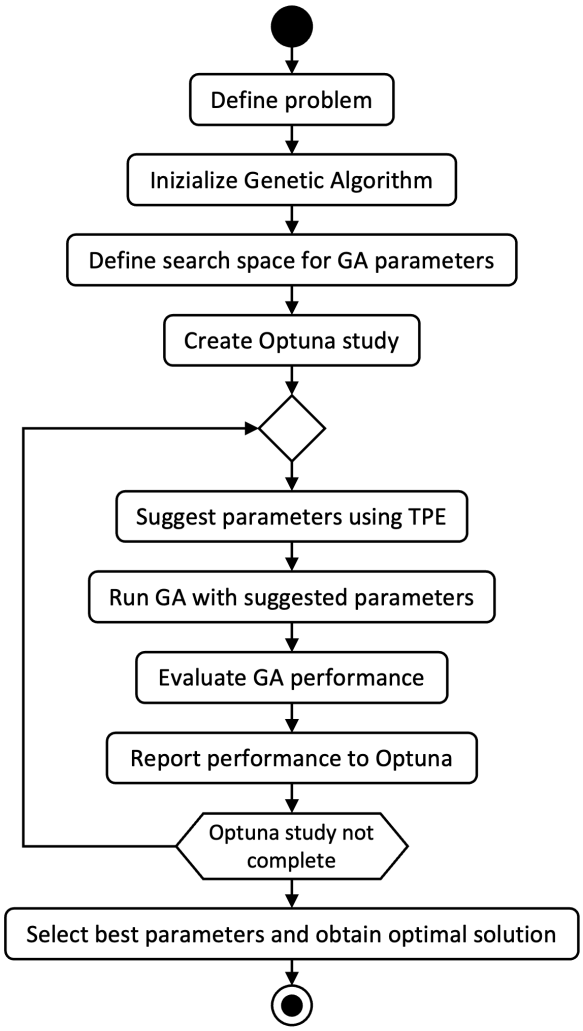

Figure 3: TPE-GA Optimization Workflow

native metaheuristics, achieving significant makespan reductions while maintaining computational efficiency across problem instances of varying complexity.

4.1 Experimental Setup

A comprehensive experimental campaign was conducted using a Python-based implementation on a macOS system (MacBook Air with Apple M1 chip and 8GB RAM). Four benchmark instances of increasing complexity were designed to evaluate algorithm performance and scalability, where complexity refers to the combined effect of job count, job type diversity, and machine availability. TABLE 5 provides a summary of the key characteristics of these benchmark instances.

Three solution approaches were compared under identical computational budgets: (1) the proposed TPE-GA method, (2) an exact MILP solver (SCIP) [39], and (3) an alternative TPE-SA configura-

Table 5: Benchmark Instance Characteristics

Instance	Jobs	Job Types	Machines
ID01	16	4	2
ID02	30	7	3
ID03	50	12	6
ID04	70	17	8

Table 6: Comparison of Exact Method, TPE-SA, and TPE-GA over 4 instances (objective: minimize). Differences are computed relative to TPE-GA.

Instance	Exact	TPE-SA	TPE-GA	Differences vs TPE-GA
ID01	275	272	**269**	Exact: +6 (2.18%) TPE-SA: +3 (1.10%)
ID02	547	530	**492**	Exact: +55 (10.06%) TPE-SA: +38 (7.17%)
ID03	575	555	**488**	Exact: +87 (15.13%) TPE-SA: +67 (12.07%)
ID04	657	514	**446**	Exact: +211 (32.12%) TPE-SA: +68 (13.23%)

tion that replaces the genetic algorithm with simulated annealing [40], while maintaining the same three-layer optimization architecture. Time limits were set proportionally to instance size (36, 39, 48, and 54 seconds per run for ID01-ID04 respectively), with the metaheuristic approaches using 40 Optuna trials for hyperparameter optimization.

4.2 Performance Comparison and Key Findings

The experimental results reveal three main findings that demonstrate the effectiveness of the proposed approach. First, TPE-GA consistently achieved the lowest makespan values across all benchmark instances, establishing it as the most effective method among those tested. Second, the performance advantage of TPE-GA becomes more pronounced as problem complexity increases, indicating strong scalability properties. Third, the integration of genetic algorithms with TPE-based hyperparameter optimization proves superior to alternative metaheuristic combinations.

The results in TABLE 6, show that TPE-GA outperformed the exact solver by margins ranging from 2.18% (ID01) to 32.12% (ID04), with an average improvement of 17.5%. When compared to the alternative TPE-SA approach, TPE-GA maintained consistent superiority across all instances, with improvements ranging from 1.10% to 13.23%. Notably, the performance gap widens substantially for larger, more complex instances, suggesting that the proposed method scales effectively with problem difficulty.

The superior performance of TPE-GA over the exact solver can be attributed to the method's ability to efficiently explore the solution space within the given time constraints, while the exact method may struggle to find optimal solutions for larger instances within the allocated time budget. The consistent advantage over TPE-SA demonstrates that genetic algorithms provide a more effective

Table 7: Optimized Genetic Algorithm Parameters for Best-Performing Solutions

Instance	Makespan	Pop. Size	Crossover	Mutation	Elitism	Tournament
ID01	269	70	0.928	0.306	2	10
ID02	492	45	0.784	0.108	1	10
ID03	488	40	0.503	0.052	4	15
ID04	446	35	0.728	0.051	4	14

search strategy than simulated annealing when combined with TPE-based hyperparameter optimization for this class of scheduling problems.

4.3 Algorithm Convergence Analysis

FIGURE 4 illustrates the convergence behavior of the TPE-GA algorithm across the 40 Optuna trials. The convergence plots reveal that the algorithm typically achieves significant improvements within the first 15-20 trials, followed by more gradual refinements. This pattern indicates effective balance between exploration and exploitation in the hyperparameter search process.

The boxplot analysis shows that solution quality becomes more consistent as problem size increases, suggesting that the algorithm's performance stabilizes for complex instances. This behavior indicates robustness in the optimization process and reliability in producing high-quality solutions across different problem scales.

4.4 Optimized Parameter Analysis

The TPE optimization process revealed meaningful patterns in parameter selection that provide insights into effective algorithm configuration for scheduling problems with sequence-dependent setup times. TABLE 7, presents the optimized parameters that yielded the best solutions for each instance.

Several important patterns emerge from the parameter optimization results. Population sizes converged to moderate values (35-70) despite the wide search range (30-150), suggesting that smaller populations with longer evolution times are more effective than larger populations with fewer generations. Crossover probabilities generally favored higher values, indicating the importance of genetic recombination in this problem domain. Mutation rates varied significantly across instances, reflecting the different landscape characteristics of problems with varying complexity. Tournament sizes consistently preferred higher selection pressure (10-15), while elitism remained conservative (1-4), suggesting that excessive preservation of elite solutions may limit exploration effectiveness.

4.5 Solution Quality Visualization

The following Gantt charts (FIGURE 5 - FIGURE 8) visualize the optimal scheduling solutions obtained by the TPE-GA method. These charts demonstrate the algorithm's effectiveness in balancing

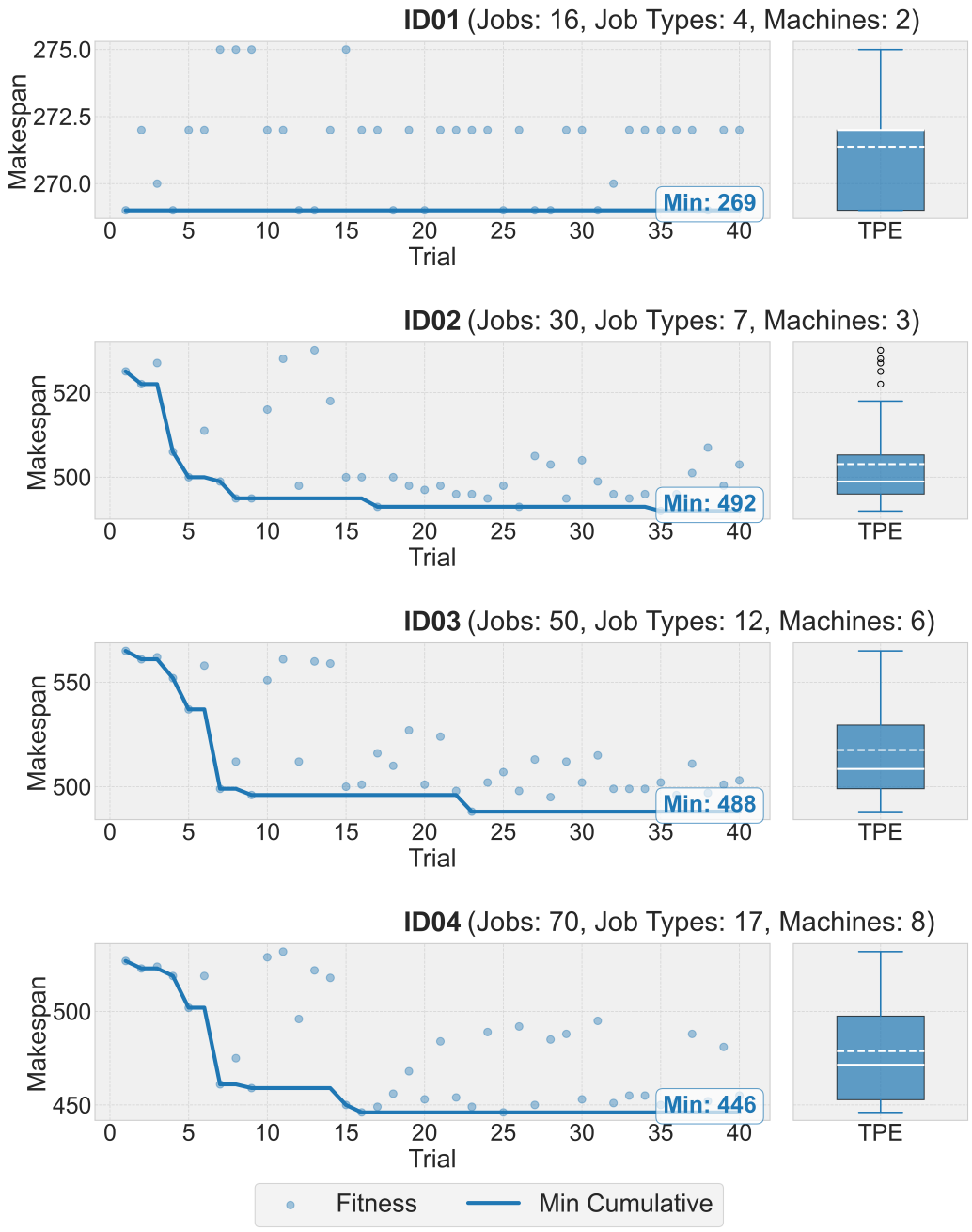

Figure 4: TPE-GA Convergence Behavior: Objective Function Evolution (Left) and Final Solution Distribution (Right)

machine workloads while minimizing overall completion time, showcasing practical applicability of the optimization results.

Additional quantitative results supporting FIGURE 5 - FIGURE 8, are provided in Appendix B.

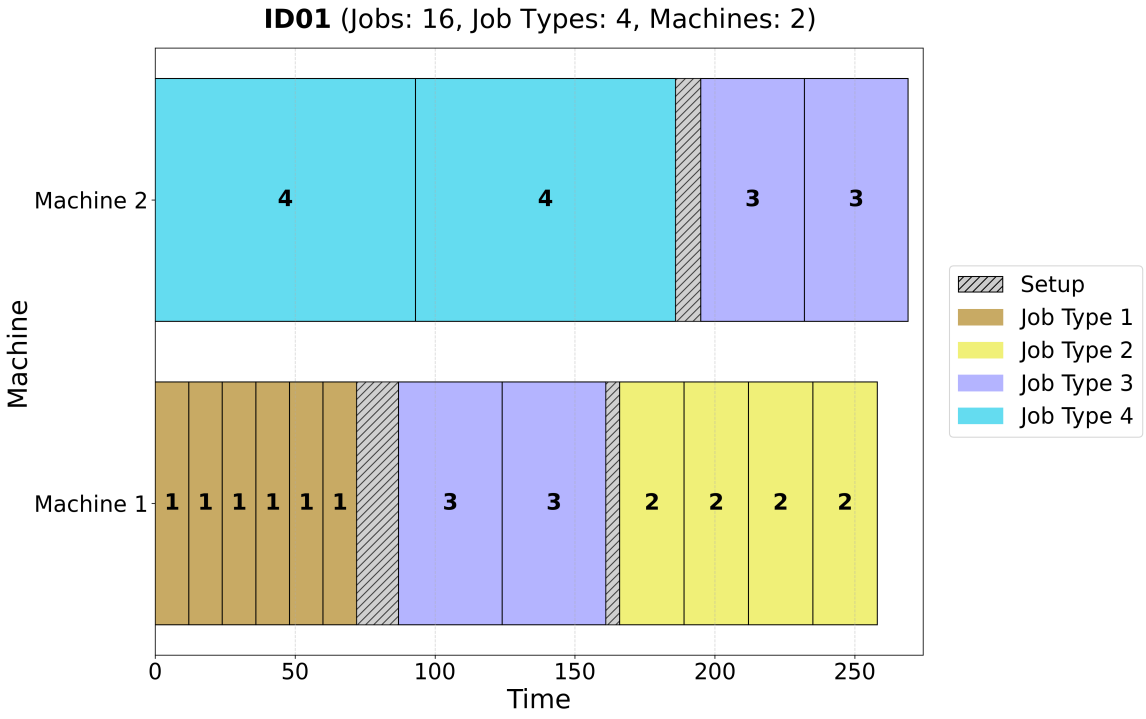

Figure 5: Gantt Chart for Optimal Scheduling Solution of ID01 using TPE-GA

The TPE-GA method effectively allocates jobs across all machines, minimizing idle times and optimizing job sequencing to reduce setup times. This behavior is evident across all instances, regardless of complexity, underscoring the robustness of the proposed approach in managing scheduling challenges.

4.6 Computational Efficiency Analysis

The actual execution times for the TPE-GA approach were 1'580, 1'690, 2'060, and 2'300 seconds for instances ID01 through ID04, respectively. These times exceeded the theoretical calculation (40 trials × time per run) due to multiprocessing overhead, but remained within acceptable bounds for practical scheduling applications. The time complexity scales reasonably with problem size, indicating the method's viability for real-world implementations.

The consistent outperformance of both exact and alternative metaheuristic methods, combined with reasonable computational requirements, establishes TPE-GA as an effective and practical solution for parallel machine scheduling problems with sequence-dependent setup times. The scalability demonstrated across increasing problem complexities further supports its potential for industrial applications where larger instances are common.

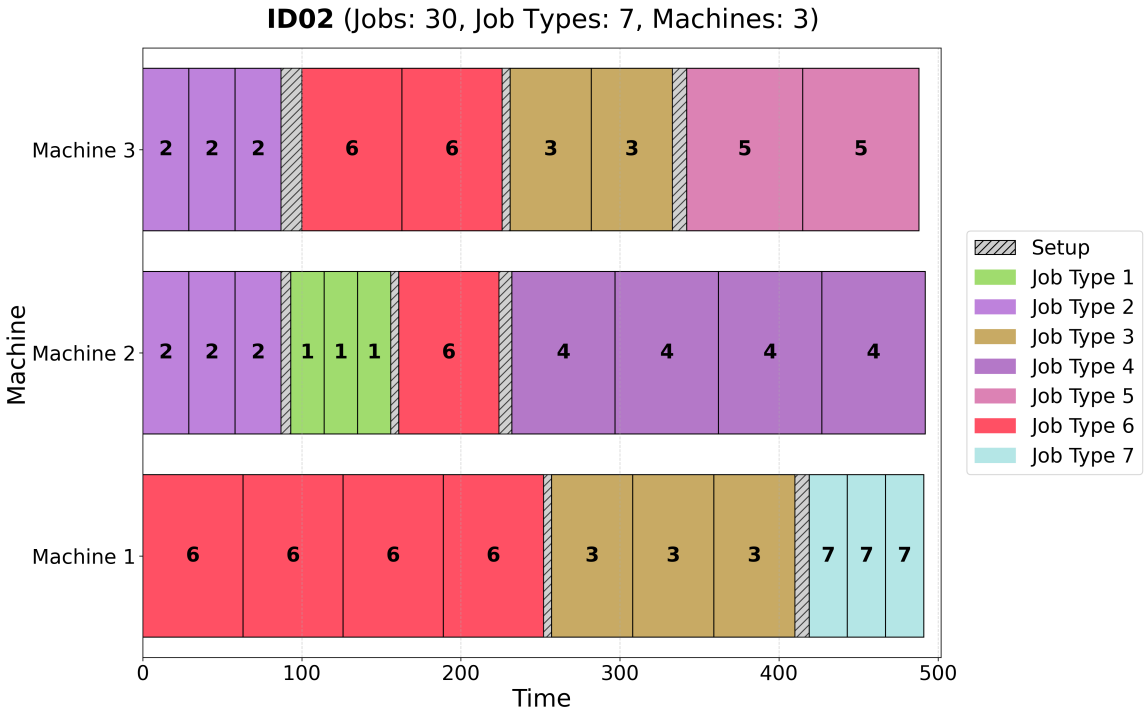

Figure 6: Gantt Chart for Optimal Scheduling Solution of ID02 using TPE-GA

5. CONCLUSIONS

This study has addressed the complex and computationally challenging problem of task scheduling in production environments, with a specific focus on minimizing makespan while accounting for realistic operational constraints such as sequence-dependent setup times (SDSTs). These constraints, often encountered in manufacturing systems, significantly increase the problem's complexity, requiring sophisticated optimization techniques beyond traditional deterministic methods.

To solve this NP-hard problem efficiently, we developed a tailored metaheuristic approach based on a Genetic Algorithm (GA). Unlike conventional heuristics that rely on manually tuned parameters, our method incorporates a learning-based parameter optimization mechanism through the Tree-structured Parzen Estimator (TPE). By leveraging information from past trials, TPE dynamically selects high-performing parameter configurations, enabling the GA to balance exploration and exploitation across the solution space more effectively.

The proposed method was evaluated on several realistic scheduling instances. The results demonstrate the successful achievement of the research objectives along three key dimensions. First, the modeling framework provides a flexible and robust structure capable of capturing industrial scheduling constraints while supporting the trade-off between computational efficiency and solution quality. Second, the GA component consistently produces diverse and high-quality solutions, offering a valuable range of alternatives for decision-makers. Third, the TPE-enhanced optimization yields a substantial improvement over a benchmark exact method, reducing makespan by an average of 15% across four problem instances, and up to 32% in the largest case—all within the same time

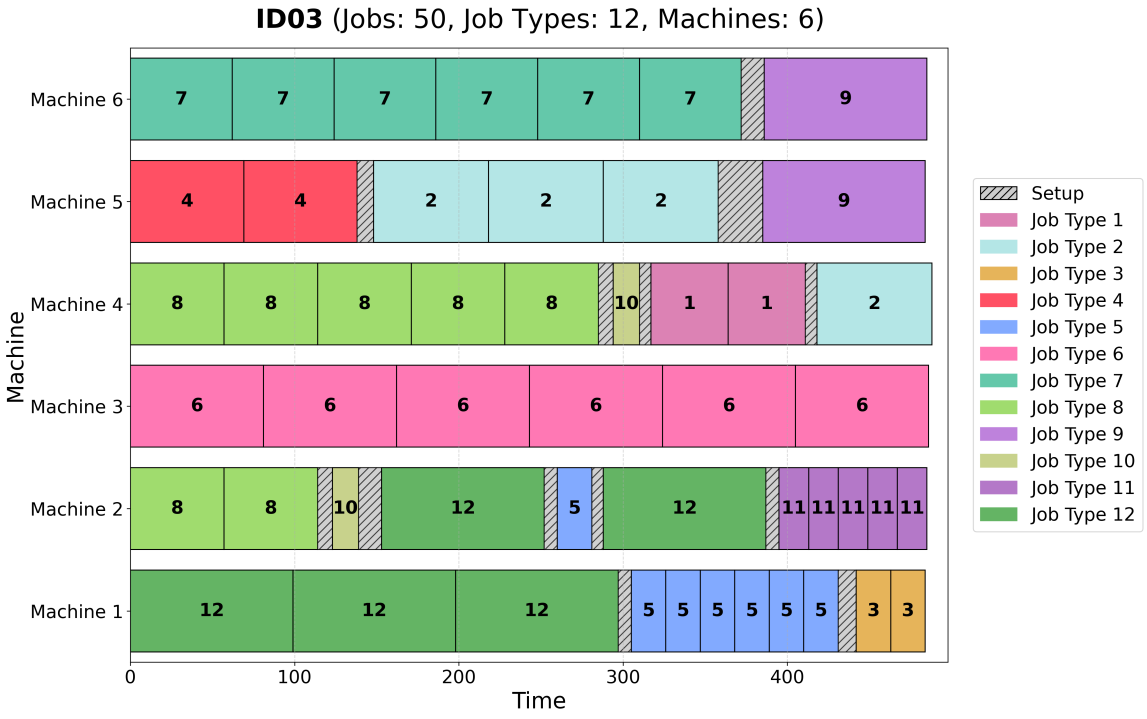

Figure 7: Gantt Chart for Optimal Scheduling Solution of ID03 using TPE-GA

budget. Applying TPE to Simulated Annealing instead of the GA results in a consistent performance decrease ranging from 1.10% to 13.23%, highlighting the greater effectiveness of the GA-based method.

These findings provide a strong and affirmative answer to the central research question: that integrating AI-driven tuning mechanisms into metaheuristic schedulers can significantly enhance performance in complex scheduling environments. The method's ability to adapt to problem characteristics without manual intervention represents a promising step toward more autonomous and intelligent optimization systems. Furthermore, the demonstrated robustness of the approach to random initialization, as shown in the seed sensitivity analysis, underscores its reliability and practical applicability.

The implications of this work extend beyond theoretical contributions. In practical terms, the proposed TPE-GA framework can be embedded into industrial scheduling dashboards, offering human operators the ability to generate, evaluate, and refine task assignments dynamically. This capability is particularly valuable in high-mix, low-volume manufacturing settings, where flexibility, responsiveness, and efficiency are critical. By minimizing makespan and reducing setup times through intelligent scheduling, organizations can achieve higher throughput, better resource utilization, and improved overall operational performance.

Future work may extend this framework to support multi-objective optimization, uncertainty-aware scheduling, and the integration of domain-specific heuristics or human-in-the-loop decision support.

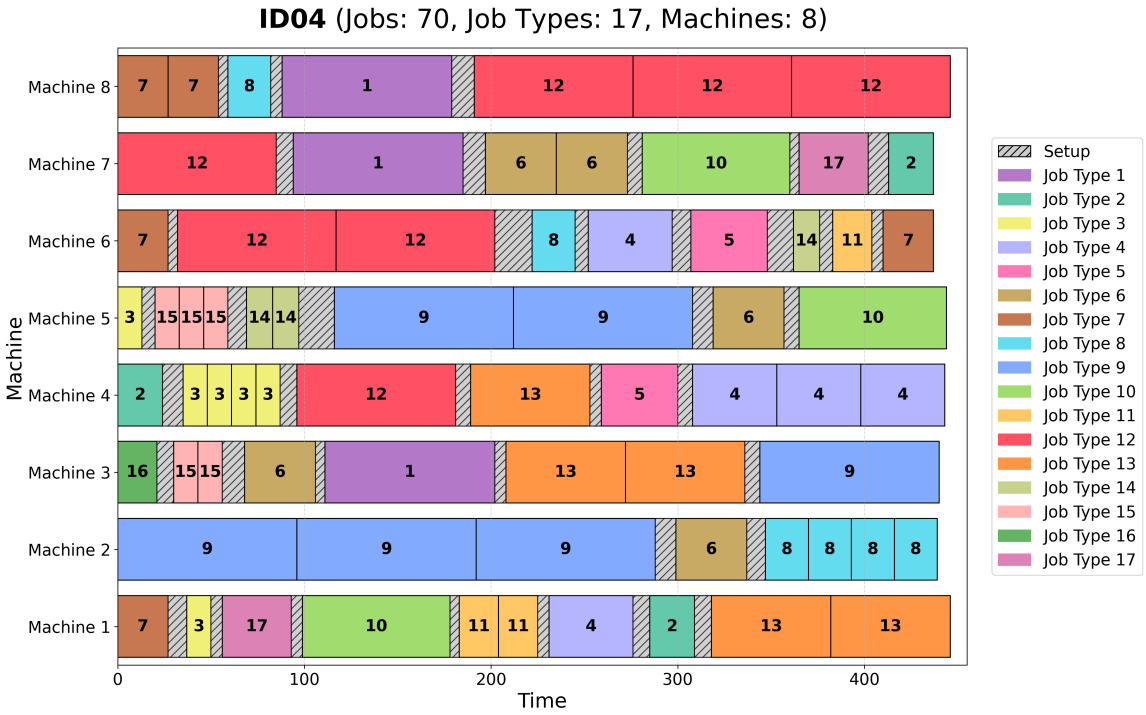

Figure 8: Gantt Chart for Optimal Scheduling Solution of ID04 using TPE-GA

These directions aim to further bridge the gap between theoretical optimization and the needs of real-world manufacturing systems.

6. ACKNOWLEDGMENTS

The authors express their gratitude for the administrative and technical assistance provided by the company Advantech S.r.l. in Lecce, Italy.

7. AUTHOR CONTRIBUTIONS

Conceptualization, F.N.; methodology, A.F.; validation, A.F. and F.N.; resources, G.P.; writing—original draft preparation, A.F. and F.N.; writing—review and editing, A.F. and F.N.; supervision, F.N. and G.P.; software, A.F.; project administration, G.P.; funding acquisition, G.P. All authors have read and agreed to the published version of the manuscript.

References

[1] Csalódi R, Süle Z, Jaskó S, Holczinger T, Abonyi J. Industry 4.0□Driven Development of Optimization Algorithms: A Systematic Overview. Complex. 2021;2021:6621235.

[2] Pinedo ML. Scheduling: Theory, Algorithms, and Systems. Springer. 2022.

[3] Blum C, Roli A. Metaheuristics in Combinatorial Optimization: Overview and Conceptual Comparison. ACM Comput Surv. (CSUR). 2003;35:268-308.

[4] Almufti SM, Shaban AA, Ali ZA, Ali RI, Fuente JD. Overview of Metaheuristic Algorithms. Polaris Global Journal of Scholarly Research and Trends. 2023;2:10-32.

[5] Xu S, Hall NG. Fatigue, Personnel Scheduling and Operations: Review and Research Opportunities. Eur J Oper Res. 2021;295:807-822.

[6] Koruca HI, Emek MS, Gulmez E. Development of a New Personalized Staff-Scheduling Method With a Work-Life Balance Perspective: Case of a Hospital. Ann Oper Res. 2023;328:793-820.

[7] Slama R, Slama I, Tlahig H, Slangen P, Ben-Ammar O. An Overview on Human-Centred Technologies, Measurements and Optimisation in Assembly Systems. Int J Prod Res. 2024;62:5336-5358.

[8] Ying KC, Pourhejazy P, Lin ZR. Scheduling With Sequence-Dependent Setup Times in Short-term Production Planning: A Main Path Analysis-Based Review. Oper Res Perspect. 2025;14:100340.

[9] Goerigk M, Hartisch M. An Introduction to Robust Combinatorial Optimization. Concepts, Models and Algorithms for Decision Making under Uncertainty. Springer Nature. 2024.

[10] Hesamoddin Tahami and Hengameh Fakhravar. A literature review on combining heuristics and exact algorithms in combinatorial optimization. *European Journal of Information Technologies and Computer Science*, 2(2):6–12, Apr. 2022.

[11] Mohamad Hjeij and Arnis Vilks. A brief history of heuristics: how did research on heuristics evolve?, 12 2023.

[12] Tahami H, Fakhravar H. A Literature Review on Combining Heuristics and Exact Algorithms in Combinatorial Optimization. Eur J Inf Technol Comput Sci. 2022;2:6-12.

[13] Hjeij M, Vilks A. A Brief History of Heuristics: How Did Research on Heuristics Evolve? Humanit Soc Sci Commun. 2023;10:1-15.

[14] Li C, Han Y, Zhang B, Wang Y, Li J, et al. A Novel Collaborative Iterative Greedy Algorithm for hybrid Flowshop Scheduling Problem With Batch Processing Machines and Variable Sublots. Int J Prod Res. 2024;62:4076-4096.

[15] Rossi FL, Nagano MS. Heuristics and Iterated Greedy Algorithms for the Distributed Mixed No-Idle Flowshop With Sequence-Dependent Setup Times. Comput Ind Eng. 2021;157:107337.

[16] Han X, Han Y, Zhang B, Qin H, Li J, et al. An Effective Iterative Greedy Algorithm for Distributed Blocking Flowshop Scheduling Problem With Balanced Energy Costs Criterion. Appl. Soft Comput. 2022;129:109502.

[17] Katoch S, Chauhan SS, Kumar V. A Review on Genetic Algorithm: Past, Present, and Future. Multimed Tools Appl. 2021;80:8091-8126.

[18] Lee CK. A Review of Applications of Genetic Algorithms in Operations Management. Eng Appl Artif Intell. 2018;76:1-12.

[19] Cheung W, Zhou H. Using Genetic Algorithms and Heuristics for Job Shop Scheduling With Sequence-Dependent Setup Times. Ann Oper Res. 2001;107:65-81.

[20] Vallada E, Ruiz R. A Genetic Algorithm for the Unrelated Parallel Machine Scheduling Problem With Sequence Dependent Setup Times. Eur J Oper Res. 2011;211:612-622.

[21] Watanabe S. Tree-Structured Parzen Estimator: Understanding Its Algorithm Components and Their Roles for Better Empirical Performance. 2023. ArXiv Preprint: https://arxiv.org/abs/2304.11127

[22] Michael E, Ma H, Li H, Qi S. An Optimized Framework for Breast Cancer Classification Using Machine Learning. Biomed Res Int. 2022;2022:8482022.

[23] Rizkallah LW. Optimizing SVM Hyperparameters for Satellite Imagery Classification Using Metaheuristic and Statistical Techniques. Int J Data Sci Anal. 2025.

[24] Jiang BN, Zhang YY, Zhang ZY, Yang YL, Song HL. Tree-Structured Parzen Estimator Optimized-Automated Machine Learning Assisted by Meta–Analysis for Predicting Biochar–Driven N2O Mitigation Effect in Constructed Wetlands. J Environ Manage. 2024;354:120335.

[25] ASayah A, Aqil S, Lahby M. Improved Bio-Inspired Algorithms for Scheduling Distributed No-Waiting Flow Shop with Setup Times. Evol Intell. 2025;18:68.

[26] Xu Y, Zhang M, Wang D, Yang M, Liang C. Hybrid Gaussian Quantum Particle Swarm Optimization and Adaptive Genetic Algorithm for Flexible Job-Shop Scheduling Problem. Eng Appl Artif Intell. 2025;154:110882.

[27] Liu F, Gao K, Słowik A, Suganthan PN. Ensemble Artificial Bee Colony Algorithm and Q-Learning for Multi-Objective Distributed Heterogeneous Flowshop Scheduling Problems With Sequence-Dependent Setup Time. Complex Syst Model and Simul. 2025:1-14.

[28] Liu F, Gao K, Słowik A, Suganthan PN. Ensemble Artificial Bee Colony Algorithm and Q-Learning for Multi-Objective Distributed Heterogeneous Flowshop Scheduling Problems With Sequence-Dependent Setup Time. Complex Syst Model and Simul. 2025:1-14.

[29] Zhang Z, Yu S, Tang Q, Zhang L, Li Z, et al. A Matheuristic-Based Self-Learning Approach for Distributed Heterogeneous Assembly Flowshop Scheduling With Multiple Assembly Factories and Make-To-Order Delivery. Swarm and Evolutionary Computation. 2025;97:101996.

[30] Xiong F, Zhou K, Ping A, Jing L. Scheduling Distributed Heterogeneous Parallel Precast Flowshop With Shared Resources via Logic-Based Benders Decomposition. Adv Eng Inform. 2025;67:103543.

[31] Deenen P, Nuijten W, Akcay A. Scheduling a Real-World Photolithography Area With Constraint Programming. IEEE Trans Semicond Manuf. 2023;36:590-598.

[32] Wu S, Liu L. Green hybrid Flow Shop Scheduling Problem Considering Sequence Dependent Setup Times and Transportation Times. IEEE Access. 2023 Apr 21;11:39726-37.

[33] Otamendi U, Martinez I, Belaunzaran X, Artetxe A, Franco J, et al. An Analytics-Based Framework for Optimizing Resource Allocation and Preemptive Scheduling in Manufacturing. Decis Anal. 2025;16:100596.

[34] Lin C, Cao Z, Zhou M. Learning-Based Grey Wolf Optimizer for Stochastic Flexible Job Shop Scheduling. IEEE Trans Autom Sci Eng. 2022;19:3659-3671.

[35] Zhai H, Sands T. Comparison of Deep Learning and Deterministic Algorithms for Control Modeling. Sensors. 2022;22:6362.

[36] Xu J, Sands T. Autonomous Drone Electronics Amplified With Pontryagin-Based Optimization. Electronics. 2023;12:2541.

[37] Wang Y, Han Y, Wang Y, Li J, Gao K, et al. An Effective Two-Stage Iterated Greedy Algorithm for Distributed Flowshop Group Scheduling Problem with Setup Time. Expert Syst Appl. 2023;233:120909.

[38] Almufti SM, Shaban AA, Ali ZA, Ali RI, Fuente JD. Overview of Metaheuristic Algorithms. Polaris Global Journal of Scholarly Research and Trends. 2023;2:10-32.

[39] Achterberg T. SCIP: Solving Constraint Integer Programs. Mathematical Programming Computation. 2009;1:1-41.

[40] Joseph SB, Dada EG, Abidemi A, Oyewola DO, Khammas BM. Metaheuristic Algorithms for PID Controller Parameters Tuning: Review, Approaches and Open Problems. Heliyon. 2022;8:e09399

Appendix A. Research Model and Solution Method

A.1 Detailed Mathematical Model Constraints

The MILP formulation minimizes makespan subject to precedence and sequencing constraints:

$$\min \quad C_{\max} \tag{1}$$

Subject to:

$$C_j \leq C_{\max}, \quad \forall j \in \mathcal{N} \tag{2}$$

$$t_{ij} + p_{ij} + s^i_{\tau(j)\tau(k)} x^i_{jk} \leq t_{ik} + (1 - x^i_{jk}) \cdot L, \quad \forall i \in \mathcal{M}, \forall j, k \in \mathcal{N}, j \neq k \tag{3}$$

$$\sum_{j=0, j\neq k}^{n} x^i_{jk} = 1, \quad \forall i \in \mathcal{M}, \forall k \in \mathcal{N} \tag{4}$$

$$\sum_{k=0, k\neq j}^{n} x^i_{jk} = 1, \quad \forall i \in \mathcal{M}, \forall j \in \mathcal{N} \tag{5}$$

$$x^i_{jk} \in \{0, 1\} \tag{6}$$

$$t_{ij} \geq 0 \tag{7}$$

The objective of the scheduling problem is to minimize the makespan ($C_{\max}$), which represents the completion time of the last job to finish processing (see Equation (1)). The complete mathematical model includes several constraints, each serving a specific purpose in the scheduling formulation. Constraint (2) ensures that the completion time of each job does not exceed the overall makespan. This implies that all jobs must be completed within the total time required to finish all tasks.

Constraint (3) enforces the sequencing logic. Specifically, if job J_k is scheduled immediately after job J_j on the same machine, indicated by $x^i_{jk} = 1$, then the start time of J_k must not be earlier than the completion time of J_j plus the setup time required for transitioning from job type $\tau(j)$ to $\tau(k)$ on machine M_i. The term $(1 - x^i_{jk}) \cdot L$ effectively deactivates this constraint when $x^i_{jk} = 0$, i.e., when J_k does not directly follow J_j.

Constraints (4) and (5) define the sequencing structure of the solution. They ensure that each job has exactly one predecessor and one successor on a given machine, thereby forming a valid, non-cyclic sequence of jobs assigned to each machine. To facilitate this sequencing, a dummy node (denoted as 0) is introduced to represent the starting and ending points of each sequence. This dummy node has no processing or setup time and does not affect the makespan.

Constraints (6) and (7) specify the domains of the decision variables. Binary variables indicate the immediate succession relationships between jobs, while continuous variables representing job start times are constrained to be non-negative. In these constraints, L denotes a sufficiently large positive constant, a standard technique in MILP modeling to handle disjunctive constraints.

A.2 Setup Time Logic and Solution Representation

In this formulation, setup times are not specific to individual jobs but depend on the job types involved in the sequence. The time required to prepare a machine for the next job is determined by the transition from the type of the preceding job to the type of the following one. For instance, in a sequence of jobs with types A → A → B → C → A, a setup time is incurred only at the transitions A → B, B → C, and C → A. No setup is needed between the two consecutive jobs of type A. This

abstraction reflects realistic production environments where setup operations are standardized for job families rather than individual jobs.

The solution to the scheduling problem can be naturally represented by a Gantt chart, which illustrates the start and end times of each operation assigned to every machine. According to Cheung (2001), when the performance measure is regular—such as minimizing the makespan $C_{\max}$—the optimal solution belongs to the class of semi-active schedules. Consequently, the schedule can be compactly represented as a set of job permutations for each machine, as shown below:

$$\begin{aligned} M_1 &: O_{1,1}, O_{1,2}, \ldots, O_{1,n_1} \\ M_2 &: O_{2,1}, O_{2,2}, \ldots, O_{2,n_2} \\ &\vdots \\ M_m &: O_{m,1}, O_{m,2}, \ldots, O_{m,n_m} \end{aligned}$$

Here, n_i represents the number of jobs assigned to machine M_i, and the total number of jobs satisfies $\sum_{i=1}^{m} n_i = n$.

A.3 Detailed TPE Implementation

The objective method, implemented within the `TPESchedulerOptimizer` class, executes the Genetic Algorithm and returns the resulting makespan, see FIGURE 9. This method incorporates a complementary adaptation mechanism that dynamically adjusts the search bounds for key parameters throughout the optimization process. During the initial phase (trials 1–15), each parameter is sampled within its full predefined range to encourage diversity. In the adaptive phase (trials 16–40), the bounds are narrowed around the most promising values observed thus far.

This contraction is guided by real-time indicators such as the coefficient of variation of recent results (to quantify stability), the improvement rate in solution quality, and the elapsed time since the last significant improvement. The adjustment is controlled by a dynamic exploration factor, which expands or contracts the sampling ranges based on performance trends. For example, if the algorithm converges prematurely around suboptimal configurations, the ranges are widened to reintroduce diversity. Conversely, if promising improvements are observed, the ranges are narrowed to intensify the search in the vicinity of high-performing configurations.

This dual-adaptation mechanism enables a fine-grained balance between diversification and intensification. It improves the algorithm's ability to escape local optima while progressively converging toward high-quality solutions. The integration of these adaptive elements is particularly beneficial in complex combinatorial problems such as scheduling with sequence-dependent setup times, where the interaction between parameters and objective performance can be highly nonlinear and instance-dependent.

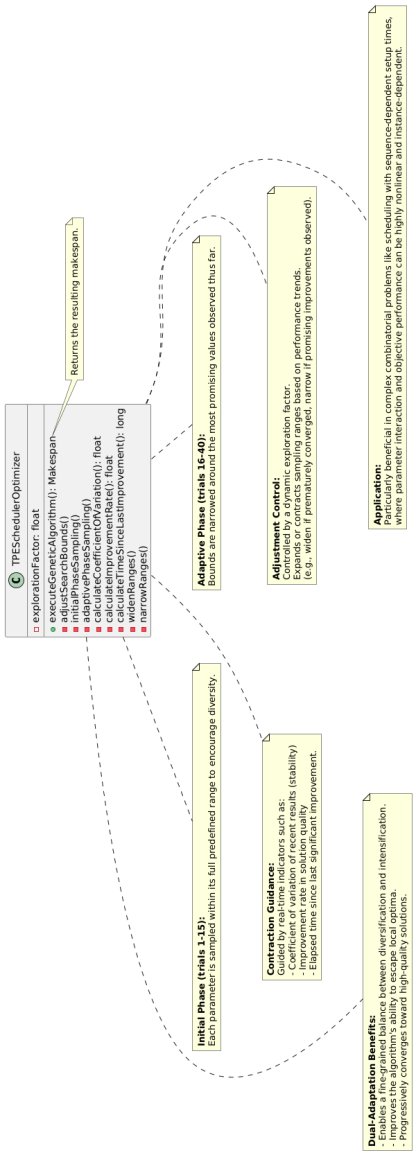

Figure 9: TPE Implementation

Appendix B. Supplementary Quantitative Analysis

This appendix provides a detailed numerical analysis to complement the comparative visualizations presented in FIGURE 5 - FIGURE 8. While the main figures illustrate the relative performance of the methods, the supplementary tables below offer standard statistical metrics to support a more accurate and quantitative interpretation.

TABLE 8–TABLE 11 present the following key performance indicators (KPIs), computed for each machine and each problem instance:

Table 8: Summary of machine-specific performance indicators for scheduling instance ID01

Resource	**No. Jobs**	**No. Setup**	**Busy time**	**Setup Time**	**Makespan**	**Utilization** (%)
Machine 1	12	2	238	20	258	92.2
Machine 2	2	1	260	9	269	96.7

Table 9: Summary of machine-specific performance indicators for scheduling instance ID02

Resource	**No. Jobs**	**No. Setup**	**Busy time**	**Setup Time**	**Makespan**	**Utilization** (%)
Machine 1	10	2	477	14	491	97.1
Machine 2	11	3	473	19	492	96.1
Machine 3	9	3	461	27	488	94.5

- **Active Processing Time (BusyTime)**: The total time spent processing all jobs assigned to a machine.
- **Total Setup Time (Setup Time)**: The cumulative setup time required between all consecutive jobs on a machine.
- **Machine Makespan (Makespan)**: The completion time of the last job processed on a machine.
- **Utilization Ratio (Utilization)**: The ratio of active processing time to the machine's makespan.

Appendix C. Sensitivity Analysis to Random Seed

The proposed TPE-GA implementation accepts a user-defined random seed, which controls the stochastic components of both the Tree-structured Parzen Estimator and the Genetic Algorithm. To verify that the algorithm's performance is not overly dependent on the chosen seed, a sensitivity analysis was conducted on a representative scheduling instance with 30 jobs, 7 job types, and 3 parallel machines.

The algorithm was executed 25 times using distinct random seeds. Each execution involved 25 TPE trials, with each trial launching 4 parallel GA runs, constrained to 6 seconds each. The resulting makespan values—one per execution—are shown in TABLE 12.

Table 10: Summary of machine-specific performance indicators for scheduling instance ID03

Resource	**No. Jobs**	**No. Setup**	**Busy time**	**Setup Time**	**Makespan**	**Utilization** (%)
Machine 1	11	2	465	19	484	96.1
Machine 2	11	5	439	46	485	90.5
Machine 3	6	0	486	0	486	100.0
Machine 4	9	3	465	23	488	95.3
Machine 5	6	2	447	37	484	92.4
Machine 6	7	1	471	14	485	97.1

Table 11: Summary of machine-specific performance indicators for scheduling instance ID04

Resource	**No. Jobs**	**No. Setup**	**Busy time**	**Setup Time**	**Makespan**	**Utilization** (%)
Machine 1	10	7	395	51	446	88.6
Machine 2	8	2	418	21	439	95.2
Machine 3	8	5	400	40	440	90.9
Machine 4	11	5	401	42	443	90.5
Machine 5	10	5	389	55	444	87.6
Machine 6	9	7	368	69	437	84.2
Machine 7	7	5	392	45	437	89.7
Machine 8	7	3	423	23	446	94.8

Table 12: Makespan values across 25 runs with different random seeds

ID	**Seed**	**Makespan**
01	1	491
02	7	496
03	8	497
04	13	495
05	21	493
06	42	495
07	55	494
08	89	493
09	123	495
10	144	496
11	233	495
12	377	495
13	456	496
14	610	495
15	789	495
16	1011	495
17	1234	493
18	2022	497
19	2023	495
20	2024	494
21	2025	494
22	2468	494
23	3141	495
24	5555	493
25	7777	493

Descriptive statistics are reported in TABLE 13. The coefficient of variation (CV), defined as CV = $(\sigma/\mu) \times 100$, was used to measure relative variability. With a CV of only 0.28%, the makespan values exhibit extremely low dispersion. The narrow range (491–497) and low interquartile range (1) further confirm the stability of the algorithm under varying seed values.

Table 13: Summary statistics for makespan across different random seeds

Mean (μ)	**Std Dev** (σ)	**CV (%)**	**Min**	**Max**	**Range**	**Mode**	**Q1**	**Q2**	**Q3**	**IQR**
494.56	1.39	0.28	491	497	6	495	494	495	495	1

These results demonstrate that the TPE-GA method provides highly consistent outcomes across different random seeds, confirming its robustness to stochastic variability and ensuring reproducibility without compromising performance.

Appendix D. Glossary

A glossary of all acronyms and variables, organized by section, is reported. TABLE 14 reports the Glossary of Acronyms. Glossary of Variables (from TABLE 3 and TABLE 4) is reported in TABLE 15.

Table 14: Glossary of Acronyms

Acronym	**Definition**	**First Appearance / Context**
ABC	Artificial Bee Colony	Background
ABC-PR	Artificial Bee Colony with Path Relinking	Background
DAI	Deterministic Artificial Intelligence	Introduction
DFSSPs	Distributed Flow Shop Scheduling Problems	Background
DHFSP	Distributed Heterogeneous Flowshop Scheduling Problems	Background
GA	Genetic Algorithm	Introduction
GA-IIS	Genetic Algorithm with Individual Improvement Scheme	Background
GA-TPR	Genetic Algorithm with Time-constrained Parallel Runs	Materials and Methods
HGQPSO-AGA	Hybrid of Gaussian Quantum Particle Swarm Optimization and Adaptive Genetic Algorithm	Background
ID01…04	Instance Data 01…04	Results
IG	Iterative Greedy	Background
JSSP	Job Shop Scheduling Problem	Background
LPT	Longest Processing Time	Materials and Methods
MBO-VNS	Migrating Birds Optimization with Variable Neighborhood Search	Background
MILP	Mixed Integer Linear Programming	Background
MIP	Mixed Integer Programming	Background
MSIG	Matheuristic-based Self-learning Iterated Greedy	Background
$Pm\|s_{\tau(j)\tau(k)}\|C_{\max}$	Parallel machine scheduling problem with sequence-dependent setup times, aiming to minimize makespan	Materials and Methods
SCIP	Solving Constraint Integer Programs (solver)	Results
SDSTs	Sequence-Dependent Setup Times	Introduction
TPE	Tree-structured Parzen Estimator	Introduction
UL-LBBD-SSR	Logic-Based Benders Decomposition approach with Strong Subproblem Relaxations and enhanced bounds	Background

Table 15: Glossary of Variables (from TABLE 3 and TABLE 4)

Variable / Parameter	Description	Context
$\mathcal{N}$	Job set $\{1, 2, \ldots, n\}$	Mathematical Model
$\mathcal{M}$	Machine set $\{1, 2, \ldots, m\}$	Mathematical Model
T	Set of job types	Mathematical Model
$\tau(j)$	Type of job j	Mathematical Model
$s^i_{\tau(j)\tau(k)}$	Setup time on machine M_i for type transition $\tau(j) \rightarrow \tau(k)$	Mathematical Model
p_{ij}	Processing time of job J_j on machine M_i	Mathematical Model
t_{ij}	Starting time of job J_j on machine M_i	Mathematical Model
C_j	Completion time of job J_j, equal to $t_{ij} + p_{ij}$	Mathematical Model
$C_{\max}$	Makespan (maximum completion time)	Mathematical Model
x^i_{jk}	Job Order: 1 if job J_k follows job J_j on machine M_i, 0 otherwise	Mathematical Model
pop_size	Population size	Genetic Algorithm
$cxpb$	Crossover probability	Genetic Algorithm
$mutpb$	Mutation probability	Genetic Algorithm
$elitism$	Elite individuals	Genetic Algorithm
$tournament\_size$	Tournament size	Genetic Algorithm
max_time	Execution time per run	Genetic Algorithm
num_runs	Parallel runs	Genetic Algorithm