Research Square

Preprints are preliminary reports that have not undergone peer review.
They should not be considered conclusive, used to inform clinical practice,
or referenced by the media as validated information.

REMoH: A Reflective Evolution of Multi-objective Heuristics approach via Large Language Models

Diego Fornies-Tabuenca
Vicomtech
Alejandro Uribe
auribev1@eafit.edu.co
Universidad EAFIT
Urtzi Otamendi
University of the Basque Country (UPV/EHU), Donostia-San Sebastián
Arkaitz Artetxe
Vicomtech
Juan Carlos Rivera
Universidad EAFIT
Oier Lopez Lacalle
University of the Basque Country (UPV/EHU), Donostia-San Sebastián

Research Article

Keywords: Large Language Models, Multi-Objective Optimization, Heuristic Generation, Scheduling, Benchmarking

Posted Date: March 5th, 2026

DOI: https://doi.org/10.21203/rs.3.rs-8837965/v1

License: This work is licensed under a Creative Commons Attribution 4.0 International License.
Read Full License

Additional Declarations: No competing interests reported.

REMoH: A Reflective Evolution of Multi-objective Heuristics approach via Large Language Models

Diego Fornies-Tabuenca[1,3], Alejandro Uribe[1,3*†], Urtzi Otamendi[1,2†], Arkaitz Artetxe[1†], Juan Carlos Rivera[3†], Oier Lopez de Lacalle[4†]

[1*]Vicomtech Foundation, Basque Research and Technology Alliance (BRTA), Donostia-San Sebastián, 20009, Spain.
[2]Department of Computer Sciences and Artificial Intelligence, University of the Basque Country (UPV/EHU), Manuel Lardizabal Pasealekua 1, Donostia-San Sebastián, 20018, Gipuzkoa, Spain.
[3]School of Applied Sciences and Engineering, Universidad EAFIT, Carrera 49 N 7 Sur-50, Medellín, 05001, Antioquia, Colombia.
[4]HiTZ Basque Center for Language Technology, University of the Basque Country (UPV/EHU), Manuel Lardizabal Pasealekua 1, Donostia-San Sebastián, 20018, Gipuzkoa, Spain.

*Corresponding author(s). E-mail(s): auribev1@eafit.edu.co, auribe@vicomtech.org;
Contributing authors: diegoforniestabuenca@gmail.com; uotamendi@vicomtech.org; aartetxe@vicomtech.org; jrivera6@eafit.edu.co; oier.lopezdelacalle@ehu.eus;
†These authors contributed equally to this work.

Abstract

Multi-objective optimization is a central challenge in complex decision-making problems. Here, solution quality depends on balancing competing performance criteria. Classical exact and heuristic optimization methods have demonstrated strong results. However, they typically require substantial problem-specific modeling effort and often face scalability limitations. Recent advances in Large Language Models (LLMs) offer new possibilities for automating the design of heuristics. Yet, their effectiveness relative to established optimization paradigms remains insufficiently benchmarked. This paper introduces *Reflective Evolution of Multi-objective Heuristics* (REMoH), a hybrid framework that integrates

LLM-generated heuristics within an NSGA-II evolutionary process. The primary contribution of this work is a systematic benchmarking of LLM-driven heuristics against exact mathematical programming, classical heuristics, and learning-based methods in a multi-objective scheduling context. Experiments on the Flexible Job Shop Scheduling Problem show that LLM-generated heuristics achieve competitive, well-balanced trade-offs across objectives. They often approach the performance of exact solvers while requiring significantly less modeling effort and computational overhead. A lightweight reflective mechanism is included to guide heuristic evolution. However, it plays a secondary role in the overall framework. The results highlight the potential of LLMs as practical heuristic generators for multi-objective optimization. They offer a flexible and scalable alternative to traditional approaches.

Keywords: Large Language Models, Multi-Objective Optimization, Heuristic Generation, Scheduling, Benchmarking

1 Introduction

Optimization remains a central challenge across many scientific and engineering domains, where decisions must be taken under increasing problem scale, structural complexity, and conflicting objectives [1, 2]. Classical optimization techniques, ranging from exact mathematical models to handcrafted heuristics and metaheuristics, have demonstrated strong performance, particularly in well-structured settings. However, these approaches typically require substantial domain expertise for modeling, algorithm design, and parameter tuning, which limits their scalability, adaptability, and transferability to new or highly dynamic problem instances [3].

Large Language Models (LLMs) have recently offered a new approach to solving optimization problems, shifting from manual algorithm design to strategies grounded in data and prior knowledge. Unlike traditional methods or Reinforcement Learning (RL), which often require extensive training and custom problem-solving, LLMs leverage pre-trained knowledge and reasoning to develop optimization strategies with minimal initial setup, enabling rapid development. This has sparked interest in using LLMs not only as solvers but also as general tools for creating and supporting decisions. However, applying LLMs directly to engineering problems can be challenging, as their outputs may not align with real-world constraints or practical requirements. There must be clear ways to guide and correct their suggestions.

A promising line of research lies in hybrid optimization frameworks that combine LLMs with established optimization paradigms. Classical algorithms provide well-defined evaluation, selection, and convergence properties. In contrast, LLMs contribute flexibility, abstraction, and automation for heuristic synthesis [1]. Evolutionary algorithms (EAs) are well-suited to this integration. They naturally support population-based search, black-box evaluation, and multi-objective optimization. When combined with LLM-generated heuristics, EAs allow systematic benchmarking of language-driven strategies against exact methods, handcrafted heuristics, and

learning-based approaches. This offers a principled way to assess their effectiveness and limitations [4].

Despite recent progress in LLM-driven heuristic and algorithm generation [1, 5, 6], a clear gap remains in rigorous benchmarking within multi-objective optimization. Empirical evidence comparing LLM-generated heuristics to classical exact models and state-of-the-art heuristics under consistent evaluation criteria is limited. This limitation is especially relevant in industrial scheduling, where competing objectives must be explicitly managed rather than reduced to a single aggregated metric [7].

In this work, this gap is addressed by introducing *Reflective Evolution of Multi-objective Heuristics* (REMoH), a hybrid framework with two key novelties: embedding LLM-generated heuristics into a multi-objective evolutionary process based on NSGA-II and integrating a reflective feedback mechanism to guide their adaptation. REMoH is novel in that it enables systematic and principled benchmarking of LLM-driven heuristics against exact mathematical programming, classical heuristics, and learning-based baselines in a multi-objective setting, something prior approaches do not provide. The reflective mechanism in REMoH provides structured feedback to facilitate heuristic evolution, complementing rather than dominating the overall framework and further distinguishing it from existing methods.

The focus on multi-objective optimization is highly relevant to real-world scheduling and resource allocation problems, in which decision quality depends on balancing conflicting criteria such as makespan, workload distribution, and robustness. By using Pareto-based evaluation rather than scalarized objectives, REMoH ensures a fair and transparent comparison across different optimization paradigms, highlighting both the strengths and limitations of LLM-generated heuristics relative to established methods.

The main contributions of this paper can be summarized as follows:

- A hybrid multi-objective optimization framework that integrates LLM-generated heuristics within an NSGA-II evolutionary process, enabling direct comparison with exact, heuristic, and learning-based methods.
- A comprehensive benchmarking study on the Flexible Job Shop Scheduling Problem (FJSSP), evaluating LLM-driven heuristics against mathematical programming models and state-of-the-art baselines under consistent multi-objective metrics.
- To bridge a gap identified in [8], this work conducts an empirical evaluation of LLM-generated heuristics, showcasing their ability to maintain modeling flexibility and accommodate complex constraints with limited human intervention.

The remainder of the paper is organized as follows. Section 2 reviews related work on LLMs in optimization with emphasis on heuristic generation and benchmarking. Section 3 describes the REMoH framework. Section 4 presents the experimental setup and baseline methods. Section 5 discusses the benchmarking results. Finally, Section 6 concludes the paper and outlines future research directions.

2 State of the Art

2.1 Classical Optimization Methods for the FJSSP

Exact methods, heuristics, metaheuristics, and hybrid approaches have traditionally dominated research on the Flexible Job Shop Scheduling Problem (FJSSP). Exact formulations based on Mixed Integer Linear Programming (MILP) and Branch and Bound (B&B) have been extensively studied, with multiple tailored models proposed for the classical FJSSP [9, 10]. Constraint Programming (CP) has also emerged as a competitive alternative, demonstrating strong performance on medium-sized instances and providing high-quality solutions for more complex variants such as distributed FJSSP and parallel batching [11, 12].

Heuristic approaches, including dispatching rules, have long been valued for their computational efficiency and simplicity, with early studies laying the groundwork for fast decision-making policies [13, 14]. More advanced, problem-specific heuristics have since improved solution quality by exploiting structural properties of the FJSSP [15, 16]. Metaheuristics further extend this landscape by offering greater generality and robustness. Trajectory-based methods, such as Simulated Annealing, Tabu Search, and Variable Neighbourhood Search, as well as population-based methods like Genetic Algorithms [17] and discrete Particle Swarm Optimization variants [18], are widely applied, particularly in highly flexible settings where the search space grows rapidly. Hybrid metaheuristics that combine intensification and diversification mechanisms have proven especially effective at balancing exploration and exploitation.

Despite their success, these classical approaches present well-known limitations that motivate new research directions. Exact methods provide optimality guarantees but quickly become computationally intractable for large or highly constrained instances, often failing to return feasible solutions under limited time budgets [19]. Heuristics are fast but typically lack robustness and generality. In contrast, metaheuristics require substantial expert knowledge for algorithm design and parameter tuning and are often developed through empirical trial-and-error [19]. These characteristics make them strong baselines for benchmarking, but also highlight the need for alternative paradigms that reduce modeling effort while maintaining competitive performance in multi-objective settings.

2.2 Language Models in Optimization and Decision-Making

Recent advances in Large Language Models (LLMs) have introduced a fundamentally different approach to optimization and decision-making. Rather than relying on manually crafted algorithms, LLMs leverage pre-trained knowledge and reasoning capabilities to automate strategy generation and adaptation. Early integration efforts have focused on combining LLMs with reinforcement learning, where language-based representations are used to support sequential decision-making, yielding measurable improvements in application domains such as agriculture [20, 21].

LLMs have also been adopted as orchestration agents in automated machine learning pipelines. Systems such as AutoML-GPT interpret high-level user objectives, generate executable workflows, and tune models and hyperparameters with minimal

human intervention [22]. Extensions of this idea treat source code itself as part of the search space, enabling large-scale exploration of algorithmic designs [23]. Agent-based frameworks such as AgentHPO further reduce expert involvement by delegating configuration proposal and validation to LLM-guided agents [24].

From a methodological perspective, advances in prompting and reasoning have enhanced the reliability of LLM-generated outputs. Chain-of-Thought prompting enables structured decomposition of complex tasks [25], while self-referential prompt evolution frameworks such as PromptBreeder demonstrate how LLMs can iteratively improve their own instructions [26]. These developments provide the foundation for using LLMs not only as assistants but also as active components within optimization pipelines.

2.3 LLMs for Heuristic Generation and Evolutionary Benchmarking

Beyond supporting existing workflows, LLMs have been increasingly explored as generators and evolvers of heuristics and algorithms. Several frameworks position LLMs as synthesizers of new metaheuristics by recombining principles from established methods into executable solvers [5, 6]. Iterative refinement pipelines have even led to the creation of entirely novel algorithms through repeated LLM-guided evaluation and revision [27].

Evolutionary perspectives further extend this idea by embedding LLMs within performance-driven feedback loops. Frameworks such as AEL, LLaMEA, and LLMs-GP iteratively mutate and refine heuristics or programs based on empirical results, demonstrating improvements on classical combinatorial optimization problems [28–30]. Other approaches integrate LLMs directly as evolutionary operators, guiding exploration, exploitation, or diversity control within population-based search processes [31–35]. These paradigms naturally extend to multi-objective optimization, where LLMs help decompose and address competing objectives without extensive expert tuning [36].

In scheduling and related combinatorial domains, LLMs have shown particular promise in the automated generation of dispatching rules. Some frameworks combine LLM-generated policies with evolutionary refinement to improve robustness and adaptability in dynamic environments [37, 38]. Low-intervention heuristic generation frameworks and their extensions introduce reflective feedback mechanisms that allow LLMs to revise heuristics based on past performance [39, 40].

Within this landscape, REMoH builds on existing evolutionary-LLM paradigms while placing explicit emphasis on systematic benchmarking. Its reflective component is designed to guide heuristic refinement through structured feedback. Still, its primary contribution is enabling direct, multi-objective comparisons among LLM-generated heuristics, exact optimization models, and classical heuristic baselines under consistent evaluation criteria. This positioning directly addresses the current lack of rigorous empirical evidence on the effectiveness and limitations of LLM-driven heuristics in multi-objective scheduling.

3 Methodology

This section presents the REMoH framework, a hybrid optimization methodology. It embeds LLM-generated heuristics within a classical multi-objective evolutionary algorithm. The methodological focus is on enabling controlled, systematic benchmarking of LLM-driven heuristics against exact and heuristic baselines. This is done using a Pareto-based evaluation scheme. REMoH is built on NSGA-II, which provides a well-established selection and diversity-preservation mechanism for multi-objective optimization.

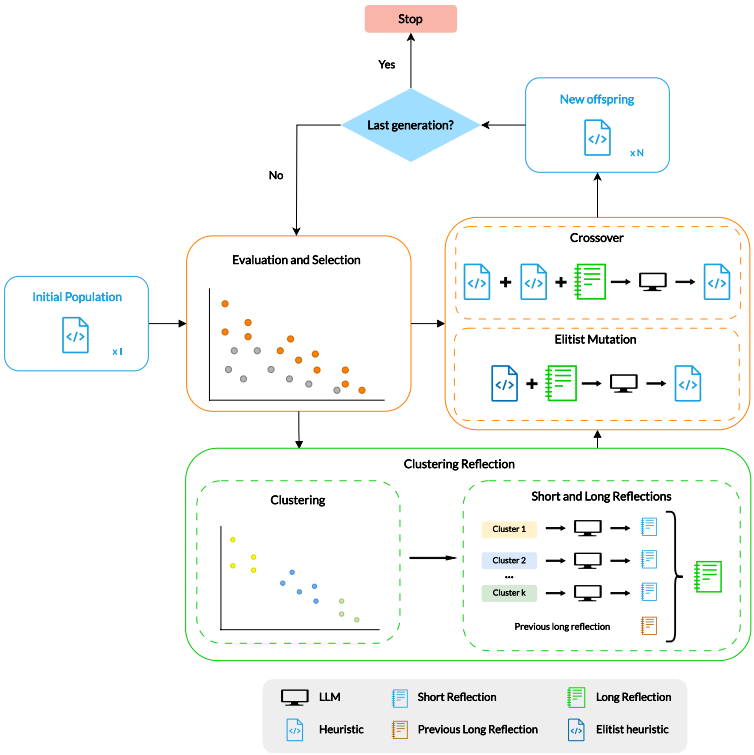


Fig. 1: Detailed diagram of our proposed Multi-Objective Evolution of Heuristics-based optimization methodology.

Figure 1 outlines the overall workflow. The process follows a standard evolutionary loop: population initialization, multi-objective heuristic evaluation, Pareto-based selection, and offspring generation. Additionally, a lightweight reflective component is included to guide heuristic variation, although it remains auxiliary to the evolutionary benchmarking pipeline.

Heuristic representation. Each individual in the population is represented as an executable heuristic. This heuristic is encoded as source code generated by an LLM. The representation follows principles from LLM-based evolutionary program search frameworks, such as EoH and ReEvo [39, 41]. This approach avoids predefined heuristic templates. It enables direct comparison with handcrafted heuristics and algorithmic baselines by evaluating all methods within the same problem environment.

Population initialization. The initial population is generated by prompting the LLM with a formal task specification. This specification describes the optimization problem, including input-output interfaces and objective functions. A seed heuristic can be provided to anchor the search. REMoH promotes diversity in the initial generation by varying role-based prompt instructions, inspired by HSEvo [40]. In keeping with standard evolutionary practice, REMoH initializes the first generation with a larger population to enhance coverage of the search space.

Evaluation and fitness assignment. Each heuristic is evaluated on a set of problem instances. This produces objective-wise performance values. These values are normalized using a standard scaling procedure and aggregated for selection purposes. Individual objective scores are preserved for Pareto dominance assessment. Infeasible heuristics or those with execution errors are discarded and regenerated to maintain population continuity. This evaluation protocol ensures LLM-generated heuristics are assessed under the same conditions as classical methods. It enables fair benchmarking.

Multi-objective selection. Selection is performed using NSGA-II. Individuals are ranked according to Pareto dominance. Crowding distance is used to preserve diversity within non-dominated fronts. This mechanism enables REMoH to compare heterogeneous optimization paradigms, LLM-generated heuristics, classical heuristics, and exact methods, using consistent multi-objective criteria rather than scalarized metrics.

Guided variation via reflection. To support informed heuristic variation, REMoH incorporates a compact reflective mechanism. Parent heuristics are clustered based on objective performance using K-Means Clustering. The number of clusters is determined via the silhouette method [42]. The LLM generates summaries of each cluster, highlighting dominant performance patterns and structural characteristics. These summaries are consolidated into a brief reflection that informs the next generation. This step guides exploration without altering the core evolutionary dynamics.

Crossover and mutation. Offspring heuristics are generated via LLM-guided crossover. This process combines elements from selected parents while taking into account the reflective summary. An elitist mutation operator also modifies the best-performing heuristic observed so far. This is defined as the individual who achieves the most balanced trade-off among objectives. Reflection is used only as a guiding signal to ensure that variation remains structured rather than purely random [40]. Overall, REMoH maintains a conventional evolutionary optimization structure centered on NSGA-II. LLMs act as heuristic generators and variation operators. This design enables direct and transparent benchmarking of language-driven heuristics against exact models and classical heuristics in a multi-objective setting. The reflective mechanism remains secondary to the evaluation-driven evolutionary process.

4 Experimental Setup

This section describes the experimental setup used to benchmark LLM-driven heuristics to exact and heuristic optimization methods in a multi-objective scheduling context. The evaluation focuses on solution quality, scalability, and generalization over diverse problem instances. All experiments use the Flexible Job Shop Scheduling Problem (FJSSP), a benchmark in scheduling research. FJSSP captures both combinatorial complexity and practical relevance [43]. As a generalization of the classic Job Shop Scheduling Problem, FJSSP allows each operation to be processed by one of several eligible machines, substantially increasing the size of the decision space and the difficulty of optimization [44, 45]. These characteristics make FJSSP particularly suitable for evaluating the effectiveness of LLM-generated heuristics against classical approaches [7].

The optimization task assigns operations to machines and sequences them to optimize multiple, possibly conflicting objectives. In line with prior work, this research focuses on two goals. First, makespan: the completion time of the last job to finish. Second, workload balance: measured as the maximum workload on any machine. While makespan is usually the dominant goal, including workload balance allows meaningful assessment of multi-objective performance and trade-off handling [43].

4.1 Evaluation and Selection Criteria

All candidate heuristics are evaluated using the complete set of FJSSP instances. Objective values for each example are normalized at the population level. This approach enables fair comparison of different heuristics. Mean objectives are computed for each individual at the instance level, then aggregated across instances and generations. This procedure applies equally to all benchmarking methods.

To assess solution quality in the multi-objective space, two standard indicators are used: Hypervolume (HV) and Inverted Generational Distance (IGD). Since true Pareto fronts are unavailable for these FJSSP benchmarks, the reference set is constructed as the non-dominated union of all solutions obtained from each method. HV and IGD, calculated relative to this reference, serve to directly compare the performance of LLM-driven heuristics, exact models, and all other baseline methods included in the study. For comparability, all metrics are averaged across three independent runs. Detailed analyses of auxiliary mechanisms are provided in the Appendix C.

4.2 Problem Instances

The experimental evaluation relies on three widely used and publicly available FJSSP benchmark suites:

- Brandimarte [46]: A set of 15 instances (MK01–MK15) extensively used in the literature and adaptable to multi-objective evaluation.
- Barnes [47]: An early FJSSP benchmark introduced together with a tabu search method, still relevant for comparative studies.
- Dauzere [7]: A collection of 18 instances with varying sizes and flexibility levels, designed to assess scalability and robustness.

The Brandimarte cases are the primary reference due to their widespread use and detailed performance testing. In the main tests, heuristics are evaluated across all datasets to assess their performance across different problems. Further details for individual cases are given separately as needed.

4.3 Benchmarking Methods

The approach is compared against classical and learning-based optimization methods. The benchmark includes three classical methods based on exact or heuristic formulations. It also features two recent learning-based methods based on Reinforcement Learning. This selection covers the current range of scheduling solvers, from mathematical to data-driven approaches.

To ensure comparability, a lexicographic optimization strategy is employed for solvers that lack native multi-objective support. For each method, two solutions are generated: one prioritizes makespan, with workload balance second; the other reverses these priorities. These solutions provide a consistent trade-off space, allowing direct comparison with Pareto-based methods. This benchmarking protocol enables systematic assessment of LLM-generated heuristics against established optimization techniques under identical criteria.

4.3.1 Mixed Integer-Linear Programming

This section presents the MILP formulation for the FJSSP, based on the model presented by Dauzere et al. [7]. The formulation captures the essential constraints of FJSSP, ensuring a feasible and optimal assignment of operations to machines while minimizing both objectives. The following describes the notations used, sets, parameters, variables, and the formulation.

Table 1: Summary of Sets, Parameters, and Variables

Sets	
O	Set of operations
$pr(i)$	Predecessor of operation i
R	Set of machines
R_i	Subset of machines R capable of performing operation i
Parameters	
$p_{k,i}$	Processing time of operation i on machine k
H	A sufficiently large number (big-M constant)
C	Fixed makespan
RM	Fixed maximum machine operative time
Variables	
$\alpha_{k,i}$	Binary variable: 1 if operation i is assigned to machine k
$\beta_{i,i'}$	Binary variable enforcing sequencing between operations on the same machine
t_i	Start time of operation i
c_{max}	Resulting makespan
r_{max}	Maximum machine operative time (used for workload balance)

$$\min \ c_{max} \tag{1}$$

$$\sum_{k \in R_i} \alpha_{k,i} = 1 \qquad \forall \ i \in O \tag{2}$$

$$t_i \geq t_{pr(i)} + \sum_{k \in R_{pr(i)}} p_{k,pr(i)} \cdot \alpha_{k,pr(i)} \qquad \forall \ i \in O \tag{3}$$

$$t_i \geq t_{i'} + p_{k,i'} - (2 - \alpha_{k,i} - \alpha_{k,i'} + \beta_{i,i'}) \cdot H \qquad \forall \ (i,i') \in O \times O, \ s.t \ i \neq i', \ \forall \ k \in R_i \cap R_{i'} \tag{4}$$

$$t_{i'} \geq t_i + p_{k,i} - (3 - \alpha_{k,i} - \alpha_{k,i'} + \beta_{i,i'}) \cdot H \qquad \forall \ (i,i') \in O \times O, \ s.t \ i \neq i', \ \forall \ k \in R_i \cap R_{i'} \tag{5}$$

$$c_{max} \geq t_i + \sum_{k \in R_i} p_{k,i} \cdot \alpha_{k,i} \qquad \forall \ i \in O \tag{6}$$

$$\alpha_{k,i} \in \{0,1\} \qquad \forall \ i \in O, \ k \in R_i \tag{7}$$

$$\beta_{i,i'} \in \{0,1\} \qquad \forall \ (i,i') \in O \times O \tag{8}$$

In this formulation, the objective is to minimize the makespan, as indicated in Equation 1. Constraint 2 ensures that each operation is assigned to exactly one machine. Constraint 3 enforces the precedence relationships between consecutive operations. Constraints 4 and 5 prevent overlapping of operations scheduled on the same machine. Constraint 6 defines the makespan, while Constraints 7 and 8 specify the domains of the decision variables.

$$\min \ r_{max} \tag{9}$$

$$c_{max} \leq C \tag{10}$$

$$r_{max} \geq \sum_{i \in O, \ s.t \ k \in R_i} p_{k,i} \cdot \alpha_{k,i} \qquad \forall \ k \in R \tag{11}$$

The following objective is to minimize workload balance, as expressed in Equation 9. To implement this approach, the makespan variable must be constrained so that it does not exceed the previously obtained value (Constraint 10), and the maximum machine productive time must also be appropriately bounded (Constraint 11). The remaining constraints from the previous model remain unchanged.

$$r_{max} \leq RM \tag{12}$$

When workload balance is prioritized as the primary objective, the objective function in Equation 1 is replaced with that in Equation 9 in the proposed model. Additionally, the constraint defining the makespan (Constraint 6) is substituted by the one describing the maximum machine productive time (Constraint 11). In this case, during the second phase of the lexicographic approach, the objective is to minimize the makespan (Equation 1). Instead of using Constraint 10 to limit the makespan, Constraint 12 is introduced to impose an upper bound on each machine's usage.

4.3.2 Constraint Programming Approach

Constraint Programming (CP) is a paradigm for solving combinatorial optimization problems [48], particularly effective in complex scheduling contexts such as the FJSSP [49]. The issue is subject to two primary constraints (see Algorithm 1). On the one hand, precedence constraints enforce the prescribed order of operations within each job. On the other hand, resource constraints prevent overlapping operations on the same machine. In the proposed pseudocode, $C_{\max}$ denotes the makespan and $L_{\max}$ denotes the maximum workload across machines.

For benchmarking, a CP model is implemented using two state-of-the-art solvers: (i) Google OR-Tools CP-SAT [50], a robust open-source constraint programming engine known for its efficiency in solving scheduling problems, and (ii) IBM CPLEX CP Optimizer (DoCplex) [51], a commercial optimization suite that offers advanced support for scheduling via interval and sequence variables.

Algorithm 1 Constraint Programming Model for FJSSP with Lexicographic Objectives

Require: Set of jobs J, operations O, machines M, processing times p_{jim}
Ensure: the Schedule minimizes the makespan and workload balance

Define interval variables iv_{jim} for all $o_{ji} \in O$ and $m \in M_{ji}$
Define optional intervals with durations p_{jim}
Define decision variables $start_{ji}, end_{ji}$ for each operation o_{ji}
for all $o_{ji} \in O$ **do**
    Add Alternative(o_{ji}, $\{iv_{jim} \mid m \in M_{ji}\}$)
end for
for all $j \in J$ **do**
    for $i = 1$ **to** $|J_j| - 1$ **do**
        Add precedence constraint: EndBeforeStart($o_{ji}, o_{j,i+1}$)
    end for
end for
for all $m \in M$ **do**
    Add NoOverlap constraint on iv_{jim} scheduled on machine m
    Define $load_m \leftarrow \sum_{ji}$ PresenceOf(iv_{jim}) $\cdot\, p_{jim}$
end for
Define makespan $C_{\max} = \max_{ji}$ EndOf(o_{ji})
Define max machine productive time $L_{\max} = \max_{m \in M} load_m$

4.3.3 Baseline dispatching rule

The third classical method evaluated is a dispatching-rule-based heuristic that sequentially schedules operations according to predefined job priorities. Instead of solving the FJSSP globally, it iteratively selects jobs and assigns operations to eligible machines based on the earliest finish time.

As shown in Algorithm 2, jobs are initially ordered randomly to encourage exploration. Each operation is then scheduled on the machine that yields the earliest feasible finish time, subject to precedence constraints and machine availability. The machine schedule is updated iteratively until all operations are assigned, using a greedy strategy that prioritizes minimizing makespan. In this pseudocode, bt stands for best time, bm best machine, and bd best duration.

Algorithm 2 Baseline dispatching rule

Input Set of jobs J, operations O_j, machines M_o, processing times p_{om}
Output Sol
$J_p \leftarrow prioritization(J)$
$Sol \leftarrow \emptyset$
for all $O_j \in J_p$ **do**
    $tc \leftarrow 0$
    for all $o \in O_j$ **do**
        $bt \leftarrow \infty$
        for all $m \in M_o$ **do**
            $t \leftarrow earliesFeasibletTime(m, p_{om}, tc, Sol)$
            if $t < bt$ **then**
                $bt \leftarrow t$
                $bm \leftarrow m$
                $bd \leftarrow p_{om}$
            end if
        end for
        $Sol \leftarrow Sol \cup (bt, bm, bd)$
        $tc \leftarrow bt + bd$
    end for
end for

4.3.4 Advances in AI techniques

The benchmark includes two state-of-the-art RL approaches tested on the Brandimarte dataset [46]: (1) Ho et al. [52], who proposed a GNN-based RL method that filters completed jobs and occupied machines to reduce decision space, and (2) Echeverria et al. [53], who combined RL with CP by training on optimal CP-generated solutions and embedding CP within the RL loop for refinement. Given their strong reported performance, both methods are included as RL baselines. Performance metrics are taken directly from the original publications, as implementations are not publicly available.

5 Results and Discussion

This section reports and analyzes experimental results for the Flexible Job Shop Scheduling Problem, with a primary focus on benchmarking LLM-generated heuristics against exact mathematical models, classical heuristics, and learning-based approaches in a multi-objective setting. The analysis emphasizes solution quality, trade-off consistency, and computational efficiency. Additional studies related to auxiliary mechanisms are reported in the appendix C and are not central to the discussion.

Preliminary experiments were conducted to select the best-performing configuration of the proposed framework. To ensure clarity and avoid confounding effects, only this configuration is considered in the benchmarking results presented below.

All experiments were executed on a standard personal computing environment (8-core Intel i7 CPU, 16 GB RAM, no dedicated GPU). This setup highlights the practicality of the approach. It also allows a fair comparison with classical solvers operating under comparable computational constraints. All LLM-based heuristics were generated and executed via publicly available APIs, reinforcing the accessibility of the proposed methodology.

5.1 Brandimarte Benchmark

The proposed approach is first tested on the Brandimarte benchmark for FJSSP. To compare multi-objective and single-objective solvers, mathematical models use a lexicographic optimization strategy. Each solver has two setups: one focuses on makespan (mb) and the other on workload balance (bm). This yields six baselines: $MILP_{mb}$, $MILP_{bm}$, OR_{mb}, OR_{bm}, DC_{mb}, and DC_{bm}, where OR and DC refer to constraint programming with OR-Tools and DoCplex, respectively.

Because FJSSP is complex, all exact solvers have time limits. The MILP model is solved using Gurobi Optimizer 11.0.1 for up to 30 minutes per objective; CP solvers are limited to 5 minutes. As a simple baseline, a greedy dispatching rule (DR) is applied 10 times per case, retaining the best result for each goal.

Two recent RL-based methods, proposed by Ho et al. ?citeho2024residual (RL_1) and Echeverria et al. [53] (RL_2), are also considered. However, both methods report results only for single-objective makespan minimization and only for instances Mk01–Mk10. They are excluded from the multi-objective benchmarking and are used solely as a single-objective reference.

After selecting models as described in Appendix C, the REMoH framework generates several top heuristics with Gemini 2.0 Flash. For benchmarking, the best heuristic for each case, based on makespan and workload balance, is selected. These values are in Tables 2 and 3.

For each instance, objective values are reported together with the best-known lower bounds (LBs). These LBs are defined as the highest LB obtained across all MILP and CP configurations and prior studies [52–54]. Performance is summarized using the mean GAP across all instances, where GAP denotes the relative percentage deviation from the corresponding LB.

The results compellingly demonstrate that REMoH delivers strong performance in makespan minimization, achieving a mean GAP of 12.60%. This performance closely

Table 2: Benchmarking of Makespan optimization on Brandimarte Instances, comparing mathematical models, Ho et al. [52] (RL_1), Echeverria et al.[53] (RL_2), and three heuristics obtained by REMoH

	$MILP_{mb}$	DC_{mb}	OR_{mb}	DR	RL_1	RL_2	REMoH	LB
Mk01	**40**	**40**	**40**	64	42	**40**	41	40
Mk02	**26**	**26**	**26**	41	29	29	32	26
Mk03	**204**	**204**	**204**	329	**204**	**204**	**204**	204
Mk04	61	**60**	**60**	117	67	64	72	60
Mk05	180	173	**172**	240	177	176	182	172
Mk06	66	59	**58**	91	71	73	73	57
Mk07	151	**139**	**139**	192	149	156	161	139
Mk08	**523**	**523**	**523**	661	**523**	**523**	**523**	523
Mk09	311	**307**	**307**	438	314	**307**	323	307
Mk10	255	**202**	211	388	218	236	225	189
Mk11	629	612	**609**	757	-	-	658	609
Mk12	**508**	**508**	**508**	674	-	-	534	508
Mk13	433	406	**403**	586	-	-	471	369
Mk14	702	**694**	**694**	1072	-	-	707	694
Mk15	360	**333**	344	552	-	-	422	333
GAP	6.46	**1.43**	1.73	54.78	8.05	8.57	12.60	

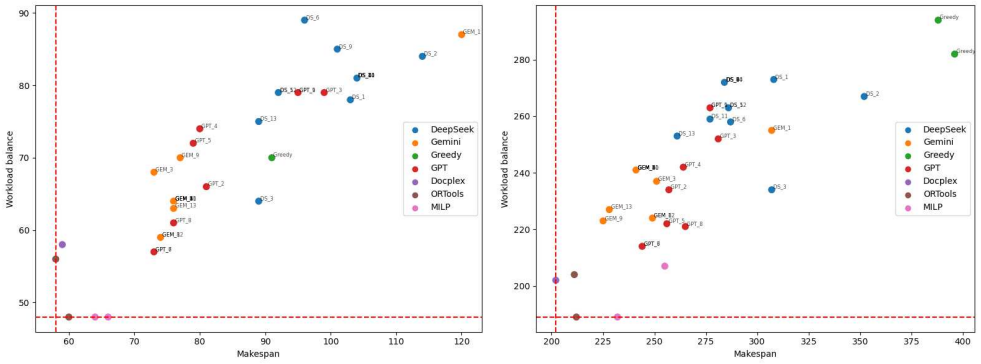
rivals that of RL-based approaches on these instances while substantially outperforming the greedy dispatching rule. Moreover, unlike RL methods, which are restricted to makespan and cannot scale beyond Mk10, REMoH simultaneously optimizes both objectives and covers all instances, clearly establishing its broader applicability and advantages.

As expected, lexicographic mathematical models achieve strong performance on their prioritized objective, with DC_{mb} achieving a mean GAP of 1.43% in makespan. However, these models provide limited clarity regarding trade-offs among objectives, rendering their solutions less informative for multi-objective problems. In sharp contrast, REMoH-generated heuristics consistently yield balanced solutions, demonstrating a holistic quality metric across objectives. Consequently, while objective-specific solvers may occasionally outperform REMoH on individual metrics, REMoH reliably delivers stable, high-quality outcomes across both objectives, underscoring its robust multi-objective performance.

To further analyze solution quality, the normalized Euclidean distance is computed between each heuristic-generated solution and the best-known reference solution, typically from exact solvers. Before distance computation, objective values are normalized per instance. As shown in Figure 3, heuristics from Gemini are consistently closest to the reference front. GPT and DeepSeek heuristics follow, while the greedy DR is further behind. This highlights Gemini-based heuristics as outperforming other approaches in generalizing to unseen instances, despite comparable training performance across some models.

Table 3: Benchmarking of Workload Balance on Brandimarte Instances comparing mathematical models, and three heuristics obtained by REMoH

	$MILP_{bm}$	DC_{bm}	OR_{bm}	DR	REMoH	LB
Mk01	**36**	**36**	**36**	55	37	36
Mk02	**26**	**26**	**26**	31	31	26
Mk03	**204**	**204**	**204**	274	**204**	204
Mk04	**60**	**60**	**60**	103	67	60
Mk05	**172**	**172**	**172**	194	180	172
Mk06	**48**	**48**	**48**	70	68	48
Mk07	**139**	**139**	**139**	158	161	139
Mk08	**523**	**523**	**523**	542	**523**	523
Mk09	**299**	**299**	**299**	347	307	299
Mk10	**189**	**189**	**189**	282	223	188
Mk11	**609**	**609**	**609**	652	658	609
Mk12	**508**	**508**	**508**	572	534	508
Mk13	**382**	**382**	**382**	477	470	382
Mk14	**694**	**694**	**694**	903	707	694
Mk15	**332**	**332**	**332**	406	422	332
GAP	**0.04**	**0.04**	**0.04**	27.79	12.15	



(a) Mk6 instance: 10 jobs, 10 machines.

(b) Mk10 instance: 20 jobs, 15 machines.

Fig. 2: Illustrative results on Brandimarte instances. The table compares outcomes from MILP, DoCplex, OR-Tools, and heuristics derived from our LLM-integrated model, which incorporates DeepSeek (DS), Gemini 2.0 Flash (GEM), and GPT-4o. (GPT)

Computational efficiency is also evaluated. Average runtimes for exact solvers are 1661.5 s for MILP, 230.9 s for CP with DoCplex, and 148.4 s for CP with OR-Tools. These reflect both solver characteristics and imposed time limits. In line with recent findings [7], CP approaches consistently outperform MILP in runtime and solution quality. The greedy DR is the fastest baseline (0.002 s). REMoH heuristics show

runtimes of 0.001 s (GPT), 0.012 s (DeepSeek), and 0.167 s (Gemini). This highlights a trade-off between solution quality and execution time among LLM-generated heuristics.

Figure 2 illustrates these dynamics for instances Mk06 and Mk10. Simple heuristics remain competitive on smaller instances. REMoH-generated heuristics become increasingly effective as problem complexity grows. MILP performance degrades with increasing instance size, whereas CP approaches remain robust.

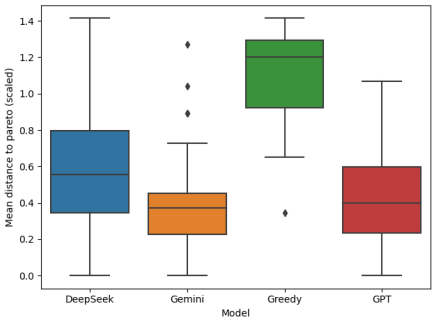


Fig. 3: Boxplot of the mean Euclidean distances to best-known solutions for different approaches.

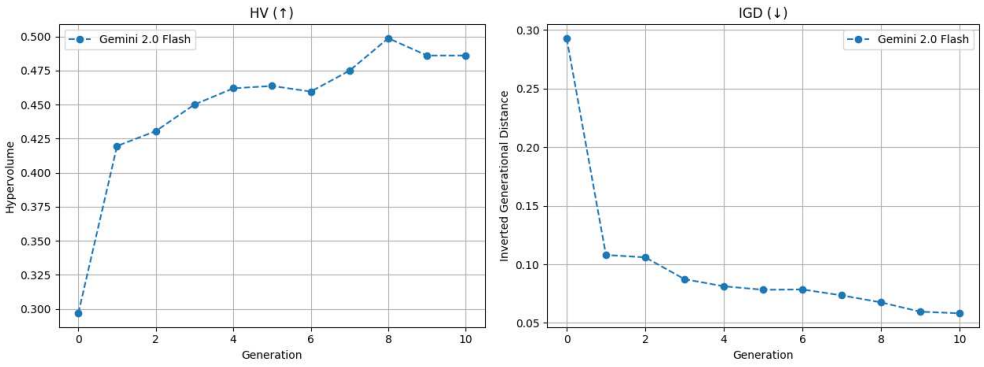
In summary, the findings demonstrate that REMoH delivers stable, competitive performance across all benchmark instances, frequently approaching or matching the best-known solutions produced by exact methods. These robust results establish LLM-driven heuristic generation as a practical, scalable alternative for multi-objective scheduling, especially where flexibility, interpretability, and balanced trade-offs are crucial. REMoH thus emerges as a valuable tool for practitioners seeking both quality and adaptability in real-world scenarios.

5.2 Flexibility Evaluation

This subsection evaluates the flexibility of the proposed approach for nonstandard FJSSP extensions. The goal is to assess how well LLM-generated heuristics adapt to increased modeling complexity relative to exact and learning-based optimization methods.

Building on this, the baseline FJSSP is extended by incorporating *Sequence-Dependent Setup Times* (SDST), a well-known generalization that increases realism in scheduling problems [55]. In SDST settings, the setup time required before processing an operation depends on both the current operation and the one previously executed on the same machine [56]. To further increase modeling difficulty, setup times are defined as a nonlinear function of machine idle time, capturing practical effects such as increased startup costs after prolonged inactivity:

$$S(t) = \begin{cases} 0 & \text{if } t < 1 \\ 10 \cdot (1 - e^{-0.5(t-1)}) & \text{if } 1 \leq t < 10 \\ 20 & \text{if } t \geq 10 \end{cases}$$



(a) HyperVolume (HV) (b) Inverted Generational Distance (IGD)

Fig. 4: Heuristic evolution performance validation with non-linear Sequence-Dependent Setup Times (SDST).

This methodological shift highlights the limitations of traditional optimization paradigms. For example, MILP formulations cannot directly handle nonlinear expressions without approximation, often at the expense of solution quality. While constraint programming offers greater modeling flexibility, incorporating nonlinear behavior typically requires specialized propagators and tailored search strategies, which significantly increases implementation effort. Similarly, RL-based approaches are affected by problem reformulation: introducing new constraints necessitates redefining reward structures, regenerating training data, and retraining policies, which is computationally expensive and sensitive to design choices.

In contrast, the proposed LLM-driven framework accommodates this extension with minimal intervention. Specifically, the nonlinear constraint is introduced by augmenting the LLM's textual description and by updating the feasibility-checking procedure. Importantly, no structural changes to the optimization process or retraining phases are required. This illustrates a key advantage of language-based heuristic generation in dynamic environments.

The experiment is conducted on the Brandimarte dataset using the same evaluation protocol described in Section 4.1. Specifically, Figures 4a and 4b report the corresponding HV and IGD results. Together, these results show that REMoH maintains stable performance under the extended problem formulation, confirming its ability to adapt to nonlinear, context-dependent constraints.

Taken together, these results reinforce the practical relevance of LLM-generated heuristics in scenarios where problem specifications evolve. Compared with exact

solvers and RL-based methods, the proposed approach offers a more flexible, lightweight pathway for incorporating complex constraints, which is critical for real-world multi-objective scheduling applications.

6 Conclusions

This work demonstrates that LLMs can act as effective heuristic generators in multi-objective optimization, rigorously evaluating their performance against exact mathematical models, classical heuristics, and learning-based methods. By integrating LLM-generated heuristics into an evolutionary framework, the study systematically benchmarks language-driven optimization on complex scheduling problems, revealing that LLMs are competitive with established approaches.

Results on the FJSSP show that LLM-generated heuristics match or surpass classical and state-of-the-art methods when optimizing for multiple objectives simultaneously. While exact methods are strong for single-objective optimization, their scalability and modeling complexity limit real-world use. The LLM-driven approach consistently achieves balanced trade-offs between makespan and workload, highlighting its suitability for dynamic, multi-objective settings.

The benchmarking analysis shows that LLM-generated heuristics approach the performance of RL-based methods in makespan minimization. They also offer broader applicability. RL approaches are typically restricted to a single objective and limited instance sizes. Instead, the proposed framework generalizes across diverse problem instances and objectives without retraining or dataset-specific tuning. The low execution times of the evolved heuristics highlight their potential for real-time or near-real-time decision support. In such cases, exact solvers often become impractical.

The flexibility analysis shows a key advantage of language-based heuristic generation beyond solution quality. It can incorporate nonlinear and context-dependent constraints with minimal reformulation. This stands in contrast to the substantial redesign required by mathematical programming and RL-based approaches. This adaptability reinforces the relevance of LLM-driven optimization in realistic industrial environments, where problem specifications often change.

The framework includes a reflective component to guide heuristic evolution, but its role is complementary rather than central. The main contribution of this work is to demonstrate that LLMs can serve as effective, competitive heuristic generators. This is clear when they are rigorously benchmarked against established optimization paradigms using consistent multi-objective evaluation criteria.

In summary, this study establishes LLM-driven heuristic evolution as a robust, practical alternative for multi-objective scheduling. The approach reduces modeling effort and delivers competitive results, showing how LLMs can be systematically applied within optimization workflows. Future work will broaden this evaluation and automate higher-level strategy generation.

Appendix A Pseudocode of the Proposed Multi-Objective EoH Algorithm

This appendix provides the detailed pseudocode of the proposed Multi EoH algorithm. This hybrid framework integrates NSGA-based multi-objective evolutionary optimization with LLMs for heuristic generation. The pseudocode 3 encapsulates the entire optimization pipeline. This detailed algorithmic structure provides a transparent foundation for reproducibility and further research.

For each iteration $t \in 1, \ldots, \mathcal{T}$, the algorithm evolves a population of heuristics through LLM-guided generation, reflection, and selection mechanisms. The process begins with an initial population $\mathcal{P}_0$, either provided or generated from scratch using the LLM $\mathcal{L}$.

At each generation step, a subset of individuals is selected to form the parent population $P_{parents}$, which is then clustered into subgroups. These clusters serve as the basis for generating short reflections via `Reflection`$(\mathcal{C}, \mathcal{L})$, which summarize the observed strategic patterns. A more comprehensive reflection $\mathcal{R}_t$ is subsequently constructed by combining the short reflections with those from previous iterations.

Two types of offspring are generated:

Crossover-based heuristics, created by combining two selected parents (p_1, p_2) guided by $\mathcal{R}_t$ and the LLM, account for $r_c \cdot \mathcal{P}$ new individuals.

Mutation-based heuristics, derived from a single elite parent p_{best} using $\mathcal{R}_t$ and the LLM, account for $r_m \cdot \mathcal{P}$ individuals.

The new population P_t is formed by merging the parent and offspring populations. All individuals generated across iterations are collected in the global archive P^*.

At the end of the evolutionary process, the global Pareto front $\mathcal{F}$ is extracted from P^*, representing the most effective heuristics across multiple objectives.

Appendix B Ablation Study

This experimental technique, commonly used in algorithmic research, helps to quantify the contribution of the reflective component to the quality and stability of the heuristic search under varying computational constraints. Comparing the proposed model with a variant that excludes the reflection mechanism can demonstrate the effectiveness in enhancing solution diversity and guiding search processes more efficiently. This experiment aims to support the novelty of our approach and highlight the relevance of incorporating self-reflective strategies into the framework.

In this phase, two experimental configurations were defined to examine the consistency and scalability of the reflection mechanism under different conditions. The first configuration was initialized with a population size of 30 individuals, and subsequent generations were limited to 10 individuals; the algorithm was improved over 10 iterations. In the second configuration, the initial population was increased to 60 individuals, with a generation size of 20 and 20 iterations. All experiments were evaluated following the criteria mentioned in the subsection 4.1.

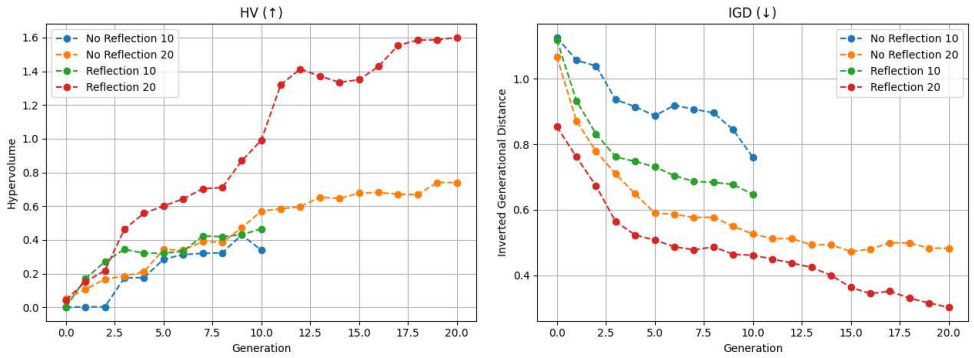
The experiments were carried out by running the algorithm on the Brandimarte dataset. As previously mentioned, this experiment involves conflicting objectives:

Algorithm 3 Proposed Multi-Evolution of Heuristics

Input: Initial population size $\mathcal{I}$, Population size $\mathcal{P}$, Iterations $\mathcal{T}$, Initial population $\mathcal{P}_\prime$, Pre-trained LLM $\mathcal{L}$, Crossover rate r_c, Mutation rate r_m
Output: Heuristics in global Pareto front $\mathcal{F}$
if $\mathcal{P}_0 = \emptyset$ **then**
  for $i = 1, ..., \mathcal{I}$ **do**
    $o \leftarrow Generation(\mathcal{L})$
    $P_0 \leftarrow P_0 \cup o$
  end for
end if
$P^* \leftarrow P_0$
for $t = 1, ..., \mathcal{T}$ **do**
  $P_{parents} \leftarrow Selection(P_{t-1})$
  $\mathcal{C} \leftarrow Clusterize(P_{parents})$
  $\mathcal{R}_{short} \leftarrow Reflection(\mathcal{C}, \mathcal{L})$
  $\mathcal{R}_t \leftarrow Reflection(\mathcal{C}, \mathcal{R}_{short}, \mathcal{R}_{t-1}, \mathcal{L})$
  $P_{sons} \leftarrow \emptyset$
  for $j = 1, ..., r_c * \mathcal{P}$ **do**
    $p_1, p_2 \leftarrow ParentSelection(P_{parents})$
    $o \leftarrow Crossover(p_1, p_2, \mathcal{R}_t, \mathcal{L})$
    $P_{sons} \leftarrow P_{sons} \cup o$
  end for
  for $j = 1, ..., r_m * \mathcal{P}$ **do**
    $p_{best} \leftarrow bestHeuristic(P_{parents})$
    $o \leftarrow Mutation(p_{best}, \mathcal{R}_t, \mathcal{L})$
    $P_{sons} \leftarrow P_{sons} \cup o$
  end for
  $P_t \leftarrow P_{parents} \cup P_{sons}$
  $P^* \leftarrow P^* \cup P_{sons}$
end for
$\mathcal{F} \leftarrow GlobalParetoFront(P^*)$

makespan and *workload balance*. Additionally, to test the clustering reflection's capability to handle challenging multi-objective problems, *operation separation* was introduced as a third objective. This objective measures the time distance between operations of the same job, a typical goal in the FJSSP to enhance scheduling efficiency.

The figures B1 present HV and IGD values across generations for both algorithm variants across the two experimental instances. Higher HV values indicate better solution convergence and diversity. In this sense, the evolution of HV in the figure B1a illustrates the improvement in solution diversity and convergence quality when using the reflection mechanism, as evidenced by consistently higher HV values in both instances. Conversely, lower IGD values in the figure B1b indicate better convergence toward the Pareto front. The figure on the right shows that the reflection-enhanced approach consistently achieves lower IGD, indicating improved convergence.



(a) HyperVolume (HV) (b) Inverted Generational Distance (IGD)

Fig. B1: Comparison between the reflective and non-reflective approaches using different population sizes and iterations.

The observed results provide empirical evidence that including the reflection mechanism significantly improves both convergence (HV) and proximity to the Pareto front (IGD). This component maintains diversity and prevents premature convergence by generating new individuals based on clustering and introspection of the search space. These findings support our hypothesis that integrating LLM-powered reflection enhances the robustness and adaptability of multi-objective optimization algorithms.

Appendix C LLM Selection

This appendix provides a comprehensive overview of the comparative evaluation conducted to select the most effective LLM in our Reflective Evolution of Multi-objective Heuristics (REMoH) framework.

This study aimed to identify the LLM that consistently produces high-quality heuristic operators and reflections, thereby maximizing the performance of the proposed multi-objective scheduling methodology. The tested LLMs include `GPT-4o` [57], `Gemini 2.0 Flash` [58], and `DeepSeek-V3` [59]. These models were selected for their competitive performance and suitability for practical optimization workflows.

The comparison was performed using training on two independent datasets from the FJSSP benchmark literature (Dauzere and Barnes) and validation on the Brandimarte dataset. The experiments focused on identifying which LLM most effectively generates high-quality Pareto fronts under these diverse conditions. The evaluation and selection criteria for this experimentation are explained in subsection IV-A. In this sense, the evaluation is performed in the same environment, with identical configurations across multiple runs.

The optimization process was initialized with a population size of 60 individuals. Then, throughout the evolutionary process, a smaller working population of 20 individuals was maintained for each generation. Each run consisted of 20 iterations of the algorithm. During the experiments, heuristics were evaluated for up to 100 seconds.

Regarding the evolutionary operators, crossover and mutation rates were fixed at 0.9 and 0.1, respectively.

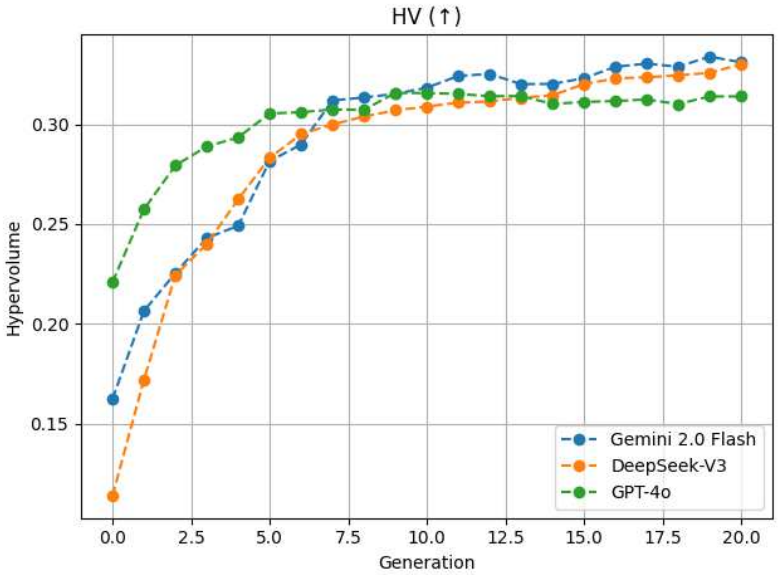


Fig. C2: Comparison between different LLMs in HV.

As shown in Figures C2 and C3, all models exhibit a significant improvement during the initial generations. After this early boost, `DeepSeek-V3` and `Gemini 2.0 Flash` follow a similar progression, gradually improving their performance over successive generations. In contrast, `GPT-4o` achieves strong results in the early stages but converges prematurely, making it increasingly difficult to improve in later iterations. As a result, although the final differences are relatively small, `GPT-4o` ends up with the lowest performance among the three models.

The Pareto front heuristics generated by each model have been evaluated on the Brandimarte datasets, with results for all instances presented in Tables C1, C2 and C3. Based on this evaluation, `Gemini 2.0 Flash` was selected as the reference model for comparison against established benchmarking methods, due to its superior performance. The results of this comparison are presented in Section V-B.

References

[1] Huang, S., Yang, K., Qi, S., Wang, R.: When large language model meets optimization. Swarm and Evolutionary Computation **90**, 101663 (2024) https://doi.org/10.1016/j.swevo.2024.101663

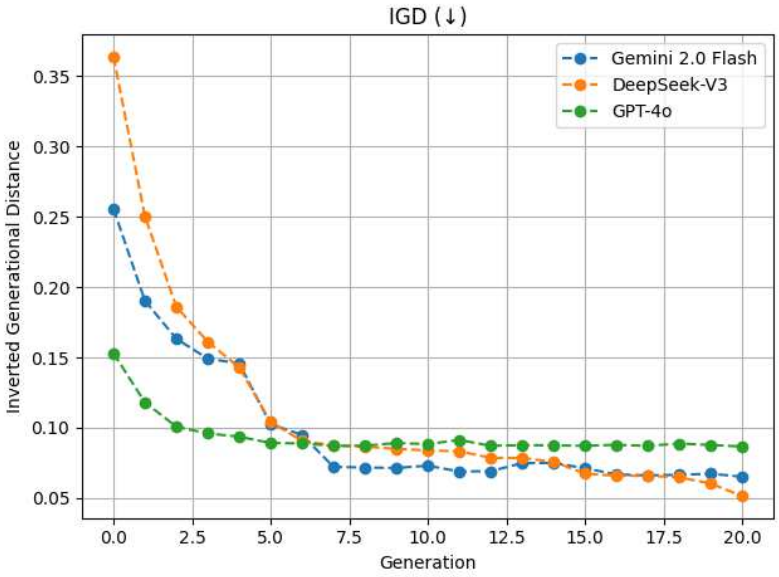


Fig. C3: Comparison between different LLMs in IGD.

[2] Gad, A.G.: Particle swarm optimization algorithm and its applications: a systematic review. Archives of computational methods in engineering **29**(5), 2531–2561 (2022)

[3] Tang, J., Liu, G., Pan, Q.: A review on representative swarm intelligence algorithms for solving optimization problems: Applications and trends. IEEE/CAA Journal of Automatica Sinica **8**(10), 1627–1643 (2021)

[4] Wu, X., Wu, S.-h., Wu, J., Feng, L., Tan, K.C.: Evolutionary Computation in the Era of Large Language Model: Survey and Roadmap (2024). https://arxiv.org/abs/2401.10034

[5] Pluhacek, M., Kazikova, A., Kadavy, T., Viktorin, A., Senkerik, R.: Leveraging large language models for the generation of novel metaheuristic optimization algorithms. In: Proceedings of the Companion Conference on Genetic and Evolutionary Computation. GECCO '23 Companion, pp. 1812–1820. Association for Computing Machinery, New York, NY, USA (2023). https://doi.org/10.1145/3583133.3596401 . https://doi.org/10.1145/3583133.3596401

[6] Zhong, R., Hussien, A.G., Yu, J., Munetomo, M.: Llmoa: A novel large language model assisted hyper-heuristic optimization algorithm. Advanced Engineering Informatics **64**, 103042 (2025) https://doi.org/10.1016/j.aei.2024.103042

[7] Dauzère-Pérès, S., Ding, J., Shen, L., Tamssaouet, K.: The flexible job shop scheduling problem: A review. European Journal of Operational Research **314**(2),

Table C1: Results of the Pareto front heuristics generated by the REMoH framework on the Brandimarte Instances, via DeepSeek-V3 [59].

		mk01	mk02	mk03	mk04	mk05	mk06	mk07	mk08	mk09	mk10	mk11	mk12	mk13	mk14	mk15
	Makespan	52	41	236	83	194	103	264	575	411	308	735	612	577	808	480
1	Balance	43	37	216	67	194	78	234	523	312	273	684	563	577	781	397
	Time (s.)	0.0010	0.0020	0.0050	0.0020	0.0040	0.0048	0.0041	0.0071	0.0083	0.0091	0.0102	0.0099	0.0124	0.0135	0.0142
	Makespan	59	48	251	107	208	114	292	605	421	352	773	650	672	996	480
2	Balance	49	40	216	83	196	84	255	523	327	267	730	565	652	801	445
	Time (s.)	0.0015	0.0020	0.0057	0.0039	0.0050	0.0040	0.0040	0.0075	0.0088	0.0115	0.0080	0.0095	0.0136	0.0123	0.0189
	Makespan	45	40	236	75	223	89	200	629	423	307	677	613	529	828	480
3	Balance	43	33	204	66	178	64	191	533	307	234	644	524	499	746	354
	Time (s.)	0.0031	0.0026	0.0155	0.0061	0.0091	0.0115	0.0085	0.0344	0.0374	0.0468	0.0338	0.0411	0.0521	0.0855	0.0836
	Makespan	48	40	227	73	189	104	215	523	337	284	663	537	489	694	468
4	Balance	43	39	213	68	189	81	215	523	322	272	663	537	489	694	468
	Time (s.)	0.0010	0.0020	0.0057	0.0030	0.0040	0.0056	0.0048	0.0099	0.0109	0.0158	0.0117	0.0130	0.0238	0.0150	0.0230
	Makespan	48	38	230	69	192	92	200	523	327	286	681	550	487	694	428
5	Balance	42	37	227	68	192	79	200	523	307	263	681	550	487	694	428
	Time (s.)	0.0009	0.0020	0.0050	0.0020	0.0055	0.0050	0.0045	0.0092	0.0099	0.0136	0.0109	0.0141	0.0216	0.0195	0.0223
	Makespan	48	38	241	69	192	96	200	523	322	287	681	550	487	694	428
6	Balance	42	37	204	68	192	89	200	523	307	258	681	550	487	694	428
	Time (s.)	0.0021	0.0025	0.0060	0.0039	0.0040	0.0051	0.0052	0.0091	0.0132	0.0151	0.0122	0.0139	0.0207	0.0221	0.0246
	Makespan	48	40	227	73	189	104	215	523	337	284	663	537	489	694	468
7	Balance	43	39	213	68	189	81	215	523	322	272	663	537	489	694	468
	Time (s.)	0.0020	0.0017	0.0085	0.0031	0.0040	0.0051	0.0057	0.0101	0.0115	0.0155	0.0137	0.0115	0.0248	0.0197	0.0252
	Makespan	48	40	227	73	189	104	215	523	337	284	663	537	489	694	468
8	Balance	43	39	213	68	189	81	215	523	322	272	663	537	489	694	468
	Time (s.)	0.0010	0.0020	0.0076	0.0042	0.0047	0.0045	0.0056	0.0080	0.0138	0.0142	0.0129	0.0110	0.0250	0.0376	0.0236
	Makespan	49	37	223	73	189	101	227	523	328	284	663	537	506	694	448
9	Balance	43	37	213	68	189	85	227	523	307	272	663	537	506	694	448
	Time (s.)	0.0010	0.0041	0.0084	0.0040	0.0050	0.0070	0.0070	0.0110	0.0143	0.0194	0.0130	0.0113	0.0311	0.0308	0.0256
	Makespan	48	40	227	73	189	104	215	523	337	284	663	537	489	694	468
10	Balance	43	39	213	68	189	81	215	523	322	272	663	537	489	694	468
	Time (s.)	0.0017	0.0035	0.0080	0.0042	0.0040	0.0041	0.0051	0.0103	0.0111	0.0181	0.0118	0.0105	0.0229	0.0271	0.0200
	Makespan	50	39	238	79	187	104	219	525	353	277	682	538	506	694	446
11	Balance	43	39	223	68	185	81	214	523	307	259	681	537	506	694	428
	Time (s.)	0.0010	0.0040	0.0077	0.0043	0.0040	0.0050	0.0056	0.0090	0.0115	0.0213	0.0137	0.0110	0.0260	0.0303	0.0216
	Makespan	48	38	230	69	192	92	200	523	327	286	681	550	487	694	428
12	Balance	42	37	227	68	192	79	200	523	307	263	681	550	487	694	428
	Time (s.)	0.0030	0.0041	0.0074	0.0031	0.0030	0.0054	0.0050	0.0081	0.0109	0.0185	0.0125	0.0122	0.0267	0.0590	0.0202
	Makespan	47	40	232	69	186	89	215	523	341	261	663	537	489	694	448
13	Balance	43	39	204	68	185	75	215	523	307	253	663	537	489	694	448
	Time (s.)	0.0009	0.0040	0.0070	0.0030	0.0033	0.0055	0.0043	0.0077	0.0098	0.0156	0.0100	0.0149	0.0262	0.0286	0.0183
	Makespan	48	40	227	73	189	104	215	523	337	284	663	537	489	694	468
14	Balance	43	39	213	68	189	81	215	523	322	272	663	537	489	694	468
	Time (s.)	0.0009	0.0031	0.0080	0.0035	0.0040	0.0049	0.0055	0.0071	0.0106	0.0173	0.0118	0.0111	0.0244	0.0273	0.0238

409–432 (2024) https://doi.org/10.1016/j.ejor.2023.05.017

[8] Ros, F.D., Soprano, M., Gaspero, L.D., Roitero, K.: Large language models for combinatorial optimization: A systematic review (2025) arXiv:2507.03637 [cs.AI]

[9] Özgüven, C., Özbakir, L., Yavuz, Y.: Mathematical models for job-shop scheduling problems with routing and process plan flexibility. Applied Mathematical Modelling **34**(6), 1539–1548 (2010) https://doi.org/10.1016/j.apm.2009.09.002 . Cited by: 276

[10] Roshanaei, V., Azab, A., Elmaraghy, H.: Mathematical modelling and a meta-heuristic for flexible job shop scheduling. International Journal of Production Research **51**(20), 6247–6274 (2013) https://doi.org/10.1080/00207543.2013.827806 . Cited by: 86

[11] Meng, L., Zhang, C., Ren, Y., Zhang, B., Lv, C.: Mixed-integer linear programming and constraint programming formulations for solving distributed flexible job shop scheduling problem. Computers and Industrial Engineering **142** (2020)

Table C2: Results of the Pareto front heuristics generated by REMoH framework on the Brandimarte Instances, via Gemini 2.0 Flash [58].

		mk01	mk02	mk03	mk04	mk05	mk06	mk07	mk08	mk09	mk10	mk11	mk12	mk13	mk14	mk15
1	Makespan	53	39	229	77	187	120	206	532	374	307	683	548	534	769	436
	Balance	40	38	216	67	181	87	206	523	299	255	683	548	534	734	398
	Time (s.)	0.0012	0.0030	0.0106	0.0040	0.0040	0.0076	0.0070	0.0110	0.0164	0.0367	0.0154	0.0147	0.0304	0.0310	0.0424
2	Makespan	41	32	221	74	186	76	161	523	323	241	683	534	471	707	422
	Balance	37	31	220	74	181	64	161	523	307	241	683	534	470	707	422
	Time (s.)	0.0102	0.0231	0.0785	0.0382	0.0417	0.1411	0.0690	0.1449	0.2377	0.4556	0.1982	0.1937	0.5520	0.4217	0.6594
3	Makespan	46	33	204	72	182	73	164	527	351	251	658	558	506	752	440
	Balance	42	33	204	67	180	68	164	523	327	237	658	535	506	734	403
	Time (s.)	0.0010	0.0026	0.0030	0.0020	0.0029	0.0051	0.0020	0.0030	0.0050	0.0060	0.0046	0.0030	0.0049	0.0051	0.0075
4	Makespan	41	32	221	74	186	76	161	523	323	241	683	534	471	707	422
	Balance	37	31	220	74	181	64	161	523	307	241	683	534	470	707	422
	Time (s.)	0.0103	0.0269	0.0945	0.0365	0.0470	0.1121	0.0785	0.1628	0.2222	0.4128	0.2016	0.1891	0.5986	0.3930	0.7291
5	Makespan	41	32	221	74	186	76	161	523	323	241	683	534	471	707	422
	Balance	37	31	220	74	181	64	161	523	307	241	683	534	470	707	422
	Time (s.)	0.0105	0.0263	0.1107	0.0330	0.0674	0.0932	0.0743	0.1749	0.2703	0.4290	0.2146	0.2012	0.6311	0.4117	0.6650
6	Makespan	41	32	221	74	186	76	161	523	323	241	683	534	471	707	422
	Balance	37	31	220	74	181	64	161	523	307	241	683	534	470	707	422
	Time (s.)	0.0105	0.0239	0.1184	0.0410	0.0501	0.0979	0.0805	0.1539	0.2228	0.4545	0.2256	0.1957	0.6254	0.3922	0.6932
7	Makespan	41	32	221	74	186	76	161	523	323	241	683	534	471	707	422
	Balance	37	31	220	74	181	64	161	523	307	241	683	534	470	707	422
	Time (s.)	0.0101	0.0233	0.1077	0.0354	0.0518	0.1121	0.0817	0.1718	0.2387	0.4742	0.2033	0.2106	0.6106	0.4129	0.7008
8	Makespan	49	33	204	73	191	74	166	531	348	249	659	559	519	753	424
	Balance	42	33	204	67	180	59	166	523	299	224	659	559	519	734	378
	Time (s.)	0.0010	0.0010	0.0030	0.0023	0.0011	0.0030	0.0020	0.0030	0.0050	0.0050	0.0021	0.0022	0.0040	0.0061	0.0054
9	Makespan	43	34	221	74	186	77	164	523	323	225	659	534	484	707	444
	Balance	43	32	220	74	178	70	164	523	307	223	659	534	484	707	442
	Time (s.)	0.0145	0.0322	0.1433	0.0404	0.0554	0.1439	0.1018	0.2264	0.3837	0.6498	0.2422	0.2435	0.9498	0.5214	0.9849
10	Makespan	41	32	221	74	186	76	161	523	323	241	683	534	471	707	422
	Balance	37	31	220	74	181	64	161	523	307	241	683	534	470	707	422
	Time (s.)	0.0100	0.0267	0.1054	0.0299	0.0412	0.1025	0.0830	0.1687	0.2517	0.4734	0.1986	0.1940	0.7519	0.3301	0.6851
11	Makespan	41	32	221	74	186	76	161	523	323	241	683	534	471	707	422
	Balance	37	31	220	74	181	64	161	523	307	241	683	534	470	707	422
	Time (s.)	0.0133	0.0374	0.1260	0.0507	0.0424	0.1082	0.0805	0.1710	0.2690	0.4869	0.2058	0.1937	0.7672	0.4124	0.7319
12	Makespan	49	33	204	73	191	74	166	531	348	249	659	559	519	753	424
	Balance	42	33	204	67	180	59	166	523	299	224	659	559	519	734	378
	Time (s.)	0.0010	0.0036	0.0030	0.0020	0.0020	0.0031	0.0020	0.0035	0.0041	0.0040	0.0028	0.0035	0.0070	0.0040	0.0050
13	Makespan	43	34	221	74	186	76	164	523	327	228	659	534	484	707	444
	Balance	43	32	220	74	178	63	164	523	307	227	659	534	484	707	442
	Time (s.)	0.0133	0.0359	0.1007	0.0528	0.0506	0.1222	0.0804	0.1697	0.2534	0.5052	0.1815	0.2112	0.6604	0.4210	0.7168

https://doi.org/10.1016/j.cie.2020.106347 . Cited by: 241

[12] Ham, A.M., Cakici, E.: Flexible job shop scheduling problem with parallel batch processing machines: Mip and cp approaches. Computers and Industrial Engineering **102**, 160–165 (2016) https://doi.org/10.1016/j.cie.2016.11.001 . Cited by: 71

[13] Jeong, K.-C., Kim, Y.-D.: A real-time scheduling mechanism for a flexible manufacturing system: Using simulation and dispatching rules. International Journal of Production Research **36**(9), 2609–2626 (1998) https://doi.org/10.1080/002075498192733 . Cited by: 117

[14] O'Keefe, R.M., Kasirajan, T.: Interaction between dispatching and next station selection rules in a dedicated flexible manufacturing system. International Journal of Production Research **30**(8), 1753–1772 (1992) https://doi.org/10.1080/00207549208948120 . Cited by: 35

[15] Alvarez-Valdes, R., Fuertes, A., Tamarit, J.M., Giménez, G., Ramos, R.: A heuristic to schedule flexible job-shop in a glass factory. European Journal of Operational Research **165**(2), 525–534 (2005) https://doi.org/10.1016/j.ejor.

Table C3: Results of the Pareto front heuristics generated by REMoH framework on the Brandimarte Instances, via GPT-4o mini [57].

		mk01	mk02	mk03	mk04	mk05	mk06	mk07	mk08	mk09	mk10	mk11	mk12	mk13	mk14	mk15
1	Makespan	50	40	232	89	189	95	212	531	369	277	669	563	533	753	435
	Balance	42	38	216	74	183	79	212	523	299	263	669	559	533	734	403
	Time (s.)	0.0010	0.0000	0.0010	0.0000	0.0000	0.0010	0.0010	0.0000	0.0011	0.0020	0.0000	0.0000	0.0010	0.0010	0.0020
2	Makespan	49	35	204	73	191	81	166	531	349	257	659	559	520	753	434
	Balance	42	34	204	67	180	66	166	523	299	234	659	559	517	734	378
	Time (s.)	0.0000	0.0010	0.0020	0.0000	0.0010	0.0030	0.0010	0.0030	0.0029	0.0050	0.0020	0.0020	0.0029	0.0020	0.0040
3	Makespan	47	40	219	76	189	99	212	539	366	281	669	560	533	753	441
	Balance	42	38	216	66	183	79	212	523	299	252	669	559	533	734	403
	Time (s.)	0.0010	0.0010	0.0010	0.0000	0.0000	0.0011	0.0010	0.0020	0.0020	0.0020	0.0010	0.0010	0.0010	0.0020	0.0010
4	Makespan	45	29	204	82	186	80	168	523	351	264	659	559	513	763	426
	Balance	42	29	204	74	184	74	168	523	336	242	659	559	513	734	396
	Time (s.)	0.0000	0.0010	0.0010	0.0010	0.0000	0.0010	0.0010	0.0020	0.0020	0.0020	0.0000	0.0010	0.0040	0.0010	0.0020
5	Makespan	45	31	204	82	186	79	163	523	352	256	653	540	520	763	430
	Balance	42	30	204	76	184	72	163	523	319	222	653	535	520	734	391
	Time (s.)	0.0010	0.0011	0.0010	0.0010	0.0000	0.0010	0.0010	0.0020	0.0010	0.0020	0.0010	0.0010	0.0020	0.0010	0.0010
6	Makespan	46	31	204	73	191	73	160	531	340	244	671	559	502	753	441
	Balance	42	29	204	67	181	57	160	523	299	214	671	559	502	734	378
	Time (s.)	0.0010	0.0010	0.0010	0.0010	0.0020	0.0020	0.0000	0.0021	0.0020	0.0037	0.0010	0.0020	0.0020	0.0010	0.0021
7	Makespan	46	31	204	73	191	73	160	531	340	244	671	559	502	753	441
	Balance	42	29	204	67	181	57	160	523	299	214	671	559	502	734	378
	Time (s.)	0.0000	0.0010	0.0020	0.0010	0.0010	0.0010	0.0020	0.0020	0.0020	0.0031	0.0010	0.0009	0.0030	0.0020	0.0020
8	Makespan	46	31	207	73	191	76	160	531	344	265	671	559	502	769	433
	Balance	42	29	204	67	181	61	160	523	299	221	671	559	502	734	378
	Time (s.)	0.0010	0.0010	0.0020	0.0010	0.0020	0.0020	0.0010	0.0010	0.0020	0.0030	0.0010	0.0009	0.0020	0.0010	0.0022
9	Makespan	50	40	226	89	189	95	212	531	369	277	669	563	533	753	449
	Balance	42	38	216	74	183	79	212	523	299	263	669	559	533	734	403
	Time (s.)	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0010	0.0010	0.0010	0.0009	0.0000	0.0010	0.0010	0.0021

2004.04.020 . Cited by: 71

[16] Chang, Y.-L., Matsuo, H., Sullivan, R.S.: A bottleneck-based beam search for job scheduling in a flexible manufacturing system. International Journal of Production Research **27**(11), 1949–1961 (1989) https://doi.org/10.1080/00207548908942666 . Cited by: 74

[17] Gao, J., Sun, L., Gen, M.: A hybrid genetic and variable neighborhood descent algorithm for flexible job shop scheduling problems. Computers and Operations Research **35**(9), 2892–2907 (2008) https://doi.org/10.1016/j.cor.2007.01.001 . Cited by: 491

[18] Ding, H., Gu, X.: Improved particle swarm optimization algorithm based novel encoding and decoding schemes for flexible job shop scheduling problem. Computers and Operations Research **121** (2020) https://doi.org/10.1016/j.cor.2020.104951 . Cited by: 111

[19] Martí, R., Sevaux, M., Sörensen, K.: Fifty years of metaheuristics. European Journal of Operational Research **321**(2), 345–362 (2025) https://doi.org/10.1016/j.ejor.2024.04.004

[20] Chen, D., Huang, Y.: Integrating reinforcement learning and large language models for crop production process management optimization and control through a new knowledge-based deep learning paradigm. Computers and Electronics in Agriculture **232**, 110028 (2025) https://doi.org/10.1016/j.compag.2025.110028

[21] Wu, J., Lai, Z., Chen, S., Tao, R., Zhao, P., Hovakimyan, N.: The New

Agronomists: Language Models are Experts in Crop Management (2024). https://arxiv.org/abs/2403.19839

[22] Zhang, S., Gong, C., Wu, L., Liu, X., Zhou, M.: AutoML-GPT: Automatic Machine Learning with GPT (2023). https://arxiv.org/abs/2305.02499

[23] Zhang, M.R., Desai, N., Bae, J., Lorraine, J., Ba, J.: Using Large Language Models for Hyperparameter Optimization (2024). https://arxiv.org/abs/2312.04528

[24] Liu, S., Gao, C., Li, Y.: Large Language Model Agent for Hyper-Parameter Optimization (2025). https://arxiv.org/abs/2402.01881

[25] Liu, Y., He, H., Han, T., Zhang, X., Liu, M., Tian, J., Zhang, Y., Wang, J., Gao, X., Zhong, T., Pan, Y., Xu, S., Wu, Z., Liu, Z., Zhang, X., Zhang, S., Hu, X., Zhang, T., Qiang, N., Liu, T., Ge, B.: Understanding llms: A comprehensive overview from training to inference. Neurocomputing **620**, 129190 (2025) https://doi.org/10.1016/j.neucom.2024.129190

[26] Fernando, C., Banarse, D., Michalewski, H., Osindero, S., Rocktäschel, T.: Promptbreeder: Self-Referential Self-Improvement Via Prompt Evolution (2023). https://arxiv.org/abs/2309.16797

[27] Zhong, R., Xu, Y., Zhang, C., Yu, J.: Leveraging large language model to generate a novel metaheuristic algorithm with CRISPE framework. Cluster Comput. **27**(10), 13835–13869 (2024)

[28] Liu, F., Tong, X., Yuan, M., Zhang, Q.: Algorithm Evolution Using Large Language Model (2023). https://arxiv.org/abs/2311.15249

[29] Stein, N., Bäck, T.: Llamea: A large language model evolutionary algorithm for automatically generating metaheuristics. IEEE Transactions on Evolutionary Computation **PP**, 1–1 (2024) https://doi.org/10.1109/TEVC.2024.3497793

[30] Hemberg, E., Moskal, S., O'Reilly, U.-M.: Evolving Code with A Large Language Model (2024). https://arxiv.org/abs/2401.07102

[31] Brahmachary, S., Joshi, S.M., Panda, A., Koneripalli, K., Sagotra, A.K., Patel, H., Sharma, A., Jagtap, A.D., Kalyanaraman, K.: Large Language Model-Based Evolutionary Optimizer: Reasoning with elitism (2024). https://arxiv.org/abs/2403.02054

[32] Liu, S., Chen, C., Qu, X., Tang, K., Ong, Y.-S.: Large Language Models as Evolutionary Optimizers (2024). https://arxiv.org/abs/2310.19046

[33] Bradley, H., Dai, A., Teufel, H., Zhang, J., Oostermeijer, K., Bellagente, M., Clune, J., Stanley, K., Schott, G., Lehman, J.: Quality-Diversity through AI

Feedback (2023). https://arxiv.org/abs/2310.13032

[34] Meyerson, E., Nelson, M.J., Bradley, H., Gaier, A., Moradi, A., Hoover, A.K., Lehman, J.: Language Model Crossover: Variation through Few-Shot Prompting (2024). https://arxiv.org/abs/2302.12170

[35] Bradley, H., Fan, H., Galanos, T., Zhou, R., Scott, D., Lehman, J.: The openelm library: Leveraging progress in language models for novel evolutionary algorithms, 177–201 (2024) https://doi.org/10.1007/978-981-99-8413-8_10

[36] Liu, F., Lin, X., Wang, Z., Yao, S., Tong, X., Yuan, M., Zhang, Q.: Large Language Model for Multi-objective Evolutionary Optimization (2024). https://arxiv.org/abs/2310.12541

[37] Huang, J., Li, X., Gao, L., Liu, Q., Teng, Y.: Automatic programming via large language models with population self-evolution for dynamic job shop scheduling problem (2024). https://arxiv.org/abs/2410.22657

[38] Yatong, W., Yuchen, P., Yuqi, Z.: TS-EoH: An Edge Server Task Scheduling Algorithm Based on Evolution of Heuristic (2024). https://arxiv.org/abs/2409.09063

[39] Ye, H., Wang, J., Cao, Z., Berto, F., Hua, C., Kim, H., Park, J., Song, G.: ReEvo: Large Language Models as Hyper-Heuristics with Reflective Evolution (2024). https://arxiv.org/abs/2402.01145

[40] Dat, P.V.T., Doan, L., Binh, H.T.T.: HSEvo: Elevating Automatic Heuristic Design with Diversity-Driven Harmony Search and Genetic Algorithm Using LLMs (2024). https://arxiv.org/abs/2412.14995

[41] Liu, F., Tong, X., Yuan, M., Lin, X., Luo, F., Wang, Z., Lu, Z., Zhang, Q.: Evolution of Heuristics: Towards Efficient Automatic Algorithm Design Using Large Language Model (2024). https://arxiv.org/abs/2401.02051

[42] Gavval, R., Ravi, V.: Clustering bank customer complaints on social media for analytical crm via multi-objective particle swarm optimization, 213–239 (2020) https://doi.org/10.1007/978-3-030-33820-6_9

[43] Xiong, H., Shi, S., Ren, D., Hu, J.: A survey of job shop scheduling problem: The types and models. Computers & Operations Research **142**, 105731 (2022)

[44] Leusin, M.E., Frazzon, E.M., Uriona Maldonado, M., Kück, M., Freitag, M.: Solving the job-shop scheduling problem in the industry 4.0 era. Technologies **6**(4), 107 (2018)

[45] Xie, J., Gao, L., Peng, K., Li, X., Li, H.: Review on flexible job shop scheduling. IET collaborative intelligent manufacturing **1**(3), 67–77 (2019)

[46] Brandimarte, P.: Routing and scheduling in a flexible job shop by tabu search. Annals of Operations Research **41**(3), 157–183 (1993) https://doi.org/10.1007/bf02023073

[47] Barnes, J.W., Chambers, J.B.: Solving the job shop scheduling problem with tabu search. IIE Transactions **27**(2), 257–263 (1995) https://doi.org/10.1080/07408179508936739

[48] Laborie, P.: Ibm ilog cp optimizer for detailed scheduling illustrated on three problems. In: Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems: 6th International Conference, CPAIOR 2009 Pittsburgh, PA, USA, May 27-31, 2009 Proceedings 6, pp. 148–162 (2009). Springer

[49] Da Col, G., Teppan, E.C.: Industrial size job shop scheduling tackled by present day cp solvers. In: Principles and Practice of Constraint Programming: 25th International Conference, CP 2019, Stamford, CT, USA, September 30–October 4, 2019, Proceedings 25, pp. 144–160 (2019). Springer

[50] Perron, L., Didier, F.: CP-SAT (2024)

[51] Cplex, I.I.: V12. 1: User's manual for cplex. International Business Machines Corporation **46**(53), 157 (2009)

[52] Ho, K.-H., Cheng, J.-Y., Wu, J.-H., Chiang, F., Chen, Y.-C., Wu, Y.-Y., Wu, I.-C.: Residual scheduling: A new reinforcement learning approach to solving job shop scheduling problem. IEEE Access **12**, 14703–14718 (2024)

[53] Echeverria, I., Murua, M., Santana, R.: Leveraging constraint programming in a deep learning approach for dynamically solving the flexible job-shop scheduling problem. Expert Systems with Applications **265**, 125895 (2025)

[54] Yuraszeck, F., Montero, E., Canut-De-Bon, D., Cuneo, N., Rojel, M.: A constraint programming formulation of the multi-mode resource-constrained project scheduling problem for the flexible job shop scheduling problem. IEEE Access **11**, 144928–144938 (2023) https://doi.org/10.1109/ACCESS.2023.3345793

[55] Mousakhani, M.: Sequence-dependent setup time flexible job shop scheduling problem to minimise total tardiness. International journal of production research **51**(12), 3476–3487 (2013)

[56] Caldeira, R.H., Gnanavelbabu, A.: A simheuristic approach for the flexible job shop scheduling problem with stochastic processing times. Simulation **97**(3), 215–236 (2021)

[57] OpenAI: GPT-4 Technical Report. https://arxiv.org/abs/2303.08774. Accessed via ChatGPT (GPT-4o), April 2025 (2023)

[58] DeepMind, G.: Gemini 2: Our Most Capable Model Yet. https://deepmind.google/discover/blog/gemini-2-our-most-capable-model-yet. Accessed: 2025-04-15 (2023)

[59] Dai, D., Guo, D., Yang, D., Zhang, H., Song, J., Zhang, R., Xu, R., Zhu, Q., Ma, S., Wang, P., et al.: Deepseek-v3: Scaling mixture-of-experts with multi-head latent attention. arXiv preprint arXiv:2412.19437 (2024)