Assessing the Effectiveness of LLMs as Autonomous Solvers for University Course Timetabling

Basma Alharbi
Department of Computer Science and Artificial Intelligence, University of Jeddah, Saudi Arabia
bmalharbi@uj.edu.sa

Abstract: *Large Language Models (LLMs) have shown promise in complex reasoning, yet their capability to solve Nondeterministic Polynomial-time Hard (NP-Hard) combinatorial optimization problems remains largely unverified. This paper evaluates the performance of high-reasoning LLMs on the Curriculum-Based Course TimeTabling (CB-CTT) problem. We conduct a systematic evaluation across four dimensions: baseline performance, the impact of prompt engineering, the effectiveness of iterative refinement with feedback, and scalability with problem complexity. Our investigation revealed that a leading model can solve small-scale instances with an 80% feasibility rate. The results showed that: 1) standard prompt engineering techniques do not improve the performance of the models, 2) Iterative feedback can trigger powerful one-shot corrections, yet unstable and unreliable, 3) model's performance degrades as problem complexity grows. The findings indicate that LLMs, in their current state, are not reliable autonomous solvers for complex, real-world scheduling problems due to their inability to consistently enforce hard constraints and maintain solution feasibility as problem size and complexity increase. However, their ability to support problem formulation, heuristic guidance, and partial solution generation makes them valuable components within hybrid scheduling frameworks. We conclude that future work should focus on hybrid approaches that integrate LLMs with established optimization techniques, leveraging LLMs for problem modeling, decomposition, and heuristic guidance while relying on classical solvers to ensure feasibility and scalability.*

Keywords: *Automated scheduling, combinatorial optimization, constraint satisfaction problems, curriculum-based course timetabling, ITC-2007 benchmark, large language models.*

Received, August,28, 2025; accepted January, 11, 2026
https://doi.org/10.34028/iajit/23/5/7

1. Introduction

University course timetabling remains a critical yet complex challenge for educational institutions worldwide [15]. The Curriculum-Based Course Timetabling (CB-CTT) problem requires assigning lectures to time periods and rooms while satisfying numerous hard constraints and minimizing soft constraint violations [11]. This problem represents a constrained combinatorial optimization challenge, requiring not only the identification of feasible solutions that meet all hard constraints (i.e., a constraint satisfaction task) but also the optimization of these solutions to minimize penalties associated with soft constraint violations. Despite decades of research producing sophisticated algorithms, creating high-quality timetables remains difficult for most institutions [16]. This paper explores whether Large Language Models (LLMs) can make this Nondeterministic Polynomial-time Hard (NP-Hard) problem more accessible while investigating their fundamental capabilities on structured combinatorial tasks.

Current CB-CTT approaches present significant barriers to practical adoption [3, 21]. State-of-the-art solutions require deep expertise in operations research, constraint programming, or metaheuristic design [23]. Institutions either invest in specialized commercial software with steep learning curves or rely on academic implementations that demand technical knowledge to configure and operate [4, 26]. They may also rely on manual or semi-automated approaches that produce suboptimal results.

The rise of LLMs offers an exciting opportunity to make complex optimization tasks more accessible to a broader audience. Their natural language understanding capabilities could enable non-experts to specify timetabling requirements conversationally, without learning formal constraint languages or mathematical notation. Recent work has begun investigating LLM reasoning capabilities on algorithmic problems. Fan *et al*. [12] introduced NP-Hard Evaluation benchmark (NPHardEval), a dynamic benchmark evaluating LLMs across 900 algorithmic questions extending to NP-Hard complexity, revealing both the potential and limitations of LLMs on computationally complex tasks. However, their evaluation focused on algorithmic reasoning rather than practical constraint satisfaction and optimization problems like timetabling.

CB-CTT poses distinct challenges that could inherently restrict the effectiveness of LLMs. Being an NP-hard problem, it demands consistent handling of numerous interrelated constraints while navigating an exponentially vast solution space. LLMs are required to reason over discrete combinatorial structures without dedicated search algorithms, carry out implicit constraint

propagation without the ability to backtrack, and optimize numerical objectives solely through the sequential generation of tokens. These challenges differ substantially from the pattern matching and language modeling tasks where LLMs excel. Moreover, the dual nature of CB-CTT-requiring both constraint satisfaction for feasibility and combinatorial optimization for quality-tests whether LLMs can handle problems that demand both logical consistency and numerical optimization.

This paper presents the first comprehensive evaluation of state-of-the-art LLMs on the CB-CTT problem. We investigate whether LLMs can generate feasible timetables that satisfy hard constraints while minimizing soft constraint violations. Our research addresses four key questions:

- **RQ1:** How do different LLMs perform on CB-CTT instances in terms of feasibility, solution quality, and stability?
- **RQ2:** What is the impact of prompt engineering strategies on timetabling performance?
- **RQ3:** Can iterative refinement with constraint violation feedback improve solutions?
- **RQ4:** How does performance scale with problem complexity?

Our main contributions are:

- Systematic evaluation of state-of-the-art LLMs (Grok-4, Gemini 2.5 Pro, OpenAI-o3 and O4-mini, Claude Opus 4, Mistral) on the standardized ITC-2007 CB-CTT benchmark, addressing the critical absence of domain-specific benchmarks for LLM performance in optimization and scheduling tasks, and establishing initial baselines for future research in AI-assisted educational timetabling.
- An analysis of prompt engineering strategies specifically designed for constraint satisfaction and optimization in timetabling contexts.
- Empirical evidence of limitations in LLM reasoning for constrained combinatorial optimization, including quantitative analysis of failure modes, scalability constraints, and the bounded effectiveness of iterative refinement strategies in overcoming these limitations.

The remainder of this paper is organized as follows. Section 2 reviews related work on traditional CB-CTT approaches and LLM applications to constraint satisfaction and combinatorial optimization. Section 3 formally defines the problem. Section 4 presents our methodology. Section 5 reports experimental results addressing our research questions, and discusses findings and limitations. Section 5 concludes with implications for future research.

2. Related Work

This section reviews the literature that forms the foundation for our research, organized into three key areas. First, we examine traditional CB-CTT Approaches to understand the established, state-of-the-art methods for solving this problem. Second, we survey the literature on LLMs for Constraint Satisfaction to assess their core logical reasoning capabilities. Finally, we analyze the direct application of LLMs for Combinatorial Optimization to see how they have been applied to similar structured problems. Together, these sections map the existing landscape and highlight the specific research gap our work aims to fill: evaluating the potential of state-of-the-art LLMs to solve a complex, real-world timetabling problem.

2.1. Traditional CB-CTT Approaches

The Curriculum-Based Course Timetabling Problem has been extensively studied since its formalization in the ITC-2007 competition [7, 10, 11]. Early and influential approaches focused on metaheuristic methods. Simulated Annealing (SA) emerged as particularly effective, with research demonstrating that properly tuned SA algorithms can achieve competitive results on the ITC-2007 benchmark instances [2, 9]. Other metaheuristics, such as genetic algorithms [23], variable neighborhood search [6], and Tabu Search [5] have also shown promising results, especially when hybridized with local search procedures to balance exploration and exploitation of the solution space [20].

Exact methods based on Integer Programming (IP) have provided important theoretical insights and practical advancements in solving university timetabling problems. For instance, early work by Burke *et al*. [8] developed branch-and-cut procedures that achieved optimal solutions for several smaller ITC-2007 instances, demonstrating how Intellectual Property (IP) formulations can handle complex curricular constraints [7]. However, scalability remains a significant limitation for larger, more complex problems. Recent research has addressed this through innovative decomposition strategies and hybrid approaches, such as [13, 25].

More recent work has begun to explore other Artificial Intelligence (AI) paradigms, such as Reinforcement Learning (RL). For example, Xiao *et al*. [29] propose an automated planning method that treats the dynamic timetabling problem as a series of decision-making tasks. In their framework, an RL agent acts as an automated planner, operating in a simulated environment that reflects the problem's constraints. The agent continuously refines its strategy by evaluating the outcomes of its actions, using a priority-driven reward system and an experience replay mechanism to enhance solution diversity and reduce the search space. This approach demonstrates the ongoing effort to find more adaptive and automated solutions, moving beyond static optimization models.

For a comprehensive analysis of state-of-the-art approaches for solving CB-CTT problems, we refer the

readers to the following reviews [10, 23].

2.2. LLMs for Constraint Satisfaction

Recent research has explored the capabilities of LLMs for constraint satisfaction problems. The ZebraLogic benchmark, for instance, revealed fundamental limitations in the ability of LLMs to handle multiple interdependent constraints [18]. Experiments show a pronounced drop in performance as the logical complexity of a problem increases, a trend that overshadows gains from model scaling or expanded training data.

Additionally, studies such as [30] provide insights into how LLM attention patterns correlate with constraint satisfaction capabilities, but practical applications remain limited. Similarly, studies on mathematical reasoning show that while LLMs perform well on elementary problems, their performance degrades significantly with increased complexity, a finding that is particularly relevant for the highly constrained, combinatorial domains like timetabling [19, 24, 31].

2.3. LLMs for Combinatorial Optimization

The use of LLMs in combinatorial optimization is still in its early stages. A highly relevant study by Abgaryan *et al*. [1] evaluated LLMs on Job Shop Scheduling Problems (JSSP). The study found that by fine-tuning models, it was possible to achieve performance comparable to specialized neural approaches, but only on small-scale instances, with effectiveness declining rapidly on larger problems. However, fine-tuning carries a significant computational overhead, which is considered a major limitation for this work.

The existing literature outlines a clear division: traditional timetabling relies on specialized, high-performing algorithms that demand substantial domain knowledge, while LLMs, when applied to structured reasoning tasks, often fall short in meeting the strict and global constraints these problems impose. This highlights a central gap our research aims to address: the lack of systematic investigation into whether a state-of-the-art LLM can function as a direct, end-to-end solver for a real-world combinatorial problem like CB-CTT without relying on fine-tuning.

Exploring this gap is crucial as it could uncover new pathways for leveraging general-purpose models in solving complex, constraint-heavy tasks. Demonstrating the potential (or limitations) of LLMs in this domain may significantly broaden their applicability beyond natural language processing, offering more accessible and adaptable solutions to traditionally expert-driven problems.

3. Problem Definition

Let the set of courses be denoted by $C=\{c_1,c_2,...,c_n\}$, where each course $c_i \in C$ is associated with a specific teacher and a set of students. Additionally, each course has a number of lectures to be scheduled over a given number of days. Let the set of rooms be $R=\{r_1,r_2,...,r_m\}$, where each room has a capacity, $cap(r_j)$, representing the maximum number of available seats. Let the set of teaching days be $D=\{d_1,d_2,...,d_p\}$ and the set of timeslots per day be $TS=\{ts_1,ts_2,...,ts_q\}$. A period is a pair (d, ts) where $d \in D$ and $ts \in TS$. The set of all periods is $P = D \times TS$. Let the set of curricula be $K=\{k_1,k_2,...,k_u\}$. A curriculum $k_l \in K$ is a set of courses that are taken together by the same group of students and therefore cannot have scheduling conflicts.

A solution to the CB-CTT problem is a mapping that assigns each course lecture to a period and a room, subject to a set of constraints. Let a solution be represented by a set of assignments $X=\{x_{c,p,r}\}$, where $X=\{x_{c,p,r}\}=1$ if a lecture of a course $c \in$ C is assigned to a period $p \in P$ and room $r \in R$, and $X=\{x_{c,p,r}\}=0$ otherwise.

The constraints are categorized as hard and soft constraints. Hard constraints are conditions that must be satisfied for a timetable to be feasible. Soft constraints are desirable properties. Violating them incurs a penalty, and the goal is to minimize the total penalty score.

In this context, there is a total of four hard constraints (H1-H4) and four soft constraints (S1-S4):

- **H1 (Lectures):** Every lecture for each course must be scheduled in a unique period.
- **H2 (Room Occupancy):** No two lectures can be held in the same room during the same period.
- **H3 (Conflicts):** Lectures from courses that are part of the same curriculum or taught by the same instructor must be scheduled at different times.
- **H4 (Availability):** Lectures cannot be scheduled in periods when the designated instructor is unavailable.
- **S1 (Room Capacity):** The room assigned to a lecture must have enough seats for all attending students.
- **S2 (Course Spreading):** Lectures of a course should be spread over at least a specified minimum number of days.
- **S3 (Curriculum Compactness):** Lectures belonging to the same curriculum should be scheduled in consecutive periods on the same day.
- **S4 (Room Stability):** All lectures of a course should be held in the same room.

4. Methodology

In this section, we present our experimental methodology for evaluating the capabilities and limitations of Large Language Models in solving the Curriculum-Based Course Timetabling problem. Our approach consists of a systematic evaluation framework designed to address each research question through

controlled experiments. We first describe the dataset and instance selection criteria (Section 4.1), followed by the rationale for LLM model selection (Section 4.2). We then detail our four-phase experimental framework (Section 4.3), which includes baseline model performance[1]and stability evaluation, prompt engineering analysis, iterative refinement evaluation, and scalability assessment. Finally, we describe our evaluation metrics (Section 4.4) and explain our validation process and metrics collection methodology (Section 4.5). This structured approach enables us to comprehensively assess both the potential and fundamental limitations of LLMs for constrained combinatorial optimization tasks. Figure 1 provides a high-level overview of the methodology.

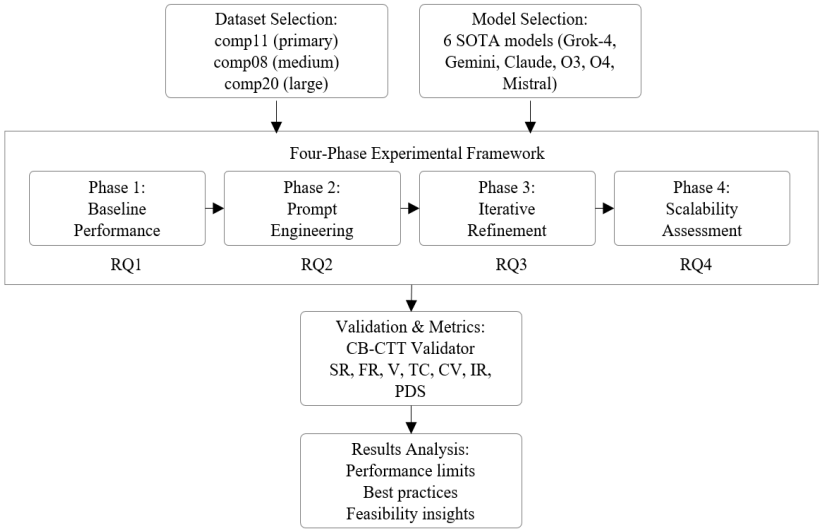

Figure 1. High-level overview of the experimental methodology for evaluating LLM performance on CB-CTT. The framework systematically addresses four research questions through controlled experiments, from baseline evaluation to scalability assessment.

4.1. Dataset

We selected three instances from the ITC-2007 CB-CTT benchmark dataset [11], which is the standard for evaluating timetabling algorithms. Table 1 summarizes the characteristics of the selected instances.

Table 1. ITC-2007 CB-CTT benchmark dataset: characteristics of selected instances.

Instance	Name	Courses	Room	Day	Periods per day	Total periods	Curricula	Constraints	Optimal total cost	Role in study
comp11	Fis0506-2	30	5	5	9	45	13	94	0	Primary (RQ1-4)
comp08	Ing0607-3	86	18	5	5	25	61	478	0	Scalability Test (RQ4)
comp20	Ing0506-2	121	19	5	5	25	78	691	0	Scalability Test (RQ4)

The primary instance, comp11 (Fis0506-2), contains 30 courses, 5 rooms, 5 days with 9 periods per day (45 total periods), 13 curricula, and 94 constraints. This instance represents a relatively manageable problem size that allows for thorough analysis while remaining computationally tractable for manual verification. Despite its moderate size, comp11 presents genuine timetabling challenges with multiple interacting constraints and limited room resources relative to the number of courses.

For scalability analysis (RQ4), we selected two additional instances with increasing complexity. The comp08 instance (Ing0607-3) represents medium complexity with 86 courses, 18 rooms, 25 periods (5 days × 5 slots), 61 curricula, and 478 constraints. The comp20 instance (Ing0506-2) represents large-scale complexity with 121 courses, 19 rooms, 25 periods, 78 curricula, and 691 constraints. The transition from 30 to 121 courses represents a four-fold increase in problem scale, substantially expanding the solution space and constraint interactions.

All three selected instances have known optimal solutions achievable by state-of-the-art traditional methods, with zero hard constraint violations and zero soft constraint penalties [29]. This provides a clear benchmark to evaluate LLM performance, as any non-zero violation or cost represents a deviation from the optimal solution.

4.2. Model Selection

We selected six state-of-the-art reasoning LLMs based on their performance on specific reasoning tasks, as ranked by online leaderboards1. Our selection prioritized models that have demonstrated strong capabilities across four key reasoning benchmarks: MMLU-Pro for advanced reasoning and knowledge integration [28], Humanity's Last Exam for complex reasoning under challenging conditions [22], IFEval for precise instruction following [32], and Arena-Hard-Auto for

[1] MMLU-pro, Humanity's last exam, IFBench as reported in https://artificialanalysis.ai/

long-context reasoning [17]. Additionally, we considered model performance on NPHardEval [12], which represents the closest available benchmark to optimization-specific evaluation, testing LLMs on algorithmic problems extending to NP-Hard complexity. These benchmarks collectively assess the cognitive capabilities essential for constraint satisfaction problems-logical reasoning, instruction adherence, and sustained coherence across complex problem specifications. While no domain-specific leaderboards exist for LLM performance on optimization or scheduling tasks, we intentionally chose models based on these general but rigorous reasoning evaluations, supplemented by NPHardEval results, to assess their ability to transfer reasoning skills to the specialized domain of timetabling. This absence of optimization-specific benchmarks underscores a critical research gap that our systematic evaluation of LLMs on the standardized ITC-2007 CB-CTT benchmark aims to address, establishing initial baselines for LLM performance in educational scheduling contexts.

The selected models represent both closed-source commercial systems and open-source alternatives. Our evaluation includes Grok-42 from xAI, which have shown strong performance on complex reasoning tasks. We also selected Gemini 2.53 Pro from Google, known for its enhanced logical reasoning capabilities. From OpenAI, we included OpenAI-o3 and o4-mini4, representing different scales of reasoning-optimized models. Claude Opus 45 from Anthropic was chosen for its reported strengths in following complex instructions and maintaining consistency across long contexts. Finally, Mistral6 was included as a representative open-source model to assess whether accessibility comes at the cost of performance on structured optimization tasks. Version of all selected models is of July 2025. Table 2 provides a list of the selected models, main characteristics, and average performance on key benchmarks.

Table 2. Selected LLM models and charactersitics. avg. per. is the average perfromance on key benchmarks1.

Model	Provider	Type	Context Window	Avg. Per.
Grok-4	xAI	Closed	128K	55%
Gemini 2.5 Pro	Google	Closed	2M	52%
O3	OpenAI	Closed	128K	59%
O4-mini	OpenAI	Closed	128K	56%
Claude Opus 4	Anthropic	Closed	200K	51%
Mistral	Mistral AI	Open	32K	40%

All models were accessed through their respective web interfaces, ensuring we evaluated the current versions available to general users. This approach reflects real-world usage scenarios where institutions would interact with these systems without specialized API access or custom configurations. The web interface also eliminates potential variations from different temperature settings or sampling parameters, as we used each system's default configuration for consistency.

4.3. Experimental Framework

Our experimental design consists of four distinct phases, each addressing a specific research question through controlled experiments. This systematic approach enables us to isolate different factors affecting LLM performance on CB-CTT instances while maintaining reproducibility and statistical validity. Figure 2 illustrates these phases, and description of each phase is provided next.

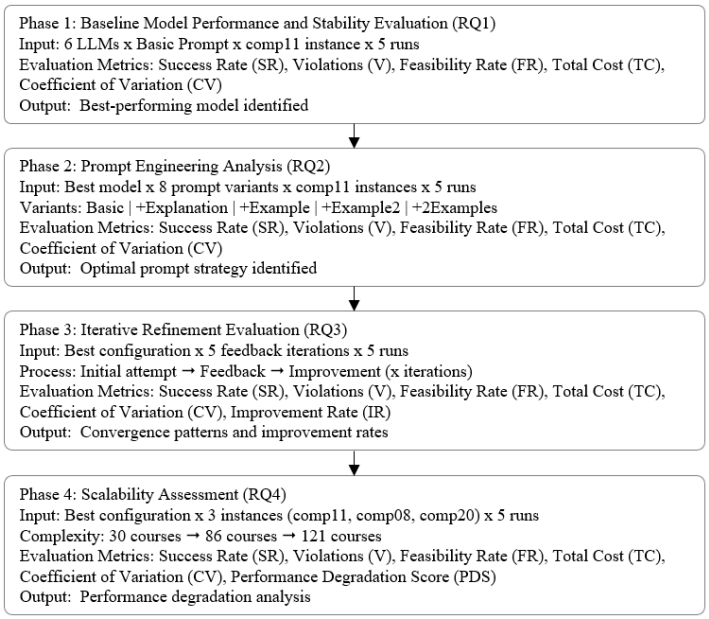

Figure 2. Four-phase experimental framework for evaluating LLM performance on CB-CTT. Each phase builds upon the results of the previous phase, using the best-performing configuration identified.

- **Phase 1: Baseline Model Performance and Stability Evaluation (RQ1)** evaluates selected models using a basic prompt on the primary instance (comp11). Each model is tested five times to assess

both performance and stability. This phase establishes baseline capabilities and identifies the best-performing model for subsequent experiments. The basic prompt provides the problem statement, instance data, and output format requirements without additional guidance or examples.

- **Phase 2: Prompt Engineering Analysis (RQ2)** investigates how different prompting strategies affect the best-performing model from Phase 1. We systematically tested eight prompt variants that combine a basic prompt with different guidance components to understand their individual and synergistic effects on solution quality.

Our prompting strategy follows a combinatorial approach where all variants build upon a foundational basic prompt. This basic prompt contains three essential components:

1) a task description requesting the model to solve the CB-CTT problem.
2) the complete instance data in text format including all courses, rooms, curricula, teacher assignments, and constraints,
3) a specified output format requiring the solution as structured assignments.

This minimal baseline tests whether LLMs can solve the problem using only fundamental specifications.

We enhance the basic prompt with three types of additional guidance:

- Input Explanation: Detailed clarifications of each constraint type, explaining operational meanings such as curriculum conflicts preventing simultaneous student attendance.
- Example 1: A demonstration of a toy instance with a known optimal solution. This example is designed to show what a successful, feasible timetable looks like, illustrating a clear pattern of constraint satisfaction.
- Example 2: A demonstration of a smaller toy instance with a suboptimal solution. This example is designed to be educational, explicitly showing how all four types of soft constraints manifest as violations when resources are tight, thereby illustrating the problem's penalty structure.

Table 3 provides a comprehensive comparison between the two examples, showing the similarities and differences between the instances and the provided solutions. These examples were chosen for their complementary design and educational coverage. The first shows that optimal solutions are achievable, while the second provides a clear model of the penalty system by demonstrating a non-optimal but valid outcome.

Table 3. Comparison of toy example 1 and toy example 2.

	Characteristic	Toy example 1	Toy example 2	Key difference
Problem size	# Courses	4	3	Example 1 is larger
	# Rooms	2	2	Same
	# Days	5	3	Example 1 has more days
	Periods per day	4	3	Example 1 has more slots
	Total periods	20	9	Example 1 has more periods (2.2X)
	Total lectures	16	7	Example 1 has more lectures (2.3X)
	# Curricula	2	2	Same
	Unavailability constraints	8	4	Example 1 has more constraints
Problem characteristics	Period utilization	80\% (16/20)	78\% (7/9)	Similar density
	Avg lectures/course	4	2.3	Example 1 courses are longer
	Room capacity pressure	Moderate	High	Example 2 has tighter capacity
	Course enrollment range	18-42	30-42	Example 2 has higher minimum
Solution quality	# Hard violations	0	0	Both feasible
	# Total soft cost	0	39	Example 1 is optimal
	# Room capacity violations	0	18 (2 instances)	Example 2 exceeds capacity
	# Min working days violations	0	15 (2 courses)	Example 2 lacks spreading
	# Compactness violations	0	4 (2 instances)	Example 2 has isolated lectures
	# Room stability violations	0	2 (2 courses)	Example 2 uses multiple rooms
Educational value	Demonstrates feasibility	Yes	Yes	Both show valid schedules
	Shows optimal solution	Yes	No	Different quality levels
	Illustrates soft constraints violations	No	Yes (all 4 types)	Example 2 shows violations
	Complexity for LLM	Moderate	Simple	Different challenge levels

These components are combined systematically to create eight distinct prompt variants, as shown in Table 4. This systematic design allows us to isolate the contribution of each component and identify potential synergies. Each variant is evaluated five times on the comp11 instance to assess both performance and consistency. The combination approach tests whether comprehensive guidance (variant 8) can overcome inherent LLM limitations, or if simpler prompts prove more effective by avoiding information overload.

Table 4. Prompt engineering combinations tested.

ID	Prompt variant	Basic	EXP	EX1	EX2
P1	basic	✓			
P2	basic + explanation	✓	✓		
P3	basic + example1	✓		✓	
P4	basic + example2	✓			✓
P5	basic + explanation + example1	✓	✓	✓	
P6	basic + explanation + example2	✓	✓		✓
P7	basic + 2examples	✓		✓	✓
P8	basic + explanation + 2examples	✓	✓	✓	✓

Across all variants, we maintain consistent output format requirements to ensure automated validation. Models must structure their responses to specify, for each course, the assigned period (day and timeslot) and room for each lecture. This standardization enables reliable parsing and constraint checking while allowing models flexibility in their solution approach.

- **Phase 3: Iterative Refinement Evaluation (RQ3)** examines whether LLMs can improve solutions through feedback. Using the best-performing model and prompt combination from Phase 2, we conduct five-iteration refinement sessions. Each iteration provides specific feedback about constraint violations and costs, requesting improvements. This entire five-iteration conversation is repeated five times to assess the consistency of the refinement process and convergence patterns.

Our iterative refinement protocol evaluates whether LLMs can improve timetabling solutions when provided with specific feedback about constraint violations. This approach mimics real-world scenarios where schedulers iteratively refine timetables based on identified conflicts and quality issues. The protocol maintains a continuous conversation thread, allowing models to build upon previous attempts while preserving context about prior violations and attempted fixes.

Each refinement session begins with the best-performing prompt configuration identified in Phase 2, generating an initial timetable attempt. After validating this solution using the automated CB-CTT validator, we provide detailed feedback to the model within the same conversation. The feedback message contains:

1) summary statistics including total hard constraint violations and soft constraint costs.
2) breakdown of violations by constraint type (H1-H4 for hard constraints, S1-S4 for soft constraints).
3) specific details about violations, such as "Too few lectures for course c0079" or "Courses c0006 and c0009 have both a lecture at period 7 (day 0, timeslot 7)".

This granular feedback enables models to understand not just that violations exist, but precisely where and why they occur.

Following feedback provision, we request improvement with a prompt such as "Based on the feedback above, please generate an improved timetable that addresses these violations while maintaining the valid assignments from your previous attempt." This phrasing encourages models to preserve correct partial solutions while fixing identified problems. The model then generates a revised timetable, which undergoes validation and receives subsequent feedback. This cycle continues for five iterations, providing models multiple opportunities to converge toward feasible solutions.

- **Phase 4: Scalability Assessment (RQ4)** compares the performance on three instances (small, medium, large) to evaluate how performance scales with problem complexity. Table 1 provides lists the characteristics of the three instances. This phase reveals whether successful strategies on smaller instances transfer to larger, more complex problems, addressing fundamental questions about LLM scalability in combinatorial optimization.

4.4. Evaluation Metrics

The quality of the generated solutions are assessed using the following metrics:

1) Success Rate (SR): The percentage of runs where LLM models generates an output in the requested format is computed via Equation (1):

$$SR = \frac{\#\ of\ runs\ with\ generated\ output}{Total\ number\ of\ runs} \times 100 \quad (1)$$

Higher success rates are better, indicating model reliability in generating responses.

2) Violations (V): The total number of hard-constraint violations is computed via Equation (2):

$$V = \sum_{h \in H} V_h \quad (2)$$

Where V_h is the number of violations of hard-constraint h, and H is the set of hard-constraints defined earlier, such as $H = \{H1, H2, H3, H4\}$. A solution with V=0 is feasible, indicating no violations of hard-constraints, and V_{H2}=1 indicates that the second hard constraint (Room occupancy) was violated, and thus meaning that there are two lectures assigned to the same room during the same period. Higher values represent more violations.

3) Feasibility Rate (FR): The percentage of feasible solutions out of the correctly generated outputs is computed via Equation (3):

$$FR = \frac{\#\ feasible\ solutions\ (V=0)}{\#\ runs\ with\ generated\ output} \times 100 \quad (3)$$

Higher feasibility rates are better, indicating model reliability in generating feasible solutions.

4) Total Cost (TC): The total cost of violating soft constraint for a generated solution is computed via Equation (4):

$$TC = \sum_{s \in S} TC_s \quad (4)$$

Where TCs is the penalty of violating soft-constraint s, and S is the set of soft-constraints, such as S={$S1$, $S2$, $S3$, $S4$} and costs are computed as follows:

- **S1**: 1 penalty point for each student above the capacity of the assigned room.
- •**S2**: 5 penalty points for each required working day that is missing.
- **S3**: 2 penalty points for every lecture that is isolated (i.e., has no adjacent lectures from the same

curriculum on that day).
- **S4**: 1 penalty point for each additional distinct room used beyond the initial assignment.

Lower total cost values are better, indicating higher solution quality (minimized soft-constrain violations).

5) Coefficient of Variation (CV): The output stability across runs for Violations (V) and Total Cost (TC) are computed via Equations (5) and (6), respectively:

$$CV_V = \frac{\sigma_V}{\mu_V} \quad (5)$$

$$CV_{TC} = \frac{\sigma_{TC}}{\mu_{TC}} \quad (6)$$

Where μ and σ denote mean and standard deviation, respectively. Lower CV values indicate consistent performance, while high CV suggests unstable model behavior across runs.

6) Improvement Rate (IR): The improvement achieved in iteration t after incorporating external feedback is computed via Equation (7):

$$IR_t = V_{t-1} - V_t \quad (7)$$

Where values greater than zero signify that the refinement improved the solution (larger positives are better), while negative values indicate deterioration.

7) Performance Degradation Score (PDS): The increase in violations when scaling from a simple to a more complex problem instance is computed via Equation (8):

$$PDS = \mu V_{complex} - \mu V_{simple} \quad (8)$$

Values greater than zero signify a degradation in performance (larger positives are worse), while negative values would indicate an improvement in performance as problem scale increases.

4.5. Validation and Results Analysis

We employed a publicly available CB-CTT validator specifically designed for the ITC-2007 competition format to ensure accurate and consistent evaluation of all generated timetables. This validator, originally developed for the competition and maintained by the timetabling research community, provides authoritative verification of both hard constraint satisfaction and soft constraint costs. Using an established validator eliminates potential implementation errors and ensures our results are directly comparable to traditional optimization approaches evaluated on the same instances.

The validation process is streamlined by requiring LLMs to output solutions directly in the validator's expected format. In our prompts, we explicitly specified the exact output structure required, providing clear examples of how to format course-period-room assignments. When models successfully attempted the problem, they generally adhered to this format, allowing direct submission to the validator without intermediate processing. This approach eliminates parsing complexity and potential errors from format conversion.

The validator automatically computes the Violations (V) and Total Cost (TC) metrics defined in Section III. For each submitted solution, it evaluates all four hard constraints (H1-H4), counting specific violations for each type. It similarly calculates all four soft constraint costs (S1-S4), applying the appropriate penalty weights: 1 point per excess student for room capacity, 5 points per missing working day, 2 points per isolated curriculum lecture, and 1 point per additional room. The validator outputs both individual constraint violations and aggregated metrics, enabling detailed analysis of where models struggle most. Additional performance metrics including Success Rate (SR), Feasibility Rate (FR), CV, Improvement Rate (IR), and Scalability Index (SI) are computed during post-processing analysis based on the validator outputs across multiple experimental runs.

Failures to produce a solution form a separate category in our analysis. In such cases, the models usually recognized the complexity of the task, often responding with phrases like "this problem is too complex to solve manually" or recommending the use of specialized optimization software or coding-based solutions. These explicit refusals to attempt the problem are recorded as unsuccessful attempts, distinguished from cases where models provided incorrect or infeasible solutions. This categorization helps differentiate between models that attempt but fail versus those that recognize their limitations upfront.

All metrics are recorded in structured logs for statistical analysis. For each experimental run, we capture: model identifier, prompt variant, iteration number (for refinement experiments), success/failure status, solution provision (attempted/refused), individual constraint violations for attempted solutions, total costs, and refusal reason if applicable. This comprehensive logging enables calculation of aggregate metrics mentioned above in addition to means and standard deviations of violations and costs.

5. Results and Discussion

This section presents the empirical results of our experiments designed to answer the four research questions.

5.1. RQ1: Model Performance and Stability

The first research question: How do different LLMs perform on CB-CTT benchmark instance in terms of feasibility, solution quality, and stability? The detailed results of the conducted experiments are available in Table 5.

Table 5. Comprehensive performance metrics of LLMs on CB-CTT COMP11 (RQ1). min vio and min cost represent the lowest number of violations and the minimum total cost obtained by the LLM across the five runs.

Model	Min Vio.	Min Cost	SR (%)	FR (%)	CV_V	Violations Avg(±SD)	Vio. Lectures	Vio. Conflicts	Vio. Avail.	Vio. Room Occ.	CVTC	Total Cost Avg (± SD)	Cost Room Cap.	Cost Min Work.	Cost Curr.Comp	Cost Room Stab.
Claude	24	525	100	0	0.87	48.8(±42.3)	23.6(±51.7)	21.0(±8.9)	0.8(±1.3)	3.4(±2.4)	0.62	628(±387.2)	382.6(±293.5)	237(±100.0)	8(±6.3)	0.4(±0.6)
Gemini	25	106	100	0	0.71	59.4(±42.4)	37.8(±54.4)	15.0(±9.1)	2.6(±2.6)	4.0(±7.9)	0.55	314.2(±171.6)	107.4(±118.2)	122(±173.3)	84.4(±58.5)	0.4(±0.9)
Grok	0	482	60	33.33	0.90	25.7(±23.2)	0.0(±0.0)	13.7(±11.9)	5.3(±4.6)	6.7(±7.6)	0.06	471.3(±29.5)	282.3(±160.3)	180(±156.6)	2(±2.0)	7(±12.1)
Mistral	69	375	60	0	0.25	85.7(±21.6)	59.0(±35.4)	8.3(±6.4)	5.7(±0.6)	12.7(±17.0)	0.55	331.7(±183.9)	0.0(±0.0)	228.3(±192.8)	102.7(±95.9)	0.7(±1.2)
OpenAI-o3	0	211	100	80	2.24	12.6(±28.2)	6.2(±13.9)	2.4(±5.4)	1.0(±2.2)	3.0(±6.7)	0.84	551.6(±463.6)	282.8(±415.0)	144(±154.5)	105.6(±94.6)	19.2(±12.5)
OpenAI-o4	3	263	60	0	0.88	93.3(±81.7)	54.0(±93.5)	36.0(±62.4)	3.3(±3.5)	0.0(±0.0)	1.07	966.7(±1032.5)	599.0(±1037.5)	335(±130.3)	6(±8.7)	26.7(±37.1)

1) Feasibility: The Ability to Satisfy Hard Constraints

The most critical measure of performance for a timetabling problem is feasibility-the ability to generate a solution that violates zero hard constraints. This must be distinguished from the success rate, which simply measures whether a model completed its generation process without errors. While several models, including Claude, Gemini, and OpenAI-o3, achieved a 100% success rate, this did not correlate with their ability to produce logically valid timetables. The results reveal a major difference among the models in this primary objective. Figure 3 demonstrates a comparison between models' feasibility rate and success rate.

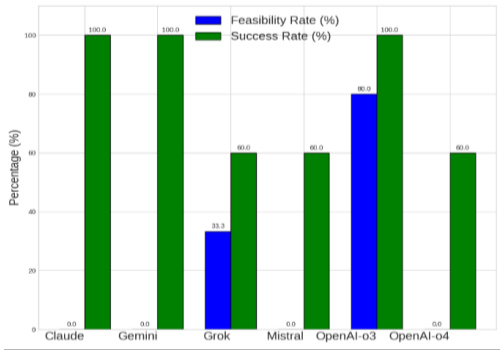

Figure 3. Comparison of model feasibility and success rates.

The OpenAI-o3 model emerged as the top performer, showing an ability to generate valid solutions. It successfully produced 4 feasible timetables out of 5 attempts, achieving an 80% feasibility rate. This stood out as a notable exception compared to the other models. The only other model to achieve valid solution was Grok, which generated 1 feasible solution in 3 attempts, resulting in a feasibility rate of 33.33%.

In contrast, the other four models-Claude, Gemini, Mistral, and OpenAI-o4-failed to produce a single feasible solution, each resulting in a 0% feasibility rate. This highlights a fundamental difficulty for these models in navigating the complex, interdependent nature of the problem's hard constraints when using a basic prompt structure. Additionally, variability in performance can be attributed to the difference in architecture and training mechanism of selected LLMs.

2) Solution Quality: Minimizing Soft Constraint Costs

Beyond feasibility, we assess the quality of solutions by the total cost incurred from soft constraint violations, where a lower cost signifies a higher-quality timetable.

Among the feasible solutions, OpenAI-o3 again demonstrated superior performance by achieving a minimum total cost of (211). The single feasible solution from Grok was of lower quality, with a total cost of (482). This indicates that OpenAI-o3 is not only more reliable at finding valid solutions but is also more capable of optimizing for the desired soft constraints. This can be attributed to differences in training and inference alignment which enable more consistent constraint-aware generation and more effective implicit optimization of soft constraints [33].

Figure 4 illustrates the trade-off between solution feasibility (x-axis, minimum hard constraint violations) and solution quality (y-axis, minimum total cost from soft constraints). The ideal position is the bottom-left corner, representing a fully feasible, high-quality solution. An interesting observation arises from the performance of the models on infeasible solutions. The Gemini model achieved the lowest minimum cost of any attempt (106), suggesting a strong capability for optimizing soft constraints. However, this was achieved at the expense of satisfying the hard constraints, rendering its solutions practically unusable. This highlights a potential trade-off where some models may prioritize quality over the essential requirement of feasibility.

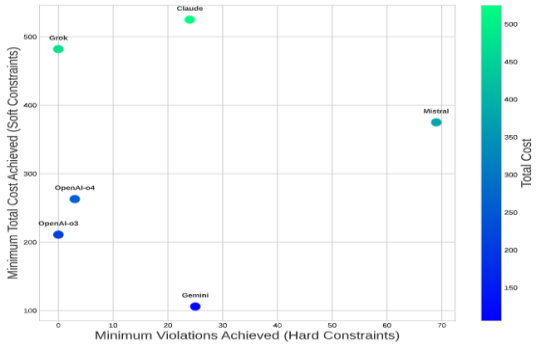

Figure 4. Solution quality analysis: minimum cost vs. minimum hard constraint violations.

3) Stability and Consistency Analysis

The stability of each model was evaluated by the CV for both the number of hard constraint violations and the total cost. Figure 5 illustrates the stability of each

model's performance across multiple runs. The CV is shown for both the number of hard constraint violations (blue) and the total solution cost (green). A lower CV indicates more consistent and predictable output.

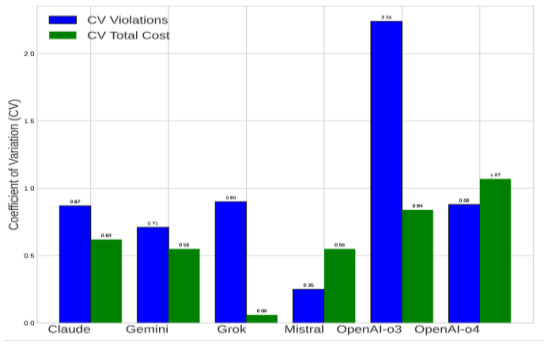

Figure 5. Model stability analysis using the coefficient of variation (CV) for hard constraint violations and total solution cost.

- Violation Stability (CV Violations): OpenAI-o3 exhibited the highest CV for violations (2.24). In this context, this high variability is a positive indicator; it reflects a mix of highly successful runs (0 violations) and unsuccessful ones, unlike other models that failed more consistently. For instance, 'Mistral' had the lowest CV (0.25), indicating that while its performance was stable, it was stably poor, consistently producing a high number of violations.
- Total Cost Stability (CV Total Cost): Grok was exceptionally stable in its solution quality, with a CV for cost of (0.06). This suggests that its attempts, whether feasible or not, consistently yielded solutions of a similar quality. In contrast, OpenAI-o4 exhibited the highest level of inconsistency, with a total cost CV of (1.07), reflecting significant fluctuations in the quality of its generated timetables.

4) Detailed Constraint Analysis

A deeper analysis of the specific constraint violations provides insight into which aspects of the problem were most challenging for the models. Figure 6 illustrates the detailed constraint performance of each model. For each model, the left stacked bar (shades of blue) corresponds to the left y-axis and shows the composition of average hard constraint violations. The right stacked bar (shades of green) corresponds to the right y-axis and shows the composition of average soft constraint costs. This visualization allows for a direct comparison of the types of errors each model is prone to making, highlighting, for example, the significant challenge of the Lectures constraint for models like Mistral and the high 'Room Capacity' cost for OpenAI-o4.

More specifically, the violation data reveals that not all hard constraints posed an equal challenge. The most difficult constraints for the models to satisfy were consistently Lectures (H1) and Conflicts (H2). Mistral and OpenAI-o4 particularly struggled with ensuring all lectures were scheduled, with average violations of 59.0 and 54.0, respectively. Claude and OpenAI-o4 had the most difficulty with conflicts, averaging 21.0 and 36.0 violations. In contrast, Availability (H4) was the easiest hard constraint for most models to adhere to, with violation counts being consistently low across the board. Notably, Grok was the only model to perfectly satisfy the Lectures constraint in its one feasible attempt.

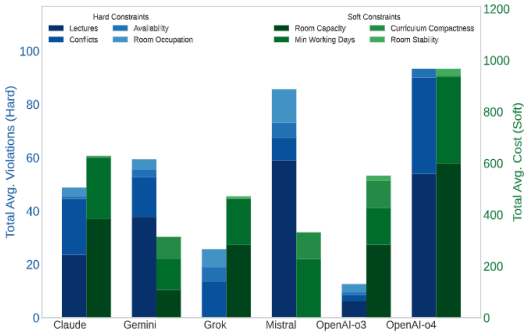

Figure 6. Distribution of average hard constraint violations (left axis) and soft constraint costs (right axis) by model.

The cost breakdown for soft constraints reveals a similar pattern of varied difficulty. The most challenging soft constraints to optimize were RoomCapacity (S1) and MinWorkingDays (S2), which consistently contributed the largest penalties to the total cost for most models. OpenAI-o4 and Claude, for example, incurred average costs of 599.0 and 382.6 for room capacity violations alone. A notable exception was Mistral, which achieved a perfect score of 0 for room capacity, suggesting a different internal strategy.

In contrast, CurriculumCompactness (S3) and RoomStability (S4) were generally easier for the models to optimize, resulting in lower average costs. Grok was particularly effective at optimizing for curriculum compactness, incurring a negligible average cost of 2.0.

To sum, the results indicate that OpenAI-o3 is demonstrably the most effective model for the CB-CTT problem among those tested. It is the only model to consistently generate feasible solutions while also proving capable of producing high-quality, low-cost timetables. While Grok shows some potential, its reliability is significantly lower.

The remaining models, in this basic configuration, are not currently viable as end-to-end solvers for this problem class due to their inability to satisfy the fundamental hard constraints. The detailed constraint analysis further suggests that the core reasoning challenge for these LLMs lies in satisfying global constraints related to completeness (scheduling all lectures) and conflict avoidance, while they are relatively more successful at handling simpler, localized constraints.

5.2. RQ2: Prompt Engineering Analysis

Our second research question determines the impact of various prompt engineering strategies on the timetabling performance of the OpenAI-o3 model. The results, summarized in Table 6, present a clear conclusion: for

this complex, globally-constrained optimization task, the most basic prompt was significantly more effective than any of the more elaborate, engineered prompts. Figure 7 illustrates this finding. The chart compares the Feasibility Rate with the Success Rate. The results clearly demonstrate that the basic prompt significantly outperforms all enhanced strategies in achieving feasibility. As illustrated in Figure 7, the basic prompt, with no additional guidance, achieved an 80% feasibility rate. In stark contrast, every attempt to enhance the prompt with additional context resulted in a degradation of performance. Feasibility rates for the engineered prompts ranged from a high of 50% down to just 20%.

Table 6. Impact of prompt engineering strategies on ChatGPT-o3 performance (RQ2).

Model/Strategy	Min Vio.	Min Cost	SR (%)	FR (%)	CV_V	Violations (Avg±SD)	Vio. Lectures	Vio. Conflicts	Vio. Avail.	Vio. Room Occ.	CVTC	Total Cost Avg (±SD)	Cost Room Cap.	Cost Min Work.	Cost Curr.Comp.	Cost Room Stab.
Basic	0	211	100	80	2.24	12.6(±28.2)	6.2(±13.9)	2.4(±5.4)	1.0(±2.2)	3.0(±6.7)	0.84	551.6(±463.6)	282.8(±415.0)	144(±154.5)	105.6(±94.6)	19.2(±12.5)
+ Expl + Toy1	0	205	80	50	1.18	35.0(±41.4)	9.5(±17.7)	11.3(±15.1)	1.5(±1.7)	12.8(±21.1)	0.29	310.3(±89.9)	45.0(±54.5)	100(±114.3)	154.0(±95.5)	11.3(±13.7)
+ Expl + Toy2	0	1196	100	20	0.70	88.0(±61.5)	39.4(±50.1)	27.2(±18.4)	6.0(±5.7)	15.4(±20.8)	0.96	1467.4(±1409.4)	1288.6(±1358.9)	51(±86.6)	112.4(±61.7)	15.4(±22.4)
+ Expl + 2 Toys	0	278	100	40	1.03	43.6(±44.7)	0.8(±1.3)	12.6(±12.0)	2.4(±2.9)	27.8(±33.3)	0.85	655.4(±555.5)	395.4(±504.0)	78(±102.1)	153.6(±91.6)	28.4(±31.4)
+ Explanation	0	464	100	40	1.01	45.8(±46.4)	7.0(±15.7)	24.4(±44.7)	1.0(±2.2)	13.4(±23.0)	0.98	955.0(±934.6)	698.0(±849.1)	143(±138.7)	75.2(±86.2)	38.8(±34.0)
+ Toy1	0	946	100	20	0.61	69.2(±42.0)	33.4(±27.6)	20.0(±16.7)	3.6(±3.4)	12.2(±18.0)	0.60	698.8(±418.9)	416.2(±416.4)	188(±118.9)	83.2(±96.4)	11.4(±20.1)
+ Toy2	0	1231	80	50	1.23	52.8(±64.9)	23.3(±40.7)	16.8(±20.3)	3.0(±3.6)	9.8(±18.8)	0.36	1071.8(±387.3)	772.8(±357.6)	216.3(±142.3)	63.5(±97.8)	19.3(±13.8)
+ Two Toys	0	255	100	40	1.27	47.0(±59.9)	5.2(±10.6)	16.6(±16.7)	3.6(±3.9)	21.6(±34.0)	0.60	256.0(±153.2)	50.4(±77.3)	83(±120.7)	112.8(±96.6)	9.8(±12.9)

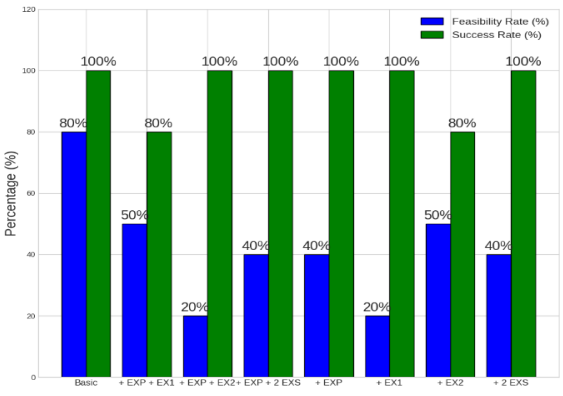

Figure 7. Comparison of feasibility and success rates across different prompt engineering strategies.

This outcome challenges the prevailing assumption in much of the prompt engineering literature that providing more context and examples should improve a model's performance on complex reasoning tasks. The results suggest that for problems with tight, interdependent constraints like CB-CTT, the opposite may be true.

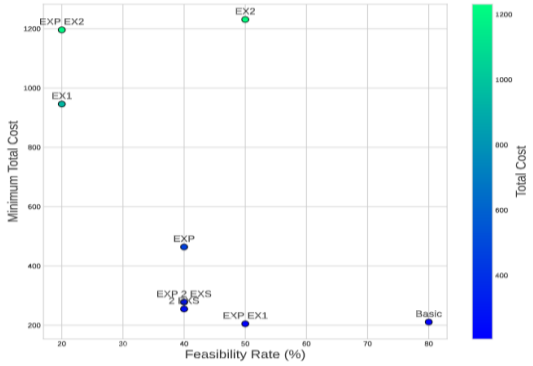

Figure 8. Feasibility rate versus minimum total cost for each prompt engineering strategy.

Although prompt engineering clearly reduced feasibility, its effect on solution quality (in terms of minimizing soft constraint costs) was more complex. Figure 8 illustrates a trade-off between achieving feasibility and optimizing solution cost. The basic prompt occupies the ideal bottom right position, indicating high feasibility. However, two engineered prompts-+Expl+Toy1 and +Two Toys-achieved minimum costs of 205 and 255, respectively. These values are highly competitive with the basic prompt's best performance of 211. This indicates that although the extra context disrupted the model's ability to consistently satisfy strict hard constraints, it effectively steered the model towards better optimizing the "desirable" soft constraints in its largely infeasible attempts.

5.3. RQ2: Prompt Engineering Analysis

The third research question investigates a critical aspect of applying LLMs to optimization: Can iterative refinement with constraint violation feedback improve solutions? The experiment was designed to simulate a generate-and-test loop where the model receives feedback on its previous attempt and is prompted to generate a corrected solution. The results of five independent runs, detailed in Table 7, reveal a process that is both powerful in its initial corrective potential and deeply flawed in its stability and reliability.

Table 7. Iterative refinement performance with violation feedback.

Run	Itr.	*V*	*TC*	IR_V	*IR_TC*
1	Basic	73	496	-	-
	feedback 1	73	325	0	171 ↑
	feedback 2	NA	NA	-	-
2	Basic	0	92	-	-
	feedback 1	6	76	-6 ↓	16 ↑
	feedback 2	0	97	6 ↑	-21 ↓
	feedback 3	0	69	0	28 ↑
	feedback 4	0	68	0	1 ↑
	feedback 5	0	64	0	4 ↑
3	Basic	128	745	-	-
	feedback 1	0	232	128 ↑	513 ↑
	feedback 2	64	155	-64 ↓	77 ↑
	feedback 3	NA	NA	-	-
4	Basic	199	1578	-	-
	feedback 1	0	127	199 ↑	1451 ↑
	feedback 2	5	126	-5 ↓	1 ↑
	feedback 3	4	140	1 ↑	-14 ↓
	feedback 4	3	142	1 ↑	-2 ↓
	feedback 5	1	136	2 ↑	6 ↑
5	Basic	NA	NA	-	-

5.4. RQ4: Scalability Assessment

The fourth research question addresses a critical aspect of practical applicability: How does the model's performance scale with problem complexity? To investigate this, we evaluated the OpenAI-o3 model, which proved most effective on the small instance, against two progressively larger and more complex problem instances: medium and large. The results-presented in Table 8 and illustrated in Figure 9-clearly show that the model struggles to scale, with performance deteriorating sharply as the combinatorial complexity of the problem grows.

Table 8. Performance scalability across different problem complexities.

Instance	Problem Size (# Courses)	Min Violations	Min Total Cost	Success Rate (%)	Feasibility Rate (%)	PDS	Violations (Avg ± SD)	Total Cost (Avg ± SD)
comp11 (small)	30	0	211	100	80	0	12.6(±28.17)	551.6(±463.55)
comp08 (medium)	86 (×3 increase)	198	5124	100	0	22.4 (×19 increase)	237(±23.65)	4059.2(±1584.47)
comp20 (large)	121 (×4 increase)	207	1837	40	0	267.4 (×22 increase)	280(±103.24)	3061(±1731.00)

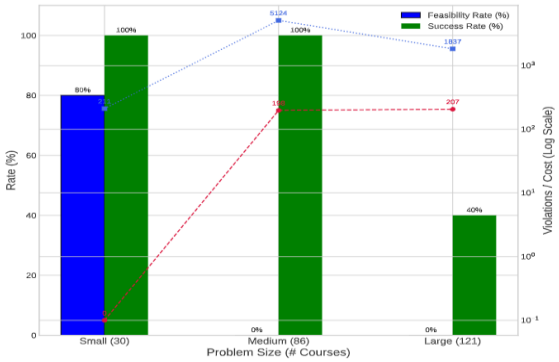

Figure 9. Scalability of performance metrics across different problem complexities.

The key finding is the model's complete failure to generate feasible solutions as problem size increases. While it achieved 80% feasibility on the small instance, performance dropped to 0% on both medium and large instances. This sharp decline highlights the model's inability to scale, as it could not handle the increased complexity and number of constraints in larger timetabling problems. This lack of scalability can be attributed primarily to architectural limitations of LLMs, which rely on next-token generation optimized for local coherence rather than systematic exploration of exponentially growing search spaces. As problem size increases, the number of interacting hard constraints grows rapidly, requiring coordinated decisions and implicit backtracking that LLMs are not designed to perform. This experiment highlighted a critical weakness of LLMs when used as end-to-end solvers for NP-hard problems such as timetabling. Although they were able to generate feasible solutions on smaller instances, they become ineffective as problem size increases.

5.5. Comparison with Standard Approaches

LLMs' performance is compared against baseline optimizers in order to provide an absolute assessment. Five classic search and optimization algorithms are selected, which are: Local Search (LS), Hill Climbing (HC), Tabu Seach (TS), Simulated Annealing (SA), and Deep Local Search (DLS). Table 9 provides a summary of their performance in terms of number of violations and total cost. The results indicate that classic algorithms outperform LLMs, as they explicitly maintain a global solution state, systematically explore the search space, and enforce hard constraints during the optimization process.

Table 9. Comparison of LLMs performance against classis methos.

	Comp11		Comp08		Comp20	
	Violations (Avg±SD)	**Total Cost Avg (±SD)**	**Violations (Avg±SD)**	**Total Cost Avg (±SD)**	**Violations (Avg±SD)**	**Total Cost Avg (±SD)**
OpenAI-o3	12(±28)	552(±463)	237(±24)	4059(±1585)	280(±103)	3061(±1731)
LS	0(±0)	40(±20)	0(±0)	165.4(±10)	0(±0)	239.4(±35)
HC	0(±0)	0(±0)	0(±0)	66.8(±7)	0(±0)	128.4(±28)
TS	0(±0)	0(±0)	0(±0)	80.2(±12)	0(±0)	126.8(±18)
SA	0(±0)	0(±0)	0(±0)	51(±2)	0(±0)	44(±3)
DLS	0(±0)	14(±9)	0(±0)	145.4(±11)	0(±0)	205.2(±23)

Nevertheless, LLMs remain attractive due to their ability to reduce the need for expert knowledge and manual problem formulation, which are typically required by traditional optimizers. These observations highlight the importance of hybrid approaches that combine the generative and heuristic strengths of LLMs with external optimization or verification mechanisms [14, 27, 33]. In this setting, LLMs can effectively support tasks such as solution initialization, heuristic guidance, and repair suggestions, while established solvers handle feasibility enforcement, backtracking, and objective optimization. As a result, integrating LLMs with classical search and optimization algorithms represents a more practical and robust approach for addressing complex, real-world scheduling challenges.

6. Conclusions

This study aimed to assess the standalone abilities of advanced, high-reasoning Large Language Models to function as end-to-end solvers for the NP-hard Curriculum-Based Course Timetabling problem.

Through experiments guided by four research questions, we present a coherent set of results that lay an essential foundation for exploring the use of LLMs in complex combinatorial optimization problems.

Our investigation revealed that while a leading model like OpenAI-o3 can solve small-scale instances with an 80% feasibility rate (RQ1), this success is inconsistent and does not generalize. Additionally, the results indicated that standard prompt engineering techniques, such as providing detailed explanations or few-shot examples, consistently degraded the model's ability to find feasible solutions (RQ2). Furthermore, while iterative feedback can trigger powerful one-shot corrections, the process is highly unstable and does not lead to reliable, convergent optimization; solutions often degrade or the process fails entirely after a few steps (RQ3). Most notably, the model's performance deteriorates drastically with increasing problem complexity-feasibility drops to zero, operational failures become common, and the results clearly show a fundamental lack of scalability (RQ4).

The implications of these findings are twofold. First, the results serve as a strong cautionary note: LLMs, in their current state, should not be considered reliable, autonomous solvers for high-stakes, real-world scheduling problems. Their lack of scalability and the unpredictability of their reasoning process make them unsuitable for production environments where correctness and robustness are critical. Second, an interesting direction for future work is to integrate LLMs with traditional solvers in a hybrid approach. In such a system, the LLM could act as a natural language front-end or a heuristic generator, leveraging its strengths while offloading the critical task of ensuring logical consistency and feasibility to a symbolic component designed for that purpose.

Acknowledgment

The author would like to acknowledge the use of generative language models for paraphrasing and proofreading sections of this manuscript. This technology assisted in improving the clarity and flow of the writing while we ensured scientific accuracy.

References

[1] Abgaryan H., Harutyunyan A., and Cazenave T., "LLMs Can Schedule," *arXiv preprint*, pp. 1-12, 2024. https://doi.org/10.48550/arXiv.2408.06993

[2] Akkan C., Gülcü A., and Kus Z., "Bi-Criteria Simulated Annealing for The Curriculum-Based Course Timetabling Problem with Robustness Approximation," *Journal of Scheduling*, vol. 25, no. 4, pp. 477-501, 2022. DOI:10.1007/s10951-022-00722-0

[3] Alonge O., Sakpere W., and Adediran E., "An Automated Timetable Scheduler Using Nsga II for Optimized Scheduling in Educational Institutions," *Advance Journal of Science, Engineering and Technology*, vol. 10, no. 4, pp. 79-97, 2025. https://aspjournals.net/ajset/index.php/ajset/article/view/130

[4] Atanda O., Adebiyi M., Lawrence M., and Adeniyi E., "Enhancing Timetable Scheduling Efficiency with a Firefly Algorithm-Based Automated System: A Case Study of Landmark University," *in Proceedings of the 3rd International Conference on Advancement in Computation and Computer Technologies*, Punjab, pp. 78-82, 2025. DOI:10.1109/InCACCT65424.2025.11011428

[5] Awad F., Al-Kubaisi A., and Mahmood M., "Large-Scale Timetabling Problems with Adaptive Tabu Search," *Journal of Intelligent Systems*, vol. 31, no. 1, pp. 168-176, 2022. DOI:10.1515/jisys-2022-0003

[6] Bashab A., Ibrahim A., Tarigo Hashem I., and Aggarwal K., "Optimization Techniques in University Timetabling Problem: Constraints, Methodologies, Benchmarks, And Open Issues," *Computers, Materials and Continua*, vol. 74, no. 3, pp. 6461-6484, 2023. DOI:10.32604/cmc.2023.034051

[7] Bettinelli A., Cacchiani V., Roberti R., and Toth P., "An Overview of Curriculum-based Course Timetabling," *TOP*, vol. 23, no. 2, pp. 313-349, 2015. DOI:10.1007/s11750-015-0366-z

[8] Burke E., Mareˇcek J., Parkes A., and Rudová H., "A Branch-And-Cut Procedure for the Udine Course Timetabling Problem," *Annals of Operations Research*, vol. 194, no. 1, pp. 71-87, 2012. DOI:10.1007/s10479-010-0828-5

[9] Ceschia S., Di Gaspero L., and Schaerf A., "Design, Engineering, and Experimental Analysis of a Simulated Annealing Approach to The Post-Enrolment Course Timetabling Problem," *Computers and Operations Research*, vol. 39, no. 7, pp. 1615-1624, 2012. https://doi.org/10.1016/j.cor.2011.09.014

[10] Chen M., Sze S., Goh S., Sabar N., and Kendall G., "A Survey of University Course Timetabling Problem: Perspectives, Trends and Opportunities," *IEEE Access*, vol. 9, pp. 106515-106529, 2021. DOI:10.1109/ACCESS.2021.3100613

[11] Di Gaspero L., McCollum B., and Schaerf A., "The Second International Timetabling Competition (itc-2007): Curriculum-Based Course Timetabling (track 3)," *in Proceedings of the 1st International Workshop on Scheduling a Scheduling Competition*, Rhode Island, pp. 1-4, 2007. https://api.semanticscholar.org/CorpusID:5872498

[12] Fan L., Hua W., Li L., Ling H., and Zhang Y., "Nphardeval: Dynamic Benchmark on Reasoning

Ability of Large Language Models Via Complexity Classes," *arXiv preprint*, vol. arXiv:2312.14890, pp. 1-23, 2023. https://doi.org/10.48550/arXiv.2312.14890

[13] Fonseca G., Santos H., Carrano E., and Stidsen T., "Integer Programming Techniques for Educational Timetabling," *European Journal of Operational Research*, vol. 262, no. 1, pp. 28-39, 2017. DOI:10.1016/j.ejor.2017.03.020

[14] Giannoulis P., Yorgos P., and Tzamos C., "Teaching Transformers to Solve Combinatorial Problems through Efficient Trial and Error." *arXiv Preprint*, *vol. arXiv:2509.22023*, pp. 1-33, 2025. https://doi.org/10.48550/arXiv.2509.22023

[15] Gu X., Krish M., Sohail S., Thakur S., and et al, "From Integer Programming to Machine Learning: A Technical Review on Solving University Timetabling Problems," *Computation*, vol. 13, no. 1, pp. 1-27, 2025. https://doi.org/10.3390/computation13010010

[16] Hafsa M., Wattebled P., Jacques J., and Jourdan L., "Solving a Multi-Objective Professional Timetabling Problem Using Evolutionary Algorithms at Mandarine Academy," *International Transactions in Operational Research*, vol. 32, no. 1, pp. 244-269, 2025. https://doi.org/10.1111/itor.13276

[17] Li T., Chiang W., Frick E., Dunlap L., and et al, "From Crowdsourced Data to High-Quality Benchmarks: Arena-Hard and Benchbuilder Pipeline," *in Proceedings of the 42nd International Conference on Machine Learning*, pp. 34209-34231, 2024.

[18] Lin B., Le Bras R., Richardson K., Sabharwal A., and et al, "Zebralogic: On The Scaling Limits of LLMs for Logical Reasoning," *in Proceedings of the 42nd International Conference on Machine Learning*, pp. 1-17, Vancouver, 2025. https://hf.co/spaces/WildEval/ZebraLogic

[19] Liu B., Bubeck S., Eldan R., and Kulkarni J., "TinyGSM: Achieving> 80% on Gsm8k with Small Language Models," *arXiv preprint*, vol. arXiv:2312.09241, pp. 1-15, 2023. https://doi.org/10.48550/arXiv.2312.09241

[20] Lu Z. and Hao J., "Adaptive Tabu Search for Course Timetabling," *European Journal of Operational Research*, vol. 200, no. 1, pp. 235-244, 2010. DOI:10.1016/j.ejor.2008.12.007

[21] McCollum B. and Ireland N., "University Timetabling: Bridging The Gap Between Research and Practice," *in Proceedings of the Practice and Theory of Automated Timetabling Conference*, Brno, pp. 15-35, 2006. https://patatconference.org/patat2006/proceedings/1_2.pdf

[22] Phan L., Gatti A., Han Z., Li N., and et al, "Humanity's Last Exam," *arXiv preprint*, vol. arXiv:2501.14249, pp. 1-29, 2025. https://doi.org/10.48550/arXiv.2501.14249

[23] Premananda I., Tjahyanto A., and Muklason A., "Timetabling Problems and the Effort Towards Generic Algorithms: A Comprehensive Survey," *IEEE Access*, vol. 12, pp. 143854-143868, 2024. DOI:10.1109/ACCESS.2024.3463721

[24] Qintong Li., Cui L., Zhao X., and Kong L., "GSM-Plus: A Comprehensive Benchmark for Evaluating the Robustness of LLMs as Mathematical Problem Solvers," *in Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). Association for Computational Linguistics*, Bangkok, pp. 2961-2984, 2024. DOI:10.18653/v1/2024.acl-long.163

[25] Rappos E., Thiémard E., Robert S., and Hêche J-F., "A Mixed-Integer Programming Approach for Solving University Course Timetabling Problems," *Journal of Scheduling*, vol. 25, no. 4, pp. 391-404, 2022. DOI:10.1007/s10951-021-00715-5

[26] Respati R. and Ramadhani D., "Optimizing Lecture Scheduling Using Genetic Algorithm: A Case Study at Universitas Riau," *International Journal of Electrical, Energy and Power System Engineering*, vol. 8, no. 2, pp. 220-232, 2025. DOI:10.31258/ijeepse.8.2.220-232

[27] Selvaraj A., R. Prabakaran, and Kanimozhi R., "A Novel Resource Scheduler for Resource Allocation and Scheduling in Big Data Using Hybrid Optimization Algorithm at Cloud Environment," *The International Arab Journal of Information Technology*, vol. 20, no. 6, pp. 863-873, 2023. https://doi.org/10.34028/iajit/20/6/3

[28] Wang Y., Ma X., Zhang G., Ni Y., and et al, "MMLU-Pro: A More Robust and Challenging Multi-Task Language Understanding Benchmark," *in Proceedings of the 38th International Conference on Neural Information Processing Systems*, Vancouver, pp. 95266-95290, 2024. DOI:10.52202/079017-3018

[29] Xiao Y., Li X., Jiang L., and Wang P., "Towards Dynamic University Course Timetabling Problem: An Automated Approach Augmented Via Reinforcement Learning," *in Proceedings of The IEEE International Conference on Data Mining*, Abu Dhabi, pp. 520-529, 2024. doi:10.1109/ICDM59182.2024.00059

[30] Yuksekgonul M., Chandrasekaran V., Jones E., Gunasekar S., and et al, "Attention Satisfies: A Constraint-Satisfaction Lens on Factual Errors of Language Models," *in Proceedings of the 12th International Conference on Learning Representations*, Vienna, pp. 1-25, 2024. https://openreview.net/pdf?id=gfFVATffPd

[31] Zhang H., Da J., Lee D., Robinson V., and et al, "A Careful Examination of Large Language Model Performance on Grade School Arithmetic,"

in Proceedings of the 38th International Conference on Neural Information Processing System, New York, pp. 46819-46836, 2024. https://dl.acm.org/doi/10.5555/3737916.3739401

[32] Zhou J., Lu T., Mishra S., and Brahma S., "Instruction-Following Evaluation for Large Language Models," *arXiv preprint*, vol. arXiv:2311.07911v1, pp. 1-43, 2023. https://doi.org/10.48550/arXiv.2311.07911

[33] Zhou Y., Ye J., Ling Z., Han Y., and et al, "Dissecting Logical Reasoning in LLMs: A Fine-Grained Evaluation and Supervision Study," *arXiv preprint*, vol. arXiv:2506.04810, pp. 1-24, 2025. DOI:10.48550/arXiv.2506.04810

Basma Alharbi is an Associate Professor of Computer Science and Artificial Intelligence, at the department of Computer Science and Artificial Intelligence, College of Computer Science and Engineering, University of Jeddah, Jeddah, Saudi Arabia.