Reasoning with Preference Constraints: A Benchmark for Language Models in Many-to-One Matching Markets

Marylou Fauchard
Université de Montréal, Mila
marylou.fauchard@mila.quebec

Florian Carichon
McGill, Mila
florian.carichon@mila.quebec

Margarida Carvalho
Université de Montréal, Mila
carvalho@iro.umontreal.ca

Golnoosh Farnadi
McGill, Mila
farnadig@mila.quebec

Abstract

Recent advances in reasoning with large language models (LLMs) have demonstrated strong performance on complex mathematical tasks, including combinatorial optimization. Techniques such as Chain-of-Thought and In-Context Learning have further enhanced this capability, making LLMs both powerful and accessible tools for a wide range of users, including non-experts. However, applying LLMs to matching problems, which require reasoning under preferential and structural constraints, remains underexplored. To address this gap, we introduce a novel benchmark of 369 instances of the College Admission Problem, a canonical example of a matching problem with preferences, to evaluate LLMs across key dimensions: feasibility, stability, and optimality. We employ this benchmark to assess the performance of several open-weight LLMs. Our results first reveal that while LLMs can satisfy certain constraints, they struggle to meet all evaluation criteria consistently. They also show that reasoning LLMs, like QwQ and GPT-oss, significantly outperform traditional models such as Llama, Qwen or Mistral, defined here as models used without any dedicated reasoning mechanisms. Moreover, we observed that LLMs reacted differently to the various prompting strategies tested, which include Chain-of-Thought, In-Context Learning and role-based prompting, with no prompt consistently offering the best performance. Finally, we report the performances from iterative prompting with auto-generated feedback and show that they are not monotonic; they can peak early and then significantly decline in later attempts. Overall, this work offers a new perspective on model reasoning performance and the effectiveness of prompting strategies in combinatorial optimization problems with preferential constraints.

1 Introduction

Recent advancements in large language models (LLMs) have revealed emergent reasoning capabilities, enabling them to address increasingly complex problems. Most notably, LLMs have been applied to solve mathematical problems that require step-by-step analytical reasoning [5]. They have also been used in complex game-theoretic scenarios with interactions and strategic decision-making [42] and combinatorial optimization tasks [49, 45], proving that LLMs can extend their reasoning abilities to various kind of complex real-world scenarios. To rigorously assess the effectiveness of LLM reasoning and its associated methods, a range of benchmarks and evaluation metrics have been

39th Conference on Neural Information Processing Systems (NeurIPS 2025) Workshop: Foundations of Reasoning in Language Models.

developed. These benchmarks can be categorized into (1) objective, ground-truth-based benchmarks such as MATH [21] or SCIBENCH [46] and (2) subjective, preference-based evaluations such as Chatbot Arena [13]. Although these benchmarks are valuable for understanding LLMs' capacity to solve these theoretical problems, they are still limited due to static question sets, the risk of data contamination, a lack of real-world complexity, high variance, or prioritizing form over reasoning quality [11]. Combinatorial optimization benchmarks such as HeuriGym [11], CO-Bench [43], or FrontierCO [17] provide ideal evaluation settings as they involve multi-step and iterative reasoning over the large solution spaces of NP-hard problems, resist memorization well, and offer robust quantitative metrics [39, 11]. While these benchmarks show that LLMs hold promise in solving NP-hard problems with cost-minimization objectives, they overlook a critical class of objective combinatorial problems involving preference-based constraints, where stability among multiple agents is the primary measure of success.

Many-to-one matching problems, which are solvable in polynomial time and involve satisfying preferences and capacity constraints, provide a natural benchmark with key properties for testing this aspect of LLMs' reasoning abilities. First, non-experts often address this optimization problem in several contexts, such as College Admission [39], Hospital-Resident matching [38], or School Choice [44]. These problems' model has been widely adopted by institutions to represent participants' preferences, such as in the US [1, 2, 38], Hungary [7], and Germany [10]. Secondly, the problem involves providing a matching, notion formally defined in Section 3.1, that satisfies desirable properties regarding the students' and colleges' preferences, introducing an additional layer of reasoning complexity. Specifically, the objective is to find a matching that minimizes the sum of ranks for students, such that no college exceeds its capacity and that no student-college pair prefers to be matched to each other over their assigned outcomes. Such a solution is considered student-optimal, feasible, and stable [18]. In the remainder of this article, we will specifically address the College Admission problem. Our contributions are the following:

- We introduce a new benchmark dataset with objective metrics specifically designed for preferential constraint problems, namely the College Admission problem.
- We conduct an empirical evaluation of several open-source LLMs, including models from Llama, Qwen, Mistral and GPT, under various prompting strategy which vary in training paradigms and model sizes.
- By analyzing the feasibility, stability, and optimality of the solutions generated by LLMs, we investigate how various dimensions of problem complexity, given by the number of students and their preferences, influence their reasoning capabilities.
- We demonstrate how several prompting approaches, such as Chain-of-Thoughts (CoT), In-Context Learning (ICL), or iterative prompting, could influence the reasoning of LLMs in solving matching problems.

2 Related Word

This work builds on two key areas of related research: two-sided many-to-one matching markets and recent developments and benchmarks for LLMs' reasoning.

2.1 Two-Sided Many-to-One Matching Markets

Matching markets are mechanisms designed to pair agents on two sides of a market based on their preferences. A foundational model is the one-to-one matching problem, famously formalized by Gale and Shapley in the context of the Stable Marriage Problem [18], where each participant on one side is matched with exactly one on the other. Many real-world applications, however, fall under the many-to-one setting. This setting mainly allows an unequal number of agents on both sides, with different capacities making the problem more challenging. The Deferred Acceptance (DA) algorithm, introduced by Gale and Shapley [18], has become a foundational tool for solving both one-to-one and, its generalization, many-to-one matching problems. The resulting matching is guaranteed to be feasible, stable, and student-optimal, notions that will be formally defined in Section 3.1. This algorithm has been implemented in several real-world settings [1, 2, 38, 7]. They demonstrate that a diversity of institutional constraints, such as stability or equitable access, exists in the College Admission problem, and more generally in many-to-one problems. They emphasize the dimensions that can be evaluated in LLMs complex reasoning. Recent work has focused on extensions and generalizations of these matching problems by introducing additional layers of complexity, such as allowing ties in preferences and enforcing common quotas [54], considering flexible capacities [9, 8, 3] or contingent priorities [37]. While existing datasets, such as the Chilean school choice dataset [14], provide valuable real-world data, they do not offer sufficient control over problem instance parameters to effectively evaluate an LLM's reasoning and understanding.

2.2 LLMs Reasoning Benchmarks

LLMs have recently demonstrated strong capabilities to tackle complex reasoning tasks in various fields [35]. In the context of mathematical reasoning, LLMs are primarily evaluated for their capabilities in logical analysis, deductive reasoning, and arithmetic operations [51]. Standard benchmarks such as MATH [21], Olympiad-Bench [20], or MathBench [30] have become central in evaluating reasoning capabilities of LLMs. Models have achieved high performances on traditional benchmarks, raising concerns about memorization instead of true reasoning capabilities with multi-step deduction, calling for a new perturbed dataset [24]. Alongside these benchmarks, combinatorial optimization problems offer a rich ground for evaluating complex reasoning. They exhibit specific characteristics: they often have numerous potential solutions that are not always directly comparable, involve different constraints on variables, and the optimization process can encounter multiple local optima [39]. Within this framework, LLMs have been tested for heuristic generation [50], algorithm design [39], and direct problem-solving via prompt-based techniques [19, 49]. The first existing benchmarks focus on the classical and potentially overused Traveling Salesman Problem (TSP) [39, 50, 49]. Other benchmarks mainly assess reasoning through real-world scenarios based on graph and set optimization problems [45, 17], pairing and scheduling problems [11, 43], or machine learning related problems such as logistic regression or grid search optimization [19, 49]. Most of these benchmarks predominantly target well-known NP-hard combinatorial problems centered on cost minimization heuristics. Therefore, they neglect structured tasks, such as problems with preference-based constraints and stability requirements. A first step toward addressing constraint-driven problems has been explored with the Transportation and Assignment Problems [28]. However, although the underlying mathematical formulations support many-to-one matchings, the prompt design appears to restrict the problem to one-to-one instances. Recent work has also studied LLMs understanding of one-on-one matching market with rigorous evaluation [23], but the many-to-one market offers a better testbed for evaluating LLM reasoning with more complexity due to potentially unmatched agents and respecting complex capacity constraints. This article introduces a novel benchmark for evaluating LLMs on the College Admission problem. By implementing and comparing several prompting strategies, our benchmark not only exposes some current limitations of LLMs in tackling complex combinatorial optimization tasks but also reveals interesting dynamics for evaluating reasoning complexity and model understanding in structured decision-making contexts.

3 Methodology

3.1 Problem Statement

In the College Admission problem, we have a set of students $\mathcal{S}$ and a set of colleges $\mathcal{C}$. Each student in $s \in \mathcal{S}$ has a strict preference order $\succ_s$ over $\mathcal{C} \cup \{\emptyset\}$. We write $c_i \succ_s c_j$ for $c_i, c_j \in \mathcal{C}$ if student s prefers college c_i over c_j. If student s prefers to be unassigned than being matched with a college $c \in \mathcal{C}$, we write $\emptyset \succ_s c$. Therefore, the student's preference list results in a ranking of the colleges. We denote by $rank_s(c)$ the rank of a college $c \in \mathcal{C}$ in the preference list of student s, e.g., the most preferred college for student s has rank 1. Similarly, each college $c \in \mathcal{C}$ has a strict order $\succ_c$ over $\mathcal{S} \cup \{\emptyset\}$. Moreover, a college c has a capacity $q_c \in \mathbb{Z}_+$. We denote $\mathcal{E} \subseteq \mathcal{S} \times (\mathcal{C} \cup \{\emptyset\})$ as the set of acceptable pairs, meaning that $(s, c) \in \mathcal{E}$ if $c \succ_s \emptyset$ and $s \succ_c \emptyset$.

An assignment is any subset $\mathcal{M} \subseteq \mathcal{E}$, interpreted as any set of student–college pairs. An assignment does not necessarily satisfy capacity constraints. An assignment $\mathcal{M}$ is *assignment stable* if for no pair $(s, c) \in \mathcal{E} \setminus \mathcal{M}$, (i) student s prefers c over their current matching, i.e., $c \succ_s \mathcal{M}(s)$ where $\mathcal{M}(s)$ is either the school assigned to student s or $\emptyset$ when the student is unmatched, (ii) school c is under-subscribed, i.e., $|\mathcal{M}(c)| < q_c$ where $\mathcal{M}(c)$ is the set of students assigned to c, or prefers student s over some student $s' \in \mathcal{M}(c)$, i.e., $s \succ_c s'$. A pair that would respect either of the previous conditions is called a blocking pair and prevents stability.

A set $\mu \subseteq \mathcal{E}$ is called a *feasible matching* when (i) each student $s \in \mathcal{S}$ is matched to at most one school, i.e., $|\{c \in \mathcal{C} : (s,c) \in \mu\}| \leq 1$, and (ii) each school $c \in \mathcal{C}$ is matched to no more students than its capacity, i.e., $|\mu(c) = \{s \in \mathcal{S} : (s,c) \in \mu\}| \leq q_c$. For clarity, we will refer to feasible matchings as "matchings" in the remainder of the paper. A matching μ is *matching stable* if for no pair $(s,c) \in \mathcal{E} \setminus \mu$, (i) student s prefers c over their current matching, i.e., $c \succ_s \mu(s)$ where $\mu(s)$ is either the school assigned to student s or $\emptyset$ when the student is unmatched, (ii) school c is under-subscribed, i.e., $|\mu(c)| < q_c$ where $\mu(c)$ is the set of students assigned to c, or prefers student s over some student $s' \in \mu(c)$, i.e., $s \succ_c s'$. By default, throughout the paper, the term stability will refer to matching stability. Finally, a stable matching μ is *student-optimal* when no student can get a better assignment in any other stable matching. It is known that a student-optimal stable matching μ coincides with the stable matching minimizing the sum of the ranks of the students, i.e., $\sum_{(s,c)\in\mu} rank_s(c)$. With these formalizations, we define our College Admission problem as computing the student-optimal stable matching.

3.2 Benchmark

We introduce a new benchmark dataset explicitly designed for controlled experimentation. Compared to the Chilean dataset [14], which focuses on large-scale problems, our benchmark provides a controlled framework over instance parameters, enabling systematic evaluation on well-defined problem settings. This controlled environment enables detailed analysis of model reasoning and decision-making processes. Additionally, the code to generate the benchmarks data[1] allows users to scale instance complexity by adjusting the number of students, colleges, and other parameters to accommodate broader generalization experiments.

In this article, we generate synthetic instances of the College Admission problem, varying the number of students across four levels: 5, 10, 15, and 20. While the number of students is limited in this setting, our benchmark is designed to be scalable, allowing easy extension to real-world sizes while remaining compatible with other experimental parameters. Each configuration is tested under three types of student preference structures: **Complete Preferences** where each student ranks all available colleges; **Incomplete Preferences** where each student ranks a fixed number of colleges, strictly fewer than the total available; and **Flexible Preferences** where the length of each student's preference list is drawn randomly between a predefined minimum and the total number of colleges. Since college preferences do not impact the complexity of the DA algorithm, they will always remain complete. Moreover, we vary the total capacity of the system across three settings: **Under-capacity** with a capacity of 80% of the number of students, forcing some students to be unassigned; **Exact-capacity** where the total capacity equals the number of students, and **Over-capacity** where total capacity exceeds the number of seats by 10. We want to test this configuration since adding capacity to certain colleges should not impact the solution, assuming that preferences remain unchanged, if the model reasons correctly. Colleges are assigned random capacities, ensuring that each college has at least one available seat. We also explore different student-to-college ratios (1:1, 2:1, 3:1, 4:1), with some configurations omitted when the total capacity falls below the number of colleges. For each combination of parameters, we generate three random seeds while creating the preferences to account for variability. In total, our benchmark comprises 369 instances, with some configurations omitted due to infeasibility. A more detailed breakdown of the number of instances is included in Section A.

3.3 Prompts

Prompt strategy with distinct formatting significantly influence LLM performance [33, 31]. To rigorously evaluate LLM performance on this reasoning task, our 369 instances are augmented with various prompt variations. We evaluate the effect of advanced prompting strategies for LLMs' reasoning, such as CoT prompting [12], ICL [15], and role-based prompting [39]. More specifically, the prompt containing only the essential information is referred to as the **Basic** prompt. The **Role** prompt follows established prompt templates from prior work [39]. This component is typically placed at the beginning of the prompt to instruct the model on the intended perspective or behavior [33]. Two of the prompting strategies fall under the category of ICL. The example is included immediately following the output format component, as commonly adopted in prior work [33]. The first one, simply named the **ICL**, consists of adding a simplified problem instance along with its corresponding student-optimal matching. To maintain the problem's core structure while adopting

[1] Github Code and Dataset Link

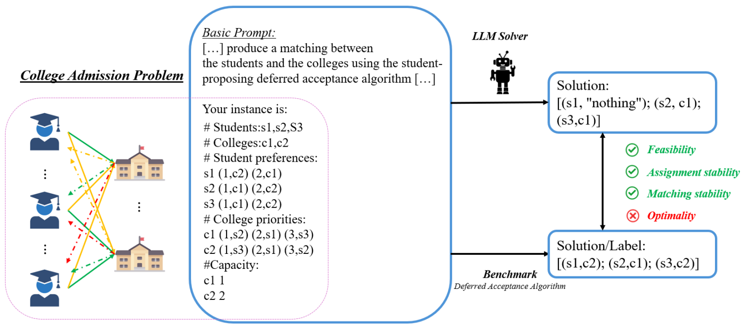


Figure 1: Experimental protocol for generating solutions to the College Admission Problem. We fed the LLM one instance associated with one prompt strategy. We compare the solution to the one of the DA algorithm to measure the feasibility, stability and optimality of the LLM-generated answer.

a simple example, we use a five-student instance with consistent student-to-school ratios, capacity types, and preference types across different random seeds. The other strategy, referred to as **ICL with steps** in the remainder of the paper, builds upon the findings of [47], which suggest that incorporating intermediate reasoning steps into examples can enhance model performance. This strategy is implemented using a fixed example with five students, complete preference lists, and exact capacity settings, where intermediate steps correspond to those of the DA algorithm. The last four prompting strategies are variations of CoT prompting, which we categorize as **CoT unsupervised**, **CoT text**, **CoT pseudocode**, and **CoT Python**. The CoT unsupervised represents the traditional unguided *Think step-by-step* instruction [52]. In contrast, the CoT text variant follows a supervised template that includes an explanation of the solution process [52]. The step-by-step explanation is based on a *natural language* description of the DA algorithm introduced in [6]. Incorporating pseudocode instructions can also improve performance in graph reasoning tasks [40]. To investigate whether similar benefits extend to matching problems, we include a pseudocode variation with a description guiding the step-by-step process. Finally, CoT prompting with Python-based, self-describing programs enhances performance on mathematical reasoning tasks [26]. Accordingly, our CoT Python template includes an implementation of the DA algorithm written in Python.

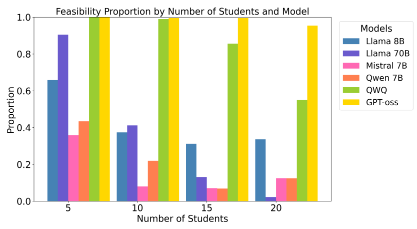
Iterative Prompting is a multi-step process that iteratively tries to improve LLMs' output by incorporating feedback on the previous failed attempts [32]. Prior works have shown promising results from this method that requires no additional training [22, 49]. More specifically, starting from the original prompt, this technique consists of adding the last model response and feedback to the preceding prompt to improve the following generated answer, up to a predefined maximum of N attempts [29]. While much of the work focuses on AI-generated feedback, we use external feedback, which is non-AI-generated, as it is more reliable [41]. In our setting, we test iterative prompting, with up to 5 iterations, adding feedback based on whether the previous output verifies our four metrics.

More details, including the structure of each prompt are available in Appendix A.

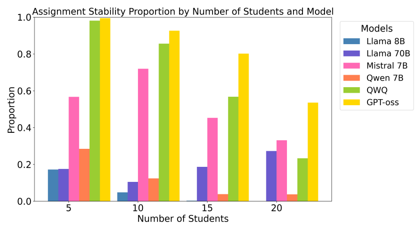
4 Empirical Evaluation

We evaluate performance across the following models: (1) **Llama 3 8B** [16] (2) **Llama 70B** [16] (3) **Mistral 7B** [25], (4) **Qwen2 7B** [48], (4) **QwQ 32B** [36], and (5) **GPT-oss 120B** [4]. We have selected these open-weight models for transparency and reproducibility purposes, ensuring that we evaluate different reasoning abilities on complex problems. While the first four models are said to be able to solve complex reasoning problems, the latter have been trained for thinking and reasoning using reinforcement learning and supervised fine-tuning. Therefore, for the remainder of the paper, we will refer to the first four models as the base models and the last two as the reasoning models. We specifically aim to assess their reasoning capabilities in solving our College Admission problem and analyze the impact of model size, model training, and problem complexity on solution quality. Figure 1 introduces our experimental protocol for generating solutions using various LLM models.

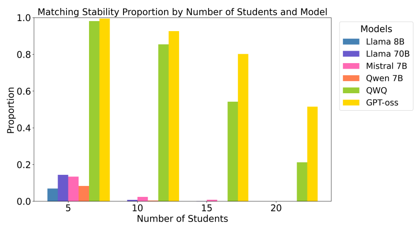
To evaluate the quality of the generated matchings, we examine four key properties: feasibility, assignment stability, matching stability, and student optimality, which are formally introduced in



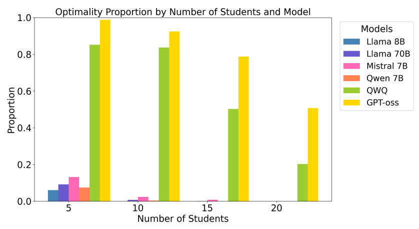
(a) Feasibility



(b) Assignment Stability

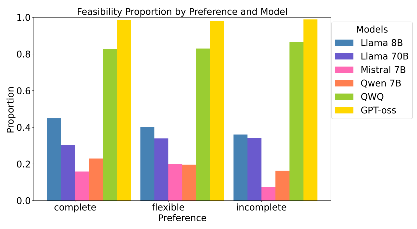


(c) Matching Stability

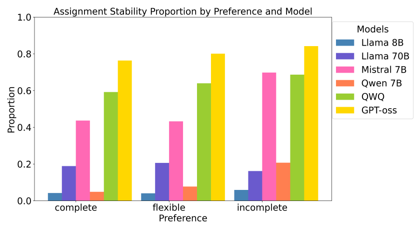


(d) Optimality

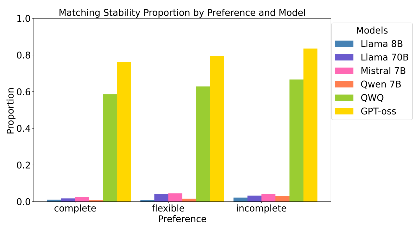
Figure 2: Proportion metrics by number of students and models. Results are aggregated over all prompts.



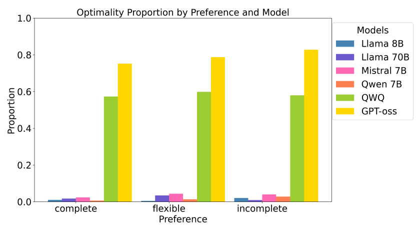
(a) Feasibility



(b) Assignment Stability



(c) Matching Stability

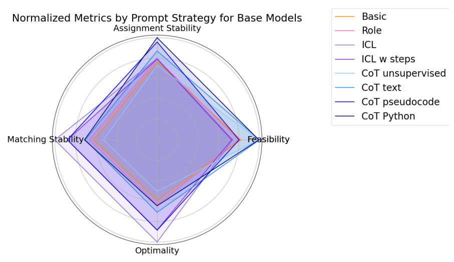


(d) Optimality

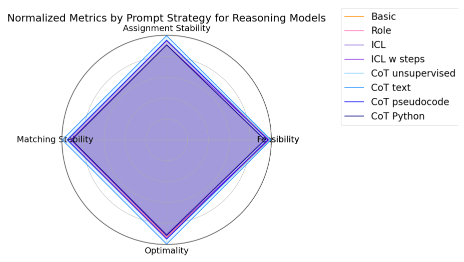
Figure 3: Normalized performance by preference for all models.

Section 3.1. Since feasibility only consists of respecting the capacities explicitly mentioned in the instance and assignment stability requires understanding the concept of blocking pairs and applying it to the given preferences, we consider that the various metrics introduced: (1a) feasibility, (1b) assignment stability, (2) matching stability, and (3) student-optimality form an ascending hierarchy of difficulty, as each level presupposes the satisfaction of the previous one. A response is considered valid if it adheres to the expected output format of the DA algorithm, allowing for minor acceptable variations such as using parentheses instead of brackets or different separators between pairs. Table 2 presents the percentage of valid outputs. Considering only the valid outputs, we compute the proportion of matchings that satisfy each metric. The dataset and the code are available on our GitHub page in footnote 1.

Our evaluation is guided by the following research questions:



(a) Performance of Prompts in Base Models



(b) Performance of Prompts in Reasoning Models

Figure 4: Performance by Model Types and Performance by Prompting Styles.

RQ1: How well do LLM-generated matchings satisfy feasibility, stability, and student-optimality? Figure 2 presents the proportion of matchings, for each model, that satisfy each metric: feasibility (Figure 2a), assignment stability (Figure 2b), matching stability (Figure 2c), and optimality (Figure 2d) based on the number of students. The results show that the matching problem is a challenging task, where base LLMs struggle even for the most explicit metric. The reasoning LLMs largely outperform the base ones on every metric and across instance sizes, highlighting the need for specialized training in reasoning. For all LLMs, performance decreases as instance size increases; however, advanced reasoning models exhibit a less abrupt drop in feasibility, indicating they can maintain track of such a simple and explicit metric. For stability and optimality, which requires understanding and preventing the notion of blocking pairs, the task is more challenging even for reasoning models. While the DA algorithm's complexity depends on the number of students, it also depends on the length of the students' preference lists. Figure 3 displays the metric distributions across different preference types and models. Figure 16 in the Appendix C presents the metrics for every prompt strategies instead of models, where the following conclusions can still be drawn. Complete preferences increase the algorithmic complexity by increasing the number of iterations required for an algorithm to arrive at the optimal solution. In contrast, incomplete preferences present more reasoning complexity, as they break completeness, creating more invalid pairs and requiring the algorithm to reason about acceptability and the presence of unmatched agents. In other words, incomplete preferences inherently increase the number of invalid pairs (i.e., the agent would rather remain unmatched than be assigned to a less preferred option). A matching that contains one invalid pair will always be unstable. However, the results suggest that the models do not struggle with this aspect. Performance remains almost unchanged with varying preference types, unlike the number of students, with a slight advantage for incomplete preferences.

RQ2: How does the prompting strategy and model choice affect the quality of the generated matchings? We previously mentioned that reasoning LLMs' performance dominates that of the base LLMs, especially when dealing with more complex instances and metrics. Although this shows that specialized training for reasoning is needed to solve even a polynomial algorithm, Figure 2b and Figure 3b reveal that Mistral can better apprehend the concept of stability with similar performances as the reasoning LLMs, but failed to perform well on the subsequent metrics because it can not satisfy feasibility as well.

In this paper, we tested multiple prompt strategies known to improve reasoning. In the Appendix C, Table 4 presents the metrics for each model and prompt. For the base models, the gap for a metric between two prompts can reach up to 40%. While reasoning models seem less impacted by the choice of prompt, hinting that their reasoning is advanced enough that they do not need to rely on CoT or ICL, the difference between strategies becomes more pronounced when looking for more challenging instances. Interestingly enough, Figure 4 shows that for the base models, the best prompts, mainly the ICL ones and CoT with pseudocode, perform worst on every metric for reasoning models. For the base LLMs, CoT unsupervised performed poorly, consistent with prior work that reported a reduction in overall performance with this strategy [52]. This same strategy, along with the CoT-text strategy, which performed similarly, are best suited for reasoning models. This suggests that we need to adapt the prompt strategy for the model used, and that LLMs benefit from having less detailed information as they develop reasoning capabilities to exploit thought processing. Ultimately, there is no single golden rule that determines which prompts are associated with higher performance for all models.

RQ3: Can iterative prompting help LLM achieve higher performances through self-verification and correction? For the iterative prompting experiments, as we had to respect the token limits of all open-source LLMs, we restricted our experiments to 5 and 10 students. Table 1 presents the results under role prompting, where the same conclusions can be draw from other prompt under Appendix C with Table 5-12. Additionally, while GPT-oss showed great results previously, the model is even more restrictive in the resources, so we limited the testing on the reasoning model to QwQ. In the cases where the model failed to produce the correct answer within the 5 attempts, we select the best attempt, i.e., the one that satisfied the highest number of metrics. Although this could give the impression that LLMs can understand the feedback given on this problem and correct the past errors to some degree, we can show that improvement is not monotonic by returning the last matching. In this case, while it is often possible to have very similar performance when returning the last or best attempt, there are other cases where it can also decrease significantly. This reveals that iterative prompting is not really iteratively improving the model's behavior and reasoning, but rather that generating multiple distinct answers is more beneficial in this scenario. Moreover, as the difference can only occur when the LLM fails to find the real student-optimal solution within 5 attempts, it is more challenging to evaluate the impact of self-correctness from the reasoning models.

Model	Attempt	Feasibility	Assignment	Matching	Optimality
Llama 8B	No	0.4678 (-)	0.1170 (-)	0.0409 (-)	0.0351 (-)
Llama 8B	Last	0.4386 (-2.9%)	0.1228 (+0.6%)	0.0702 (+2.9%)	0.0585 (++2.3%)
Llama 8B	Best	0.6023 (+13.5%)	0.1462 (2.9%)	0.0819 (+4.1%)	0.0585 (+2.3%)
Llama 70B	No	0.6316 (-)	0.0819 (-)	0.0468 (-)	0.0234 (-)
Llama 70B	Last	0.8538 (+22.2%)	0.1170 (+3.5%)	0.1111 (+6.4%)	0.0819 (+5.6%)
Llama 70B	Best	0.8889 (+25.7%)	0.1404 (+5.9%)	0.1404 (+9.4%)	0.0819 (+5.6%)
Mistral 7B	No	0.2242 (-)	0.5614 (-)	0.0585 (-)	0.0585 (-)
Mistral 7B	Last	0.1930 (-3.1%)	0.5673 (+0.6%)	0.0819(+2.3%)	0.0819(+2.3%)
Mistral 7B	Best	0.2281 (+0.4%)	0.6140 (+5.3%)	0.0819 (+2.3%)	0.0819 (+2.3%)
Qwen 7B	No	0.2339 (-)	0.2632 (-)	0.0351 (-)	0.0351
Qwen 7B	Last	0.4386 (+20.5%)	0.0760 (-18.7%)	0.0526 (+1.8%)	0.0468 (+1.2%)
Qwen 7B	Best	0.5497 (+31.6%)	0.1988 (-6.4%)	0.0585 (+2.3%)	0.0468 (+1.2%)
QwQ 32B	Non	1.0000 (-)	0.9181 (-)	0.9181 (-)	0.8304 (-)
QwQ 32B	Last	1.0000 (-)	0.9942 (+7.6%)	0.9942 (+7.6%)	0.9532 (+12.3%)
QwQ 32B	Best	1.0000 (-)	0.9883 (+7.0%)	0.9883 (+7.0%)	0.9415 (+11.1%)

Table 1: Iterative prompting with Role prompting for instance with 5-10 students. For each model, we have the metrics for no iterative prompting and iterative prompting with the last and best attempt

5 Discussion, Limitations and Future Work

The results of RQ1 showed that the many-to-one matching problem is a challenging task, where LLMs fail to satisfy the notions of stability, feasibility, and optimality, regardless of the prompts, as the number of students increases. Indeed, they indicate that certain aspects of the DA algorithm's runtime complexity, which expand the solution space, such as an increase in the number of students, are associated with a sharp decline in performance. However, the impact is not the same as the preferences, which also impact the algorithm's complexity and solution space. LLMs do not have a problem with incomplete preferences, which create invalid pairs, but it also does not lead to significantly better performance. One possible hypothesis is that when students are matched early in the iterative process, the distinction between preference types has limited impact, whereas the number of students consistently influences model performance. However, complete preferences may introduce greater difficulty in instances where the student-optimal matching assigns students to colleges ranked among their least preferred options. More experiments are needed to analyze how preference constraints impact the solution when scaling, or whether they would still lead to the same conclusions if the models were not instructed to follow this algorithm, where invalid pairs are naturally avoided.

Moreover, the results of RQ2 highlight the sensitivity of the models. Indeed, we observe that there is no prompt which systematically offers better performances over all models. More precisely, the performances under a prompting strategy depend on several aspects, including the model used, the corresponding number of parameters it has, and the observed metric. Additionally, even if we attempt to enhance the performance of base LLMs with these strategies, they will still fall short of matching the performance of reasoning LLMs with minimal prompting. However, there is a trend between strategies that are better fitted for base or reasoning LLMs. We hypothesize that base models can benefit from having detailed prompts, which will guide their thought process, making a substantial difference in the final results, while reasoning models have sufficient capabilities to do so with general guidelines.

Additionally, RQ3 demonstrates that iterative prompting is a promising approach for improving LLMs' reasoning under challenging tasks. However, the fact that results do not converge iteratively toward a better solution raises questions regarding the actual capacity of LLMs to integrate feedback and use it to improve their answers. Indeed, more experiments are necessary to determine whether the performances increase as a result of potential self-correction or simply by an increase in the answer diversity. An interesting direction could be to experiment with different feedback forms that vary in the level of detail provided regarding the encountered errors, like specifying the exact unstable pairs rather than only reporting the total number of unstable pairs in the matching.

While this work provides a valuable benchmark for a previously underexplored setting, it still has some limitations. Although testing each strategy allowed us to observe its individual impact on performance, further improvements could be achieved by combining strategies or by extending our current designs to include multiple examples in ICL. A common challenge in evaluating the most accessible LLMs, such as Llama, is fitting the prompt and output within the limit of allowed tokens, which also prevent us from scaling the instances to bigger sizes. Another limitation concerns the models tested in this article. Although other LLMs have achieved stronger performance on reasoning tasks, we limited our study to open-source models that can be used with reasonable resources. While future work could address current limitations, this work could also serve as an initial step toward aligning optimization outcomes with individual preferences, a concept often referred to as preferential alignment. While the DA algorithm guarantees stability through a strategy-proof mechanism, it can be inefficient [34]. Alternatives like Top Trading Cycles (TTC) and Risk Minimization (RM) offer solutions that may lead to higher overall satisfaction, even though they may produce matchings that are not stable [34]. Therefore, depending on the application, some users may prefer an outcome that prioritizes achieving their first rank through a highly efficient approach. In contrast, others may prefer stability, strategy-proofness, or fairness. LLMs offer a promising complement: not only as solvers but as interpreters capable of suggesting appropriate approaches based on the context and users' priorities through even deeper reasoning.

6 Conclusions

Solving a two-sided many-to-one matching problem remains highly challenging for LLMs. Although an algorithm exists that can compute a feasible, stable, and student-optimal solution in polynomial time, ensuring that the outputs of the models respect these properties is far from straightforward. Our benchmark reveals how well models handle structured preference-based reasoning, providing direct insight into their ability to align with domain-specific constraints. Specific properties, such as achieving a student-optimal stable matching, are particularly difficult to attain. Our iterative prompting experiments highlight the core challenge of self-correction based on feedback. We also observed that the models are more significantly affected by the number of students than by the type of preferences, even though both factors impact the running time of the DA algorithm. This suggests that the models struggle to follow the iterative structure required by this polynomial-time procedure correctly. Reasoning LLMs consistently outperform base ones across all metrics. Surprisingly, Mistral shows strong performance on stability despite being a base model. While no prompting strategy is universally optimal across all models, each model has a CoT variant that performs reliably well, making it a broadly effective approach.

Acknowledgements

Funding support for project activities has been partially provided by the Canada CIFAR AI Chair, IVADO, and CIFAR Catalyst award. We also thank Compute Canada and Mila clusters for their support in providing facilities for our evaluations. This work was funded by the NSERC Grant No. 2024-04051, and the Canada Graduate Research Scholarship. It is also funded by the FRQNT Master Scholarship No. B1X-341759.

References

[1] Atila Abdulkadiroğlu, Parag A. Pathak, and Alvin E. Roth. The New York City high school match. *American Economic Review*, 95(2):364–367, May 2005.

[2] Atila Abdulkadiroğlu, Parag A. Pathak, Alvin E. Roth, and Tayfun Sönmez. The Boston public school match. *American Economic Review*, 95(2):368–371, May 2005.

[3] Mustafa Oğuz Afacan, Umut Dur, and Martin Van der Linden. Capacity design in school choice. *Games and Economic Behavior*, 146:277–291, 2024.

[4] Sandhini Agarwal, Lama Ahmad, Jason Ai, Sam Altman, Andy Applebaum, Edwin Arbus, Rahul K Arora, Yu Bai, Bowen Baker, Haiming Bao, et al. gpt-oss-120b & gpt-oss-20b model card. *arXiv preprint arXiv:2508.10925*, 2025.

[5] Janice Ahn, Rishu Verma, Renze Lou, Di Liu, Rui Zhang, and Wenpeng Yin. Large language models for mathematical reasoning: Progresses and challenges. *arXiv preprint arXiv:2402.00157*, 2024.

[6] Haris Aziz, Hans Seedig, and Jana Wedel. On the susceptibility of the deferred acceptance algorithm. 02 2015.

[7] Péter Biró. Student admissions in Hungary as gale and shapley envisaged. 01 2008.

[8] Federico Bobbio, Margarida Carvalho, Andrea Lodi, Ignacio Rios, and Alfredo Torrico. Capacity planning in stable matching: An application to school choice. In *Proceedings of the 24th ACM Conference on Economics and Computation*, EC '23, New York, NY, USA, 2023. Association for Computing Machinery.

[9] Federico Bobbio, Margarida Carvalho, Andrea Lodi, and Alfredo Torrico. Capacity variation in the many-to-one stable matching, 05 2022.

[10] Sebastian Braun, Nadja Dwenger, and Dorothea Kübler. Telling the truth may not pay off. Discussion Papers 759, DIW Berlin, German Institute for Economic Research, 2007. Available at EconPapers: https://econpapers.repec.org/paper/diwdiwwpp/dp759.htm.

[11] Hongzheng Chen, Yingheng Wang, Yaohui Cai, Hins Hu, Jiajie Li, Shirley Huang, Chenhui Deng, Rongjian Liang, Shufeng Kong, Haoxing Ren, et al. Heurigym: An agentic benchmark for LLM-crafted heuristics in combinatorial optimization. *arXiv preprint arXiv:2506.07972*, 2025.

[12] Qiguang Chen, Libo Qin, Jinhao Liu, Dengyun Peng, Jiannan Guan, Peng Wang, Mengkang Hu, Yuhang Zhou, Te Gao, and Wangxiang Che. Towards reasoning era: A survey of long chain-of-thought for reasoning large language models. *arXiv preprint arXiv:2503.09567*, 2025.

[13] Wei-Lin Chiang, Lianmin Zheng, Ying Sheng, Anastasios Nikolas Angelopoulos, Tianle Li, Dacheng Li, Banghua Zhu, Hao Zhang, Michael Jordan, Joseph E Gonzalez, et al. Chatbot arena: An open platform for evaluating LLMs by human preference. In *Forty-first International Conference on Machine Learning*, 2024.

[14] Jose Correa, Rafael Epstein, Juan Escobar, Ignacio Rios, Bastian Bahamondes, Carlos Bonet, Natalie Epstein, Nicolas Aramayo, Martin Castillo, Andres Cristi, and Boris Epstein. School choice in Chile. In *Proceedings of the 2019 ACM Conference on Economics and Computation*, EC '19, page 325–343, New York, NY, USA, 2019. Association for Computing Machinery.

[15] Qingxiu Dong, Lei Li, Damai Dai, Ce Zheng, Jingyuan Ma, Rui Li, Heming Xia, Jingjing Xu, Zhiyong Wu, Tianyu Liu, et al. A survey on in-context learning. *arXiv preprint arXiv:2301.00234*, 2022.

[16] Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, Arun Rao, Aston Zhang, Aurélien Rodriguez, Austen Gregerson, Ava Spataru, Baptiste Rozière, Bethany Biron, Binh Tang, Bobbie Chern, Charlotte Caucheteux, Chaya Nayak, Chloe Bi, Chris Marra, Chris McConnell, Christian Keller, Christophe Touret, Chunyang Wu, Corinne Wong, Cristian Canton Ferrer, Cyrus Nikolaidis, Damien Allonsius, Daniel Song, Danielle Pintz, Danny Livshits, David Esiobu, Dhruv Choudhary, Dhruv Mahajan, Diego Garcia-Olano, Diego Perino, Dieuwke Hupkes, Egor Lakomkin, Ehab AlBadawy, Elina Lobanova, Emily Dinan, Eric Michael Smith, Filip Radenovic, Frank Zhang, Gabriel Synnaeve, Gabrielle Lee, Georgia Lewis Anderson, Graeme Nail, Grégoire Mialon, Guan Pang, Guillem Cucurell, Hailey Nguyen, Hannah Korevaar, Hu Xu, Hugo Touvron, Iliyan Zarov, Imanol Arrieta Ibarra, Isabel M. Kloumann, Ishan Misra, Ivan Evtimov, Jade Copet, Jaewon Lee, Jan Geffert, Jana Vranes, Jason Park, Jay Mahadeokar, Jeet Shah, Jelmer van der Linde, Jennifer Billock, Jenny Hong, Jenya Lee, Jeremy Fu, Jianfeng Chi, Jianyu Huang, Jiawen Liu, Jie Wang, Jiecao Yu, Joanna Bitton, Joe Spisak, Jongsoo Park, Joseph Rocca, Joshua Johnstun, Joshua Saxe, Junteng Jia, Kalyan Vasuden Alwala, Kartikeya Upasani, Kate Plawiak, Ke Li, Kenneth Heafield, Kevin Stone, and et al. The llama 3 herd of models. *CoRR*, abs/2407.21783, 2024.

[17] Shengyu Feng, Weiwei Sun, Shanda Li, Ameet Talwalkar, and Yiming Yang. A comprehensive evaluation of contemporary ML-based solvers for combinatorial optimization. *arXiv preprint arXiv:2505.16952*, 2025.

[18] David Gale and Lloyd S Shapley. College admissions and the stability of marriage. *The American Mathematical Monthly*, 69(1):9–15, 1962.

[19] Pei-Fu Guo, Ying-Hsuan Chen, Yun-Da Tsai, and Shou-De Lin. Towards optimizing with large language models. *arXiv preprint arXiv:2310.05204*, 2023.

[20] Chaoqun He, Renjie Luo, Yuzhuo Bai, Shengding Hu, Zhen Leng Thai, Junhao Shen, Jinyi Hu, Xu Han, Yujie Huang, Yuxiang Zhang, Jie Liu, Lei Qi, Zhiyuan Liu, and Maosong Sun. OlympiadBench: A challenging benchmark for promoting AGI with olympiad-level bilingual multimodal scientific problems, 2024.

[21] Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the math dataset. *arXiv preprint arXiv:2103.03874*, 2021.

[22] Duc Hieu Ho and Chenglin Fan. Self-critique-guided curiosity refinement: Enhancing honesty and helpfulness in large language models via in-context learning, 2025.

[23] Hadi Hosseini, Samarth Khanna, and Ronak Singh. Matching markets meet LLMs: Algorithmic reasoning with ranked preferences, 2025.

[24] Kaixuan Huang, Jiacheng Guo, Zihao Li, Xiang Ji, Jiawei Ge, Wenzhe Li, Yingqing Guo, Tianle Cai, Hui Yuan, Runzhe Wang, Yue Wu, Ming Yin, Shange Tang, Yangsibo Huang, Chi Jin, Xinyun Chen, Chiyuan Zhang, and Mengdi Wang. MATH-Perturb: Benchmarking LLMs' math reasoning abilities against hard perturbations, 2025.

[25] Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. Mistral 7b. *arXiv preprint arXiv:2310.06825*, 2023.

[26] Zhanming Jie, Trung Quoc Luong, Xinbo Zhang, Xiaoran Jin, and Hang Li. Design of chain-of-thought in math problem solving. *arXiv preprint arXiv:2309.11054*, Sep 2023. v2.

[27] Gurusha Juneja, Gautam Jajoo, Nagarajan Natarajan, Hua Li, Jian Jiao, and Amit Sharma. Task facet learning: A structured approach to prompt optimization. *arXiv preprint arXiv:2406.10504*, 2025. revised May 19, 2025; appeared in ACL Findings 2025.

[28] Muhammad Asif Khan and Layth Hamad. On the capability of LLMs in combinatorial optimization. *Authorea Preprints*, 2024.

[29] Satyapriya Krishna, Chirag Agarwal, and Himabindu Lakkaraju. Understanding the effects of iterative prompting on truthfulness, 2024.

[30] Hongwei Liu, Zilong Zheng, Yuxuan Qiao, Haodong Duan, Zhiwei Fei, Fengzhe Zhou, Wenwei Zhang, Songyang Zhang, Dahua Lin, and Kai Chen. MathBench: Evaluating the theory and application proficiency of LLMs with a hierarchical mathematics benchmark, 2024.

[31] Yuanye Liu, Jiahang Xu, Li Lyna Zhang, Qi Chen, Xuan Feng, Yang Chen, Zhongxin Guo, Yuqing Yang, and Cheng Peng. Beyond prompt content: Enhancing LLM performance via content-format integrated prompt optimization. *arXiv preprint arXiv:2502.04295*, Feb 2025.

[32] Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, Shashank Gupta, Bodhisattwa Prasad Majumder, Katherine Hermann, Sean Welleck, Amir Yazdanbakhsh, and Peter Clark. Self-refine: Iterative refinement with self-feedback, 2023.

[33] Yuetian Mao, Junjie He, and Chunyang Chen. From prompts to templates: A systematic prompt template analysis for real-world LLMapps, 04 2025.

[34] Josué Ortega and Thilo Klein. The cost of strategy-proofness in school choice. *Games and Economic Behavior*, 141:515–528, 2023.

[35] Shubham Parashar, Blake Olson, Sambhav Khurana, Eric Li, Hongyi Ling, James Caverlee, and Shuiwang Ji. Inference-time computations for LLM reasoning and planning: A benchmark and insights. *arXiv preprint arXiv:2502.12521*, 2025.

[36] Qwen Team. QwQ-32B: Embracing the Power of Reinforcement Learning. https://qwenlm.github.io/blog/qwq-32b/, March 2025. Published March 6, 2025; accessed 2025-08-29.

[37] Ignacio Rios, Federico Bobbio, Margarida Carvalho, and Alfredo Torrico. Stable matching with contingent priorities. *arXiv preprint arXiv:2409.04914*, 2024.

[38] Alvin E Roth. The Evolution of the Labor Market for Medical Interns and Residents: A Case Study in Game Theory. *Journal of Political Economy*, 92(6):991–1016, December 1984.

[39] Camilo Chacón Sartori and Christian Blum. Combinatorial optimization for all: Using LLMs to aid non-experts in improving optimization algorithms. *arXiv preprint arXiv:2503.10968*, 2025.

[40] Konstantinos Skianis, Giannis Nikolentzos, and Michalis Vazirgiannis. Graph reasoning with large language models via pseudo-code prompting, 09 2024.

[41] Kaya Stechly, Matthew Marquez, and Subbarao Kambhampati. GPT-4 doesn't know it's wrong: An analysis of iterative prompting for reasoning problems, 2023.

[42] Haoran Sun, Yusen Wu, Yukun Cheng, and Xu Chu. Game theory meets large language models: A systematic survey. *arXiv preprint arXiv:2502.09053*, 2025.

[43] Weiwei Sun, Shengyu Feng, Shanda Li, and Yiming Yang. CO-Bench: Benchmarking language model agents in algorithm search for combinatorial optimization. *arXiv preprint arXiv:2504.04310*, 2025.

[44] Tayfun Sönmez, Atila Abdulkadiroglu, and Tayfun Sonmez. School choice: A mechanism design approach. *American Economic Review*, 93:729–747, 02 2003.

[45] Heng Wang, Shangbin Feng, Tianxing He, Zhaoxuan Tan, Xiaochuang Han, and Yulia Tsvetkov. Can language models solve graph problems in natural language? *Advances in Neural Information Processing Systems*, 36:30840–30861, 2023.

[46] Xiaoxuan Wang, Ziniu Hu, Pan Lu, Yanqiao Zhu, Jieyu Zhang, Satyen Subramaniam, Arjun R Loomba, Shichang Zhang, Yizhou Sun, and Wei Wang. Scibench: Evaluating college-level scientific problem-solving abilities of large language models. *arXiv preprint arXiv:2307.10635*, 2023.

[47] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, brian ichter, Fei Xia, Ed H. Chi, Quoc V Le, and Denny Zhou. Chain of thought prompting elicits reasoning in large language models. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho, editors, *Advances in Neural Information Processing Systems*, 2022.

[48] An Yang, Baosong Yang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Zhou, Chengpeng Li, Chengyuan Li, Dayiheng Liu, Fei Huang, Guanting Dong, Haoran Wei, Huan Lin, Jialong Tang, Jialin Wang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Ma, and Zhihao Fan. Qwen2 technical report, 07 2024.

[49] Chengrun Yang, Xuezhi Wang, Yifeng Lu, Hanxiao Liu, Quoc V Le, Denny Zhou, and Xinyun Chen. Large language models as optimizers. *arXiv preprint arXiv:2309.03409*, 2023.

[50] Haoran Ye, Jiarui Wang, Zhiguang Cao, Federico Berto, Chuanbo Hua, Haeyeon Kim, Jinkyoo Park, and Guojie Song. Reevo: Large language models as hyper-heuristics with reflective evolution. *arXiv preprint arXiv:2402.01145*, 2024.

[51] Zheng Yuan, Hongyi Yuan, Chuanqi Tan, Wei Wang, and Songfang Huang. How well do large language models perform in arithmetic tasks? *arXiv preprint arXiv:2304.02015*, 2023.

[52] Xiang Zhang, Juntai Cao, Jiaqi Wei, Chenyu You, and Dujian Ding. Why prompt design matters and works: A complexity analysis of prompt search space in LLMs. *arXiv preprint arXiv:2503.10084*, Mar 2025. Presented at ACL 2025.

[53] Zhuosheng Zhang, Aston Zhang, Mu Li, and Alex Smola. Automatic chain of thought prompting in large language models. In *The Eleventh International Conference on Learning Representations*, 2023.

[54] Kolos Csaba Ágoston, Péter Biró, Endre Kováts, and Zsuzsanna Jankó. College admissions with ties and common quotas: Integer programming approach. *European Journal of Operational Research*, 299(2):722–734, 2022.

A Instances and Prompts

An instance of the College Admission Problem consists of students and colleges that must be written in a specific format to be compatible with the DA algorithm. Therefore, we introduce in the following Figure 5 an example of such an instance. Specifically, each instance consists of the number of students and colleges, followed by a list stating their names. The capacities of each school will follow it, and then we will have the preferences of students and colleges, respectively. As we compare the performance of the output of LLMs with the results of the DA algorithm, we compile the matching given by the DA algorithm for this prompt. The Figure 6 provides the answer format, as we expect it to be returned by LLM models. Note that when a student is matched to "nothing", it simply means that this student is unassigned. There is a pair for each student in the matching, but it is not necessarily for each school if no students are assigned to that school.

As detailed in Section 3.2, our benchmark evaluates model performance across four levels of student population, three types of preference, 3 different capacity level, four ratios and three random seeds. Certain combinations are excluded due to infeasibility, like under-capacity settings with a 1:1 ratio, which would imply fewer seats than schools, which is not meaningful in this context. The final dataset comprises 369 total instances per model–prompt combination. Specifically, for each preference type, there are 123 instances. For each model-prompt combination, we have 99 instances corresponding to 10,15 or 20 students, while there are 72 for 5 students. This design enables fine-grained analysis across different parameter settings while ensuring a sufficient number of instances to support statistically meaningful conclusions.

Each of the 369 benchmark instances was evaluated across six language models and eight distinct prompting strategies, as described in Section 3.3. Following conventions established in prior work

```
# Num. students:10
# Num. colleges:3
# Students:s1,s2,s3,s4,s5,s6,s7,s8,s9,s10
# Colleges:c1,c2,c3
# Capacities:
c1 3
c2 2
c3 3
# Student preferences:
s1 (1,c3) (2,c2) (3,c1)
s2 (1,c1) (2,c2) (3,c3)
s3 (1,c3) (2,c1) (3,c2)
s4 (1,c1) (2,c3) (3,c2)
s5 (1,c3) (2,c1) (3,c2)
s6 (1,c3) (2,c1) (3,c2)
s7 (1,c3) (2,c1) (3,c2)
s8 (1,c3) (2,c2) (3,c1)
s9 (1,c3) (2,c2) (3,c1)
s10 (1,c2) (2,c3) (3,c1)
# College priorities:
c1 (1,s2) (2,s1) (3,s10) (4,s9) (5,s7) (6,s6) (7,s4) (8,s3) (9,s5) (10,s8)
c2 (1,s9) (2,s8) (3,s6) (4,s5) (5,s10) (6,s1) (7,s4) (8,s3) (9,s2) (10,s7)
c3 (1,s8) (2,s1) (3,s6) (4,s3) (5,s7) (6,s10) (7,s5) (8,s9) (9,s4) (10,s2)
```

Figure 5: Prompt example of an College Admission instance.

```
# DA matching of corresponding input
[("s1", "c3"), ("s2", "c1"), ("s3", "nothing"), ("s4", "nothing"), ("s5", "c2"), ("s6", "c3"), ("s7",
"c1"), ("s8", "c3"), ("s9", "c2"), ("s10", "c1")]
```

Figure 6: Example of the results proposed by the DA algorithm.

[27, 31], each prompt was structured into distinct components, with explicit headers indicating the purpose of each section. We also appended the phrase "Final Matching:" at the end of each prompt to explicitly signal the model to return the final assignment. This technique, which is commonly used in question-answering tasks, helps improve output validity [53]. The prompts are presented from Figure 7 to Figure 14, with redundant content omitted for brevity.

Iterative Prompting Feedback: For iterative prompting, we are providing general feedback on the solution returned by the LLM. More specifically, if there is a matching in the previous output, we will return the feedback following this example that was feasible and stable, but not optimal.

B Valid Output

We defined validity in Section 4. Table 2 reports the proportion of the 369 instances that are valid. Invalid outputs include cases where the generated matching contains incorrect names (e.g., referring to students as "Alice" and "Bob" instead of s1 and s2), or when the model returns code rather than a matching. As shown in Table 2, there is no major problem and the proportions of valid output are almost always close to 1. However, Mistral exhibits a comparatively higher rate of invalid outputs, particularly when using the ICL with steps prompt. Qwen also produces a smaller number of invalid outputs under certain CoT prompting strategies. In contrast, both Llama models consistently generate valid outputs across all instances. It is worth noting that most invalid responses occurred for larger instances, suggesting that model can have a harder time following the instruction with bigger instances.

Task
Implement the student-proposing deferred acceptance algorithm (Gale-Shapley) to solve a many-to-one matching problem for college admissions.

Instance Format
The instance will contain the following elements:

- Num. students: Integer, the number of students
- Num. colleges: Integer, the number of colleges
- Students: List of student names
- Colleges: List of college names
- Capacities: Dictionary, each college mapped to its capacity (integer)
- Student preferences: Dictionary, each student is mapped to a list of tuples (rank,college)
- College preferences: Dictionary, each college is mapped to a list of tuples (rank, student)

Ranks are integers where 1 indicates the highest preference.

Constraints
The output matching must satisfy the following:
Feasibility:

- Each student is matched to at most one college.
- No college is matched with more students than its capacity.

Stability:

There must be no blocking pair (student, college) such that:

- The student prefers the college over their current match, and
- The college either has not reached its capacity or prefers the student to at least one of its current matches.

Optimality:

Among all stable matchings, students are matched to the best possible college based on their preferences.

Output Format
Return the matching as a list of tuples of the form (student, match). If a student is unmatched, use “nothing” as the college.

Instance
The instance to solve is: [ADD INSTANCE HERE]

Instruction
Return only the final matching as a list of tuples.

Final Matching:

Figure 7: Template of our **Basic** prompt strategy

	Llama 8B	Llama 70B	Mistral	Qwen	QwQ	GPT-oss
Basic	1.0000	1.0000	0.9837	1.0000	0.9892	0.9919
Role	1.0000	1.0000	0.9864	1.0000	0.9973	0.9919
ICL	1.0000	1.0000	0.7859	1.0000	0.9783	0.9946
ICL w steps	0.9458	1.0000	0.9729	1.0000	0.9729	0.9973
CoT unsupervised	1.0000	1.0000	0.9566	0.9973	1.0000	0.9946
CoT text	1.0000	1.0000	0.9837	0.8808	0.9946	0.9892
CoT pseudocode	1.0000	1.0000	0.9702	0.9864	0.9946	0.9973
CoT Python	1.0000	1.0000	0.9675	1.0000	0.9837	0.9973

Table 2: Percentage of output, out of the 369 possibilities, that generate an acceptable matching (i.e a matching that we could extract from the output)

You are an optimization algorithm expert specializing in stable matching algorithms, particularly the Gale-Shapley deferred acceptance algorithm.

Task
...

Instance Format
...

Constraints
...

Output Format
...

Instance
...

Instruction
...

Final Matching:

Figure 8: Template of our **Role** prompt strategy

Task
...

Instance Format
...

Constraints
...

Output Format
...

Example
Below is a solved example demonstrating input format and expected output. [ADD EXAMPLE HERE]

Instance
...

Instruction
...

Final Matching:

Figure 9: Template of our **ICL** prompt strategy

Task
...
Instance Format
...
Constraints
...
Output Format
...
Example
Here is an example consisting of an instance, the step-by-step application of the algorithm, and the final matching.
Instance
# Num. students: 5
# Num. Colleges: 3
# Students: s1, s2, s3, s4, s5
# Colleges: c1, c2, c3
# Capacities:
c1 2,
c2 1,
c3 2
# Student preferences:
s1 (1,c1) (2,c2) (3,c3)
s2 (1,c1) (2,c3) (3,c2)
s3 (1,c2) (2,c1) (3,c3)
s4 (1,c3) (2,c2) (3,c1)
s5 (1,c1) (2,c2) (3,c3)
# Colleges priorities:
c1 (1,s2) (2,s1) (3,s5) (4,s3) (5,s4)
c2 (1,s3) (2,s1) (3,s4) (4,s5) (5,s2)
c3 (1,s4) (2,s2) (3,s5) (4,s1) (5,s3)
Step by Step DA Algorithm
Round 1 :

s1, s2, s3, s4, s5 propose to their 1st choices, c1, c1, c2, c3, c1 respectively.

c1 (capacity of 2) gets proposals from s1, s2, s5. It prefers s1 and s2. It rejects s5.

c2 (capacity of 1) gets a proposal from s3 and holds it

c3 (capacity of 2) gets a proposal from s4 and holds it.

Applications of hold : c1 :s1,s2, c2 :s3 and c3:s4. Rejected:s5

Round 2 :

s5 (previously rejected) proposes to its next choice, c2.

c2 (capacity of 1) compares the new proposal (s5) to its current hold (s3). It prefers s3. It rejects s5.

Applications of hold : c1 :s1,s2, c2 :s3 and c3:s4. Rejected:s5

Round 3 :
s5 (previously rejected) proposes to its next choice, c2.
c3 (capacity of 2) gets a proposal from s5. It is holding s4 and has an open spot. It holds s5.
Applications of hold : c1 :s1,s2, c2 :s3 and c3:s4,s5. Rejected:
There was no rejection on the last round, the algorithm ends and the colleges accept the students on hold.
Final Matching
The final matching is :
(s1,c1),(s2,c1),(s3,c2),(s4,c3),(s5,c3)

Instance
...
Instruction
...
Final Matching:

Figure 10: Template of our **ICL w steps** prompt strategy

Task
...

Instance Format
...

Constraints
...

Instance
...

Instruction
Return only the final matching as a list of tuples. Think step by step.

Final Matching:

Figure 11: Template of our **CoT unsupervised** prompt strategy

Task
...

Instance Format
...

Constraints
...

Output Format
...

Instance
...

Step by Step Process

1. Each student applies to their favorite college.
2. Each college rejects all applications from students that are unacceptable to it. If a college received at most q applications from acceptable students so far, all those students are put on the colleges waiting list. Otherwise the college puts its favorite q students among all applicants on the waiting list and rejects all remaining ones.
3. Each student that was rejected in the previous step applies to his favorite among the colleges he or she has not yet applied to.
4. Steps 2 and 3 are repeated until it holds for all students that they were either not rejected in the previous step or already applied to all colleges acceptable to them.
5. Each college admits all students on its waiting list.

Instruction
...

Final Matching:

Figure 12: Template of our **text CoT** prompt strategy

Task
...

Instance Format
...

Constraints
...

Output Format
...

Instance
...
Step by Step Process

```
# INITIALIZE:
for each student s in S:
  s.next_choice ← 1        # index into Pref(s), starting at top choice
  s.is_free     ← true
for each college c in C:
  c.waiting_list ← ∅       # no one tentatively held yet
  c.applicants   ← ∅       # applications in the current round

REPEAT
  # Free students apply to their next choice
  for each student s in S:
    if s.is_free and s.next_choice ≤ length(Pref(s)):
      let c = Pref(s)[s.next_choice]
      add s to c.applicants
      s.next_choice ← s.next_choice + 1

  # Colleges review their applicant pools
  any_rejection ← false
  for each college c in C:
    pool ← c.waiting_list ∪ c.applicants
    remove from pool any student unacceptable to c
    sort pool by c's priority order (best first)
    c.waiting_list ← first q(c) students in pool
    rejected ← pool minus c.waiting_list
    for each student r in rejected:
      r.is_free ← true
      any_rejection ← true
    for each student a in c.waiting_list:
      a.is_free ← false
    c.applicants ← ∅

UNTIL any_rejection is false

# Final assignments
for each college c in C:
  admit every student in c.waiting_list
```

Instruction
...

Final Matching:

Figure 13: Template of our **pseudocode CoT** prompt strategy

Task
...

Instance Format
...

Constraints
...

Output Format
...

Instance
...

Step by Step Process

```
priority_ranks = {
    c: {s: rank for rank, s in enumerate(priority)}
    for c, priority in college_priorities.items()
}

proposals = {s: deque(prefs) for s, prefs in student_prefs.items()}
college_matches = defaultdict(list)
student_match = {s: None for s in students}

while True:
    free_students = [
        s for s in students
        if student_match[s] is None and proposals[s]
    ]
    if not free_students:
        break

    for student in free_students:
        college = proposals[student].popleft()
        college_matches[college].append(student)

        accepted = sorted(
            college_matches[college],
            key=lambda s: priority_ranks[college][s]
        )[:capacities[college]]

        for s in college_matches[college]:
            if s in accepted:
                student_match[s] = college
            else:
                student_match[s] = None

        college_matches[college] = accepted
```

Instruction
...

Final Matching:

Figure 14: Template of our **Python CoT** prompt strategy

Attempted Response:

Previous matching include here

This matching is incorrect. Here is some feedback based on four metrics:

- Feasibility: The matching is feasible. (number of seats over capacity= 0)
- Assignment stability: The matching is assignment stable. (number of blocking pairs=0)
- Matching stability: The matching is matching stable. (number of blocking pairs=0)
- Student-optimality: The matching is not student optimal. Some students do not have their best matching of all stable outcomes. (proportion of correct pairs= 0.8)

Figure 15: Feedback template example for a feasible and stable, but not student-optimal matching with iterative prompting.

C Tables and Graphs

C.1 Time Table

We present the time each LLM took in order to generate the output of 10 random instances across our datasets, making sure we had similar numbers for different complexity.

Model	Llama 8B	Llama 70B	Mistral 7B	Qwen 7B	QwQ 32B	GPT-oss 120B
Average Time	14.3111	6.9977	1.7036	6.5190	391.7150	324.5929
Standard deviation	5.347	2.918	0.9432	4.930	129.0224	308.9703

Table 3: Average time (in seconds) to generate the output for the College Admission Problem across 10 intances.

C.2 Aggregated Tables

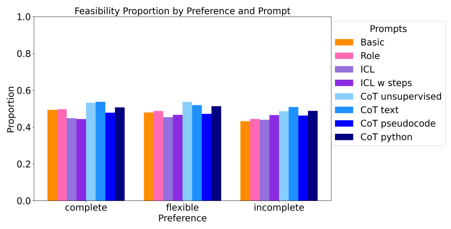
We include in this section, under Table 4 the aggregated results across the 369 instances, when are of them are valid, for each prompt strategy and model combination. While our figures included in the paper represent very well the data by number of students and preference types, this table allows to have a more broad view of the overall performances.

C.3 Iterative Prompting Tables

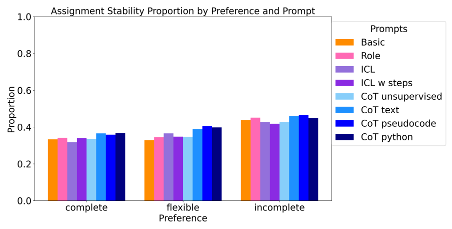
While Table 1 showed the main conclusions we can draw from iterative prompting, we present here the exact performances for role-based prompting, but also for every other prompt to show that the conclusions drawn before stay relevant with other prompts.

C.4 Graphs

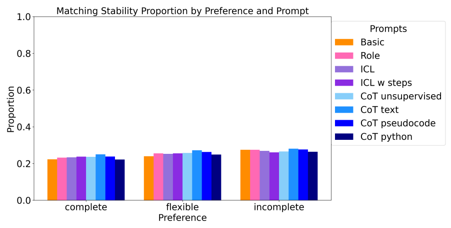
While the Figure 4 presented the trend for every prompt, the Figure 16 and Figure 17 show the detailed results for every prompt, under both complexity dimensions, the number of students and the preference types.



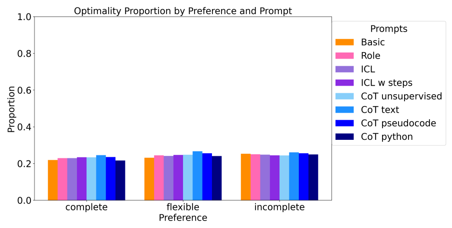
(a) Feasibility



(b) Assignment Stability

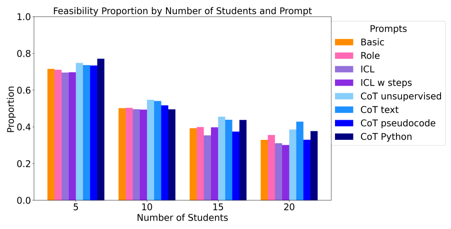


(c) Matching Stability

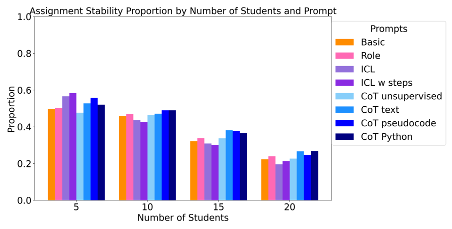


(d) Optimality

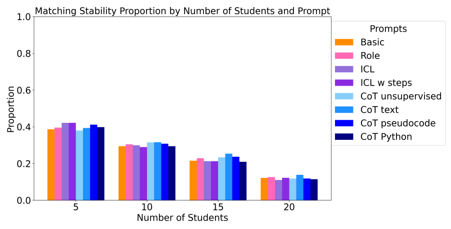
Figure 16: Proportion metrics by preference for all prompts.



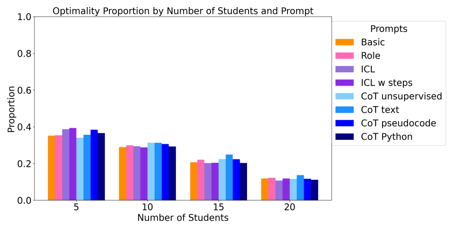
(a) Feasibility



(b) Assignment Stability



(c) Matching Stability



(d) Optimality

Figure 17: Proportion metrics by number of students for all prompts.

Prompt	Model	Count	Feasibility	Assignment	Matching	Optimality
Basic	Llama8B	369	0.3496	0.0542	0.0163	0.0136
Basic	Llama70B	369	0.3550	0.1463	0.0244	0.0136
Basic	Mistral	363	0.1460	0.4793	0.0303	0.0303
Basic	Qwen	369	0.1707	0.1084	0.0108	0.0108
Basic	QwQ	365	0.8750	0.6467	0.6332	0.5788
Basic	GPT-oss	366	0.9836	0.8115	0.8060	0.7924
Role	Llama8B	369	0.3523	0.0542	0.0190	0.0163
Role	Llama70B	369	0.3360	0.1680	0.0217	0.0108
Role	Mistral	364	0.1593	0.4560	0.0302	0.0302
Role	Qwen	369	0.1463	0.1409	0.0163	0.0163
Role	QwQ	368	0.8750	0.6467	0.6332	0.5788
Role	GPT-oss	366	0.9891	0.8115	0.8060	0.7978
ICL	Llama8B	369	0.2087	0.0867	0.0271	0.0217
ICL	Llama70B	369	0.3306	0.2087	0.0407	0.0298
ICL	Mistral	290	0.1897	0.4586	0.0448	0.0448
ICL	Qwen	369	0.1951	0.0705	0.0190	0.0190
ICL	QwQ	361	0.7756	0.5900	0.5789	0.5319
ICL	GPT	367	0.9864	0.8065	0.8011	0.7929
ICL w steps	Llama8B	349	0.3352	0.0201	0.0057	0.0057
ICL w steps	Llama70B	369	0.2791	0.2493	0.0407	0.0298
ICL w steps	Mistral	359	0.1086	0.4680	0.0446	0.0418
ICL w steps	Qwen	369	0.1870	0.0813	0.0244	0.0244
ICL w steps	QwQ	359	0.8524	0.5989	0.5989	0.5599
ICL w steps	GPT	368	0.9891	0.7989	0.7935	0.7908
CoT unsup.	Llama8B	369	0.4038	0.0461	0.0136	0.0108
CoT unsup.	Llama70B	369	0.3604	0.1653	0.0190	0.0108
CoT unsup.	Mistral	353	0.1728	0.4674	0.0255	0.0255
CoT unsup.	Qwen	368	0.3071	0.0734	0.0109	0.0109
CoT unsup.	QwQ	369	0.8943	0.6748	0.6612	0.6070
CoT unsup.	GPT	367	0.9728	0.7956	0.7902	0.7875
CoT text	Llama8B	369	0.5989	0.0244	0.0054	0.0054
CoT text	Llama70B	369	0.3198	0.1843	0.0325	0.0217
CoT text	Mistral	363	0.1515	0.5289	0.0331	0.0331
CoT text	Qwen	325	0.1785	0.1631	0.0215	0.0215
CoT text	QwQ	367	0.8965	0.7411	0.7248	0.6866
CoT text	GPT	365	0.9863	0.7918	0.7890	0.7781
CoT pseudo	Llama8B	369	0.3821	0.0379	0.0081	0.0081
CoT pseudo	Llama70B	369	0.3252	0.1463	0.0379	0.0271
CoT pseudo	Mistral	358	0.1257	0.6508	0.0447	0.0447
CoT pseudo	Qwen	364	0.1676	0.1566	0.0247	0.0220
CoT pseudo	QwQ	367	0.8420	0.6512	0.6349	0.5995
CoT pseudo	GPT-oss	368	0.9837	0.8125	0.8043	0.7935
CoT python	Llama8B	369	0.6016	0.0542	0.0136	0.0136
CoT python	Llama70B	369	0.3198	0.2168	0.0271	0.0190
CoT python	Mistral	357	0.1036	0.6723	0.0392	0.0392
CoT python	Qwen	369	0.2168	0.0921	0.0136	0.0027
CoT python	QwQ	363	0.7851	0.6088	0.5950	0.5592
CoT python	GPT-oss	368	0.9891	0.7880	0.7826	0.7799

Table 4: Overall performances over our 4 metrics for all model and prompt. For brevity, Assignment means Assignment Stability, Matching means Matching Stability, CoT unsup. refers to CoT unsupervised and CoT pseudo refers to CoT pseudocode.

Model	Iterative prompting	Feasibility	Assignment	Matching	Optimality
Llama8B	No	0.4561	0.1111	0.0351	0.0292
Llama8B	Last	0.4795	0.0936	0.0585	0.0468
Llama8B	Best	0.6140	0.1228	0.0702	0.0468
Llama70B	No	0.6491	0.0994	0.0526	0.0292
Llama70B	Last	0.8889	0.1287	0.1170	0.0819
Llama70B	Best	0.8947	0.1520	0.1345	0.0819
Mistral 7B	No	0.1988	0.6082	0.0585	0.0585
Mistral 7B	Last	0.1637	0.6257	0.0760	0.0760
Mistral 7B	Best	0.1871	0.6901	0.0760	0.0760
Qwen 7B	No	0.2632	0.1988	0.0234	0.0234
Qwen 7B	Last	0.4561	0.0643	0.0351	0.0234
Qwen 7B	Best	0.5322	0.1579	0.0351	0.0234
QwQ 32B	No	0.9883	0.8713	0.8713	0.8070
QwQ 32B	Last	1.0000	0.9883	0.9883	0.9181
QwQ 32B	Best	1.0000	0.9942	0.9942	0.9240

Table 5: Iterative prompting results with Basic prompting on an instance with 5-10 students. For each model, we have the metrics for no iterative prompting and iterative prompting with the last and best attempt, respectively.

Model	Iterative prompting	Feasibility	Assignment	Matching	Optimality
Llama 8B	Non	0.4678	0.1170	0.0409	0.0351
Llama 8B	Last	0.4386	0.1228	0.0702	0.0585
Llama 8B	Best	0.6023	0.1462	0.0819	0.0585
Llama 70B	Non	0.6316	0.0819	0.0468	0.0234
Llama 70B	Last	0.8538	0.1170	0.1111	0.0819
Llama 70B	Best	0.8889	0.1404	0.1404	0.0819
Mistral 7B	Non	0.2242	0.5614	0.0585	0.0585
Mistral 7B	Last	0.1930	0.5673	0.0819	0.0819
Mistral 7B	Best	0.2281	0.6140	0.0819	0.0819
Qwen 7B	Non	0.2339	0.2632	0.0351	0.0351
Qwen 7B	Last	0.4386	0.0760	0.0526	0.0468
Qwen 7B	Best	0.5497	0.1988	0.0585	0.0468
QwQ 32B	Non	1.0000	0.9181	0.9181	0.8304
QwQ 32B	Last	1.0000	0.9942	0.9942	0.9532
QwQ 32B	Best	1.0000	0.9883	0.9883	0.9415

Table 6: Iterative prompting with Role prompting for instance with 5-10 students. For each model, we have the metrics for no iterative prompting and iterative prompting with the last and best attempt

Model	Iterative prompting	Feasibility	Assignment	Matching	Optimality
Llama8B	No	0.3333	0.1871	0.0585	0.0468
Llama8B	Last	0.3333	0.1696	0.0526	0.0468
Llama8B	Best	0.2982	0.1754	0.0585	0.0526
Llama70B	No	0.6257	0.1754	0.0819	0.0585
Llama70B	Last	0.7953	0.1520	0.1345	0.1170
Llama70B	Best	0.8538	0.1930	0.1579	0.1170
Mistral 7B	No	0.2182	0.5758	0.0727	0.0727
Mistral 7B	Last	0.1930	0.5906	0.0819	0.0819
Mistral 7B	Best	0.1988	0.5965	0.0819	0.0819
Qwen 7B	No	0.3041	0.1520	0.0409	0.0409
Qwen 7B	Last	0.4386	0.0819	0.0409	0.0351
Qwen 7B	Best	0.4678	0.1287	0.0585	0.0409
QwQ 32B	No	1.0000	0.8824	0.8824	0.8118
QwQ 32B	Last	1.0000	0.9766	0.9766	0.9298
QwQ 32B	Best	1.0000	0.9766	0.9766	0.9181

Table 7: Iterative prompting results with ICL prompting on an instance with 5-10 students. For each model, we have the metrics for no iterative prompting and iterative prompting with the last and best attempt, respectively.

Model	Iterative prompting	Feasibility	Assignment	Matching	Optimality
Llama8B	No	0.5146	0.0409	0.0117	0.0117
Llama8B	Last	0.3684	0.0936	0.0468	0.0292
Llama8B	Best	0.5731	0.1053	0.0468	0.0292
Llama70B	No	0.5205	0.2105	0.0877	0.0643
Llama70B	Last	0.8304	0.1462	0.1404	0.0994
Llama70B	Best	0.8596	0.1871	0.1520	0.0994
Mistral 7B	No	0.1813	0.6959	0.0877	0.0819
Mistral 7B	Last	0.2105	0.6082	0.0994	0.0936
Mistral 7B	Best	0.2105	0.6842	0.0994	0.0936
Qwen 7B	No	0.2690	0.1754	0.0526	0.0526
Qwen 7B	Last	0.3392	0.0936	0.0643	0.0526
Qwen 7B	Best	0.3743	0.1462	0.0702	0.0526
QwQ 32B	No	0.9883	0.8772	0.8772	0.8304
QwQ 32B	Last	1.0000	0.9942	0.9942	0.9649
QwQ 32B	Best	1.0000	1.0000	1.0000	0.9708

Table 8: Iterative prompting results with ICL with steps prompting on an instance with 5-10 students. For each model, we have the metrics for no iterative prompting and iterative prompting with the last and best attempt, respectively.

Model	Iterative prompting	Feasibility	Assignment	Matching	Optimality
Llama8B	No	0.5029	0.0994	0.0292	0.0234
Llama8B	Last	0.4971	0.1053	0.0702	0.0468
Llama8B	Best	0.6608	0.1345	0.0760	0.0468
Llama70B	No	0.6550	0.0760	0.0409	0.0234
Llama70B	Last	0.8947	0.1228	0.1228	0.0702
Llama70B	Best	0.9006	0.1345	0.1287	0.0702
Mistral 7B	No	0.2222	0.6023	0.0526	0.0526
Mistral 7B	Last	0.1813	0.6374	0.0702	0.0702
Mistral 7B	Best	0.2164	0.6842	0.0702	0.0702
Qwen 7B	No	0.4152	0.1345	0.0234	0.0234
Qwen 7B	Last	0.4795	0.0643	0.0292	0.0234
Qwen 7B	Best	0.6257	0.1053	0.0292	0.0234
QwQ 32B	No	1.0000	0.9532	0.9532	0.8713
QwQ 32B	Last	1.0000	1.0000	1.0000	0.9415
QwQ 32B	Best	1.0000	1.0000	1.0000	0.9298

Table 9: Iterative prompting results with CoT unsupervised prompting on an instance with 5-10 students. For each model, we have the metrics for no iterative prompting and iterative prompting with the last and best attempt, respectively.

Model	Iterative prompting	Feasibility	Assignment	Matching	Optimality
Llama8B	No	0.6608	0.0526	0.0117	0.0117
Llama8B	Last	0.4678	0.0936	0.0468	0.0409
Llama8B	Best	0.7602	0.1170	0.0526	0.0409
Llama70B	No	0.6257	0.1345	0.0702	0.0468
Llama70B	Last	0.8129	0.1345	0.1287	0.0994
Llama70B	Best	0.8538	0.1930	0.1579	0.0994
Mistral 7B	No	0.1988	0.6199	0.0643	0.0643
Mistral 7B	Last	0.1988	0.6082	0.0702	0.0702
Mistral 7B	Best	0.2047	0.6374	0.0702	0.0702
Qwen 7B	No	0.2515	0.2573	0.0409	0.0409
Qwen 7B	Last	0.3567	0.1228	0.0468	0.0468
Qwen 7B	Best	0.4211	0.2164	0.0468	0.0468
QwQ 32B	No	1.0000	0.9649	0.9649	0.8947
QwQ 32B	Last	1.0000	0.9942	0.9942	0.9474
QwQ 32B	Best	1.0000	0.9883	0.9883	0.9415

Table 10: Iterative prompting results with CoT text prompting on an instance with 5-10 students. For each model, we have the metrics for no iterative prompting and iterative prompting with the last and best attempt, respectively.

Model	Iterative prompting	Feasibility	Assignment	Matching	Optimality
Llama8B	No	0.4912	0.1170	0.0292	0.0292
Llama8B	Last	0.4620	0.1111	0.0643	0.0643
Llama8B	Best	0.6316	0.1520	0.0702	0.0643
Llama70B	No	0.6023	0.1345	0.0585	0.0409
Llama70B	Last	0.8070	0.1404	0.1345	0.0994
Llama70B	Best	0.8304	0.2105	0.1579	0.0994
Mistral 7B	No	0.1696	0.8129	0.0819	0.0819
Mistral 7B	Last	0.1871	0.7368	0.0877	0.0877
Mistral 7B	Best	0.1930	0.8012	0.0877	0.0877
Qwen 7B	No	0.4211	0.1228	0.0292	0.0058
Qwen 7B	Last	0.4152	0.1228	0.0117	0.0117
Qwen 7B	Best	0.4620	0.1520	0.0351	0.0117
QwQ 32B	No	0.9883	0.8947	0.8947	0.8480
QwQ 32B	Last	1.0000	0.9942	0.9942	0.9532
QwQ 32B	Best	1.0000	0.9883	0.9883	0.9474

Table 11: Iterative prompting results with CoT python prompting on an instance with 5-10 students. For each model, we have the metrics for no iterative prompting and iterative prompting with the last and best attempt, respectively.

Model	Iterative prompting	Feasibility	Assignment	Matching	Optimality
Llama8B	No	0.5205	0.0760	0.0175	0.0175
Llama8B	Last	0.4561	0.1111	0.0760	0.0760
Llama8B	Best	0.6257	0.1637	0.0877	0.0760
Llama70B	No	0.6433	0.1637	0.0819	0.0585
Llama70B	Last	0.7485	0.1579	0.1520	0.1228
Llama70B	Best	0.8304	0.2105	0.1754	0.1228
Mistral 7B	No	0.1754	0.7661	0.0877	0.0877
Mistral 7B	Last	0.1930	0.6257	0.0877	0.0877
Mistral 7B	Best	0.1930	0.7544	0.0877	0.0877
Qwen 7B	No	0.3216	0.2281	0.0526	0.0468
Qwen 7B	Last	0.3977	0.0994	0.0643	0.0585
Qwen 7B	Best	0.4620	0.1988	0.0702	0.0585
QwQ 32B	No	0.9883	0.9064	0.9006	0.8538
QwQ 32B	Last	1.0000	0.9942	0.9942	0.9766
QwQ 32B	Best	1.0000	0.9942	0.9942	0.9766

Table 12: Iterative prompting results with CoT pseudocode prompting on an instance with 5-10 students. For each model, we have the metrics for no iterative prompting and iterative prompting with the last and best attempt, respectively.