Matching Markets Meet LLMs: Algorithmic Reasoning with Ranked Preferences

Hadi Hosseini
Penn State University, USA
hadi@psu.edu

Samarth Khanna
Penn State University, USA
samarth.khanna@psu.edu

Ronak Singh
Penn State University, USA
ronak.singh@psu.edu

Abstract

The rise of Large Language Models (LLMs) has driven progress in reasoning tasks, from program synthesis to scientific hypothesis generation, yet their ability to handle ranked preferences and structured algorithms in combinatorial domains remains underexplored. We study matching markets, a core framework behind applications like resource allocation and ride-sharing, which require reconciling individual ranked preferences to ensure stable outcomes. We evaluate seven state-of-the-art models on a hierarchy of preference-based reasoning tasks—ranging from stable-matching generation to instability detection, instability resolution, and fine-grained preference queries—to systematically expose their logical and algorithmic limitations in handling ranked inputs. Surprisingly, even top-performing models with advanced reasoning struggle to resolve instability in large markets, often failing to identify blocking pairs or execute algorithms iteratively. We further show that *parameter-efficient fine-tuning* (LoRA) significantly improves performance in small markets, but fails to bring about a similar improvement in large instances, suggesting the need for more sophisticated strategies to improve LLMs' reasoning with larger-context inputs.

Data and Code: github.com/SamarthKhanna/LLM_Matching_Markets

1 Introduction

The emergence of Large Language Models (LLMs) has positioned them as integral components in a wide range of reasoning-intensive tasks such as program synthesis, logical inference, mathematical problem solving, and scientific hypothesis generation, highlighting the importance of structured problem-solving capabilities. Despite their recent success in symbolic and logical reasoning, their capacity to reason over ranked preferences and to execute structured algorithms within combinatorial domains remains largely unexplored. Preference reasoning constitutes a foundational component in numerous domains, including economic contexts—e.g., auctions, voting systems, and market design—and in the architecture of pre-trained generative models using Reinforcement Learning from Human Feedback (RLHF) to capture and internalize human value judgments. These methods often have to execute algorithms on a large number of preference lists (either pairwise, partial, or complete rankings) to aggregate the rankings through constitutional AI [2] or social choice theory [15].

Despite substantial progress, reasoning over preferences remains a non-trivial endeavor: ensuring transitivity [64, 75], accurately augmenting ordinal rankings [23], and achieving coherent value alignment pose significant challenges. Without robust mechanisms for preference elicitation and the capacity to execute the requisite combinatorial procedures, even state-of-the-art LLMs may produce outputs that diverge from true human preferences [33] or fail to satisfy desirable properties [24].

We consider matching markets, a domain that constitutes a fundamental class of problems underlying diverse applications—from healthcare resource allocation to ride-sharing platforms and recommender

39th Conference on Neural Information Processing Systems (NeurIPS 2025).

https://doi.org/10.52202/085713-0639

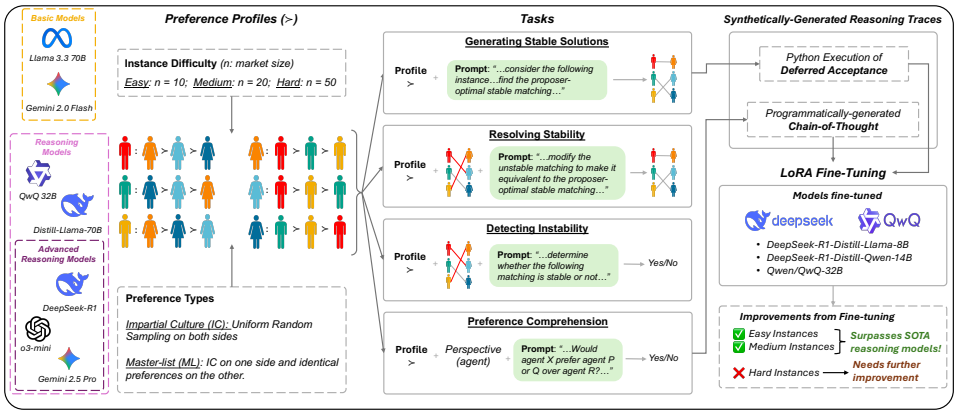

Figure 1: The framework for reasoning with ranked preferences through matching markets.

systems—and demand accurate comprehension of individual preferences and reconciliation of conflicting choices to guarantee system-wide stability. Matching markets are an ideal testbed for studying reasoning in AI models for two key reasons: First, they provide a structured platform for evaluating reasoning over ranked preferences and algorithmic thinking. They provide a framework that is axiomatically rich yet computationally tractable where solution quality (e.g., stability and efficiency) can be rigorously evaluated. Second, LLMs are increasingly utilized as black-box systems in a variety of economic, social, or medical settings to inform automated screening in recruitment pipelines [28], investigating market behavior [39], market clearing in ride-hailing platforms [42], and in general simulating economic interactions [32]. This makes it imperative to evaluate their ability in terms of computing "desirable" solutions from stakeholders' preferences. Additionally, their potential to serve as *interfaces* between stakeholders and established black-box systems (e.g., the National Residents Matching Program [61]) requires benchmarking their ability to reason about provided solutions and address aspects that users might find undesirable.

1.1 Our Results

We focus on four preference-based tasks: (i) *generating* stable solutions, requiring LLMs to produce stable matchings directly from ranked inputs, (ii) *instability resolution*, demanding preference reasoning to transform unstable matchings to stable ones, (iii) *instability detection*, in which models detect blocking pairs within a proposed solution, and (iv) *preference reasoning*, assessing nuanced query answering over ranked lists. We evaluate seven large language models with varying reasoning capabilities, including basic models, those with some limited reasoning, and advanced reasoning models. Our methodology and results are summarized in Figure 1.

Benchmark. We introduce a benchmark with instances and questions aimed at evaluating the above tasks involving reasoning over ranked preferences. These tasks are categorized into three levels of difficulty—Easy, Medium, and Hard—based on problem size. Each task utilizes ranked preferences sampled from two statistical distribution models: Impartial Culture (IC) and Master-list (ML).

Generating Stable Solutions. Although models with advanced-reasoning capabilities generally outperform other LLMs on Easy and Medium instances, *all* models struggle to generate stable solutions on Hard instances—indicating that the combinatorial reasoning capability of LLMs does not necessarily extend to larger-context inputs. Interestingly, the fraction of invalid and failed solutions is significantly lower for models with higher reasoning abilities, indicating their understanding of constraints, despite their inability to perform precise and step-by-step reasoning with preferences.

Instability Detection and Resolution. We find that LLMs frequently make mistakes in determining whether solutions are stable, with hallucinations about blocking pairs being the most common among basic models. Additionally, LLMs' ability to *correct* unstable solutions is (at best) as good as their ability to generate them from scratch, in some cases making the provided incorrect solutions worse.

Preference Reasoning. We consider tasks based on three levels of inference over ranked preferences. Large language models with advanced reasoning capabilities generally demonstrate a strong comprehension of preferences across levels of inference. However, even small errors compound in tasks

requiring multi-step sequences of reasoning (e.g., generating stable solutions or resolving instability), or in other words, small errors multiply!

Supervised Parameter-Efficient Fine-Tuning. We demonstrate that fine-tuning an open-source reasoning model using synthetically generated reasoning traces substantially improves performance, significantly outperforming advanced-reasoning models on Easy and Medium instances. However, we find that this approach does not address the challenges LLMs face with large inputs (Hard).

1.2 Related Work

Reasoning Capabilities of LLMs. Mathematical problem solving has been a key area of focus in evaluating the reasoning ability of LLMs, through a variety of benchmarks such as [14, 30, 31, 57]. LLMs have also demonstrated remarkable capabilities on coding benchmarks such as SWE-Bench [41] and CodeForces. As SOTA benchmark scores improve, recent work studies whether these improvements reflect genuine logical reasoning through benchmarks assessing logical consistency [50] and rule understanding/execution/planning [29]. Furthermore, the recent rise of LLM agents has increased interest in benchmarking LLMs' causal reasoning [12] and strategic planning abilities [20, 40, 66]. Additionally, the emergence of reasoning models has led to benchmarks evaluating these models' improved reasoning and planning abilities [10, 49].

Enhancing Reasoning Capabilities of LLMs. Specialized prompting strategies like Chain-of-Thought (CoT) [72], Tree-of-Thought (ToT) [77], and Graph-of-Thought (GoT) [3] have performance abilities on a variety of reasoning benchmarks. Fine-tuning has also been demonstrated to improve CoT in model outputs [78], as well as economic rationality [11] and abstract reasoning [74]. Additionally, instruction-tuning has been shown to enhance reasoning in several works [8, 48, 51, 71]. More advanced techniques build upon CoT [68, 80], or utilize multi-agent architectures that leverage cooperative LLMs [69, 79]. More recently, reinforcement-learning (GRPO) has been used to improve model reasoning, the most popular example being the Deepseek-R1 reasoning model [17].

LLMs in Social and Economic Decision Making. While still an emerging area of research, multiple works have focused on the collective decision-making capabilities of LLMs. One particular area of interest is the use of LLMs in preference elicitation [35, 65]. Fish et al. [24] benchmark the ability of models to learn and strategize in unknown economic environments using deliberate exploration. Another notable avenue of work is the study of how well LLMs can represent humans in collective decision-making, an understudied component of LLM alignment [33, 76].

2 Methodology

2.1 Problem Formulation

A *two-sided matching market* consists of two disjoint sets of agents (e.g., riders and drivers, freelancers and job requesters, and content creators and ads) denoted by M and W, where $|M| = |W| = n$. The preference list of an agent i, denoted by $\succ_i$, is a ranked order list over the agents on the other side. A *preference profile*, $\succ$, denotes the collection of preferences of all agents. We write $w_1 \succ_m w_2$ and $m_1 \succ_w m_2$ to denote that m *prefers* w_1 to w_2 and w *prefers* m_1 to m_2 respectively. In this paper, we primarily consider the standard model, which assumes a complete and strict preference list (no ties) and aims at finding a *one-to-one* matching between the agents in two sets.[1]

Matching and Stability. A *matching* is a function $\mu : M \cup W \to M \cup W$ such that $\mu(m) \in W$ for all $m \in M$, $\mu(w) \in M$ for all $w \in W$, and $\mu(m) = w$ if and only if $\mu(w) = m$. Given a matching μ, a *blocking pair* with respect to the preference profile $\succ$ is a pair (m, w) who prefer each other over their assigned partners in μ, i.e., $w \succ_m \mu(m)$ and $m \succ_w \mu(w)$. A matching is said to be *stable* if it does not have any blocking pairs. Given an instance of the problem, the set of all possible stable solutions forms a distributive lattice and can be *exponential* in size [45].

In their seminal work, Gale and Shapley [27] proposed an iterative procedure—the *deferred acceptance* algorithm (DA)—that always guarantees to find a stable solution. It proceeds by a series of proposals and rejections. In the initial *proposal* phase, each of the unmatched agents on one side (aka *proposers*) proposes to their favorite agent from the other side (aka *receivers*) according to their

[1]This is the standard model considered by the seminal works of Gale and Shapley [27] and Knuth [45].

 https://doi.org/10.52202/085713-0639

preference list. In the subsequent *rejection phase*, each agent on the receiving side tentatively accepts its preferred proposal, rejecting the others. The algorithm terminates when no further proposals can be made. The details of this algorithm can be found in Appendix C. The underlined solution in Example 1 is simultaneously *optimal* for the proposing side and *pessimal* for the receiving side [56]. We refer to the former as the **Optimal** matching and the latter as the **Pessimal** matching.

Example 1 (An instance with multiple stable solutions.)**.** *A preference profile for a sample instance of size $n = 4$; underlined agents indicate the Optimal matching, the Pessimal matching is indicated with a *, and the † indicates a stable matching that is different from the first two.*

$$\begin{array}{llll} \boldsymbol{m_1}: \underline{w_4} & w_3 & w_1^{*,\dagger} & w_2 \\ \boldsymbol{m_2}: \underline{w_3} & w_4^{\dagger} & w_2^{*} & w_1 \\ \boldsymbol{m_3}: \underline{w_1} & w_3^{\dagger} & w_2 & w_4^{*} \\ \boldsymbol{m_4}: w_1 & \underline{w_2^{\dagger}} & w_3^{*} & w_4 \end{array} \qquad \begin{array}{llll} \boldsymbol{w_1}: m_2 & m_1^{*,\dagger} & \underline{m_3} & m_4 \\ \boldsymbol{w_2}: m_2^{*} & m_3 & \underline{m_4^{\dagger}} & m_1 \\ \boldsymbol{w_3}: m_4^{*} & m_3^{\dagger} & m_1 & \underline{m_2} \\ \boldsymbol{w_4}: m_4 & m_3^{*} & m_2^{\dagger} & \underline{m_1} \end{array}$$

2.2 Dataset, Models, and Setup

Preference Instances. We synthetically sample a set of 300 preference profiles, partitioned into three sets of 100 instances for each *difficulty level*, namely **Easy** ($n = 10$ agents on each side of the market), **Medium** ($n = 20$), and **Hard** ($n = 50$). The preference profiles are sampled from two types of distributions **Impartial Culture** (IC) and **Master-list** (ML), each constituting 50 questions at each difficulty level. An impartial culture (IC) is a well-studied probabilistic model for generating preference profiles in which every agent's strict preference ranking is drawn independently and uniformly at random [4, 22]. It has been extensively studied in the context of economics, matching, and voting theory [5, 7, 9, 70]. A profile with a master-list (ML) is a highly structured preference profile in which all agents on one side of the market share exactly the same strict ranking over the agents on the other side. They represent the *homogeneity* in settings ranging from the labor market to organ allocation in healthcare [6, 38, 43] While an arbitrary instance generated by IC may admit *exponentially* many stable solutions [45], with a Master-list, only a single *unique* stable solution exists, indicating a difficulty level proportional to the size of the space of stable outcomes. In Appendix C, we discuss a simpler version of the DA algorithm for computing stable solutions with ML instances.

Matching Dataset. We curate a dataset comprising 2850 questions derived from the instances described above. These questions cover four *task categories*, each applied to the same pool of profiles to ensure consistency: (i) **generating stable solutions**, given a preference profile (300 questions); (ii) **instability resolution**, given a profile and an unstable matching; (iii) **instability detection**, given a profile and a solution (1050 questions); and (iv) **preference reasoning**, given a single preference list or a profile (900 questions).

Models. We select a representative suite of both open-source and closed-source models for evaluation. Since our benchmark is based on a reasoning task, we categorize models by their reasoning ability. We evaluate two **basic** models (those not specifically trained for reasoning), namely Llama-3.3-70B [21] and Gemini-2.0-Flash [60], and five **reasoning** models, namely Qwen-QwQ-32B [67], DeepSeek-70B (Llama-distilled) [17], OpenAI o3-mini [59], DeepSeek-R1 [17], and Gemini-2.5-Pro [16]. Among reasoning models, we classify the last three as **advanced reasoning** models, based on their SOTA performance on reasoning benchmarks [49].

Prompting. The prompt for each task consists of the preference profile for a given instance, followed by task-specific instructions (e.g., computing the "proposer-optimal" matching, or resolving a given unstable matching). While we adopt the *stable-marriage* setting, considered by Gale and Shapley [27], where *men* propose to *women*, we show (in Appendix D) that the results do not change if a different setting—where *workers* are assigned *tasks*—is used. To scale up the verification of solution correctness, we instruct LLMs to adhere to a predefined answer format. Additionally, we allow LLMs two *re-tries* to correct solutions that are either invalid, partial, or do not adhere to the specified format. See Appendix I for details about the inference setup, and Appendix J for sample prompts.

2.3 Evaluation Criteria

We consider several metrics for evaluating the quality of returned responses depending on the task. To account for cases in which LLM outputs violate task requirements, we categorize responses into

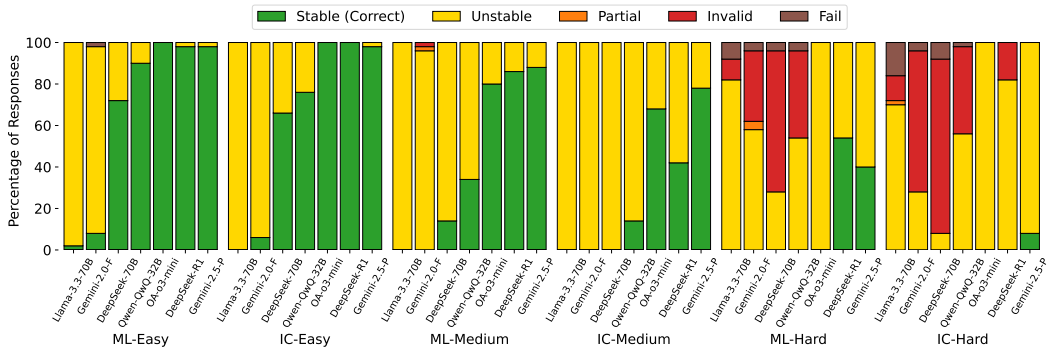

Figure 2: The generated responses by LLMs with Master-list (ML) and Impartial Culture (IC) preferences at different difficulty levels. *Stable* indicates one-to-one matchings with no blocking pairs; otherwise it is *unstable*. *Invalid* do not adhere to one-to-one constraint, *partial* are one-to-one but leave some unmatched, and *Fail* indicates models' failure to return any matching.

the following types: A solution is **invalid** if *some* agent from one side is matched to more than one agent from the other side. It is **partial** if it is not invalid, but some agents remain unmatched. A matching is **stable** if it is a *perfect* one-to-one matching that admits no blocking pair. Otherwise, it is **unstable** if it matches all the agents but admits a blocking pair. The following metrics apply primarily to valid responses. Informally, these metrics measure the distance from a reference stable outcome.

Instability Rate (IR): The instability rate measures the proportion of agents involved in blocking pairs, and thus the degree to which a matching violates the stability criterion. Given a complete matching, instability rate measures the percentage of unstable agents, i.e., those involved in at least one blocking pair. Formally, $\text{IR}(\mu, \succ) = \frac{|\{i \in M \cup W \text{ s.t. } j \succ_i \mu(i) \wedge i \succ_j \mu(j) \text{ for some } j \in M \cup W\}|}{2n}$.

Optimality/Pessimality Rate: This rate assesses the overlap between the model's matching and a reference stable matching, thereby capturing how closely the model's output mirrors the stepwise proposals and acceptances of a canonical algorithm. Formally, given two perfect matchings, μ and μ', in a one-to-one market where each matching is viewed as a set of unordered pairs between agents, the Jaccard similarity is defined as $\text{JS}(\mu, \mu') = \frac{|\mu \cap \mu'|}{|\mu \cup \mu'|}$. Then, we define **optimality rate** (OR) of a stable matching as its similarity to the proposer-optimal stable solution, which is unique.

3 Generating Stable Solutions

The first task involves evaluating LLMs' abilities to generate valid, stable matchings in markets with various difficulty levels. This task ideally requires models to reason over ranked preferences while iteratively executing a structured algorithm.

We consider two sub-categories for generating matchings depending on declarative knowledge about algorithms: i) prompt without specifying any algorithm, and ii) prompt with exact step-by-step instructions on how to execute the DA algorithm [27] (see Section 2.1 for details). Our ablation studies showed that the above prompting strategies did not result in qualitatively different outcomes, as all models were able to correctly identify the requirement for considering preferences, the DA algorithm, and its execution steps.[2] The detailed results are presented in Appendix D.

Difficulty, Model Size, and Reasoning. Figure 2 demonstrates the performance of the models in generating stable outcomes. Baseline models without explicit reasoning mechanisms are unable to solve even Easy instances, whereas reasoning-enabled models achieve high accuracy on Easy instances but suffer dramatic performance drops on Hard instances. Furthermore, for Hard problems, even reasoning models frequently produce invalid outputs or fail to return any solution. Interestingly,

[2]Furthermore, converting the *stable-marriage* setting to a *task-scheduling* setting [24], where "men" and "women" are replaced by "workers" and "tasks" (respectively), does not have a significant impact on performance.

https://doi.org/10.52202/085713-0639

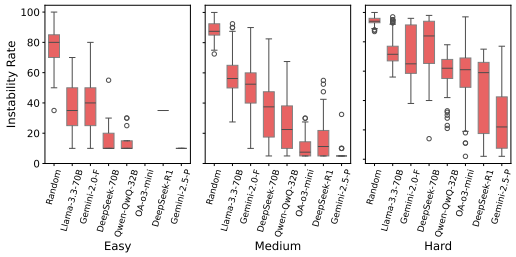

Figure 3: Instability Rate (lower is better) within unstable outcomes returned by each model as compared to randomly selected valid (but not necessarily stable) solutions (Random).

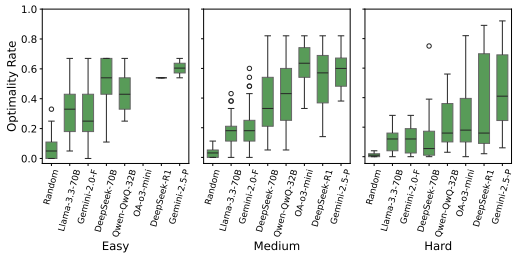

Figure 4: Optimality Rate within unstable outcomes returned by each model as compared to randomly selected valid (but not necessarily stable) solutions (Random).

Qwen-QwQ-32B significantly outperforms a much larger model, DeepSeek-70B, indicating that LLMs' combinatorial reasoning capability does not necessarily scale with model size.[3]

IC vs. ML Profiles. Under Impartial Culture (IC) profiles, the number of stable solutions can grow exponentially as the problem scales (increase of n) [45]. This combinatorial explosion poses a significant challenge for LLMs attempting to identify stable solutions, especially when solely using implicit reasoning over preference lists (without executing a concrete matching algorithm). In contrast, master-list profiles (ML)—irrespective of the underlying sampling method used to generate preferences—admit exactly *one* stable solution. Moreover, this unique stable matching can be constructed in $\mathcal{O}(n)$ steps by (i) extracting the common Master-list and then (ii) pairing agents in order of their shared priority [38]. See the details of the algorithm in Appendix C.

We observe that there is a significant performance gap between ML and IC instances—this disparity is especially marked for DeepSeek-R1. With IC profiles and Hard instances, *all* models are unable to compute a stable solution.[4] They perform slightly better under ML profiles, and while this performance drops for Hard instances, these models almost never return invalid or partial matchings.

Prompting Techniques. Prompt-engineering techniques have been empirically demonstrated to enhance the performance of LLMs on mathematical reasoning and formal logic inference tasks [3, 72, 77]. We evaluated a range of prompt-engineering techniques—including few-shot prompting and Chain-of-Thought (CoT) prompting, which supply exemplar "thought processes" and intermediate reasoning steps—in an attempt to bolster LLM performance. However, none of these strategies qualitatively improved on medium- or hard-difficulty instances. See Appendix D for further details.

3.1 Measuring Instability

A natural question is how far LLM-produced responses deviate from stable outcomes. To quantify this, we use two complementary metrics: instability rate and optimality rates (see Section 2.2). The **instability rate** directly reflects the distance from any stable solution, whereas the **optimality rate** implicitly evaluates the model's success in executing the underlying matching procedure. Figure 3 and Figure 4 illustrate comparisons of LLMs with a baseline of randomly generated outcomes.[5]

Broadly, the advanced-reasoning models generate significantly closer approximations to stability and optimality than their non-reasoning counterparts. Moreover, all evaluated LLMs (regardless of reasoning sophistication) substantially outperform random baselines on both metrics, indicating that they inherently leverage preference structures and exhibit nontrivial reasoning about ranked inputs.

Interestingly, the performance distinction between basic and reasoning models becomes less clear. While the intermediate reasoning models return a lower instability rate in Easy and Medium problems, their performance significantly drops in larger-scale problems (Hard). In fact, the performance of DeepSeek-70B becomes worse than even basic non-reasoning models. We attribute this behavior to

[3]Throughout the paper, all statistical comparisons between the percentages of stable solutions returned (across LLMs or across treatments) are done using Fisher's Exact test [25]. Similarly, any two distributions of Instability or Optimality Rate are statistically compared using Welch's T-test [73].

[4]Gemini-2.5-Pro is the only model with a positive success rate ($= 8\%$) with IC preferences Hard instances.

[5]Note that the plots only illustrate unstable but valid one-to-one outputs.

Table 1: The percentage of stable matchings returned when tasked with resolving instability starting from One-BP or Random matchings. The numbers in bold represent the highest accuracy (across all LLMs) of resolving the corresponding type of unstable matching.

		Basic LLMs				Reasoning LLMs				Advanced Reasoning LLMs					
		Gemini-2.0-Flash		Llama-3.3-70B		Qwen-QwQ-32B		DeepSeek-70B		o3-mini		DeepSeek-R1		Gemini-2.5-Pro	
Difficulty	Preference	One-BP	Random	One-BP	Random	One-BP	Random	One-BP	Random	One-BP	Random	One-BP	Random	One-BP	Random
Easy	IC	2	2	2	0	60	36	46	54	**100**	**100**	96	98	96	92
	ML	4	2	0	0	88	78	68	62	96	**100**	**100**	98	**100**	98
Medium	IC	0	0	0	0	22	0	10	0	64	**64**	28	32	**74**	60
	ML	0	0	0	0	17	7	20	6	82	78	**88**	76	80	**82**
Hard	IC	0	0	0	0	**4**	0	0	0	0	0	0	0	2	**2**
	ML	0	0	0	0	0	0	0	0	6	0	**28**	24	16	**34**
Average		1.00	0.67	0.33	0.00	31.83	20.16	24.00	20.33	58.00	57.00	56.67	54.67	**61.33**	**61.33**

the model's diminished capacity for handling larger input lengths, a hypothesis supported by their lower proportion of valid outcomes (as seen in Figure 2).

4 Resolving Instability

Generating stable solutions requires both exact reasoning over agents' preference lists and the execution of a stability-guaranteeing procedure (e.g., the DA algorithm). As demonstrated in Section 3, all evaluated models—irrespective of their reasoning capabilities—exhibit severe performance degradation as the problem size grows. This leads to the natural question of whether these models can resolve instability in a given matching—a task that entails detecting blocking pairs through preference reasoning and applying an appropriate sequence of adjustments to restore stability.

We provide LLMs with unstable (but valid) matchings along with preference profiles, and instruct them to convert these initialized solutions to stable matchings. To assess how the instability rate may influence solution quality, we distinguish two classes of initial matchings: (i) **One-BP**, matchings containing exactly one blocking pair (i.e., "almost stable") such that their stability may be resolved through simpler proposal-rejection iterations, and (ii) **Random**, matchings sampled uniformly at random from the set of all valid one-to-one pairings, which typically contain a large number of blocking pairs and thus exhibit high degrees of instability. See Appendix F for detailed steps and pseudo-code for generating one-BP and random initialization. Note that starting from an arbitrary matching, sequentially resolving blocking pairs may result in a cycle—as shown by Knuth [44]. However, a random sequence converges to stability with probability one [1, 62].

Table 1 displays the fraction of responses in which LLMs return stable matchings when asked to resolve the above types of unstable matching. Surprisingly, our experiments illustrate that in the task of resolving instability, the performance of all evaluated models does not exceed—and even degrades—their performance in generating stable solutions. This behavior persists regardless of initial matchings (One-BP or Random) and LLMs' reasoning capability. In fact, on Hard instances, the output returned by advanced reasoning models on One-BP matchings (i.e., containing a single blocking pair) contains substantially more than one blocking pair. In other words, even for the most basic instances, LLMs often introduce additional instabilities beyond the original single violation. We elaborate on this in Appendix F, demonstrating how all models, including those with advanced reasoning, often return solutions with a higher instability rate, highlighting their inability to systematically eliminate blocking pairs in accordance with preference lists.

5 Detecting Instability

The findings in previous sections raise the question of whether LLMs can reliably detect instability in a given matching—a simpler task that involves iterating over each unmatched pair to determine whether both agents prefer one another over their assigned partners. This procedure entails only a straightforward comparison of preferences and requires $\mathcal{O}(n^2)$ steps.

For this task, we evaluate the performance of valid one-to-one matchings initialized under two instability conditions: (i) One-BP, representing nearly stable matchings containing a single blocking pair, and (ii) Random, representing highly unstable matchings with numerous blocking pairs. To detect false-positives, we additionally include two extreme cases of stable matchings: the proposer-optimal (Optimal) and the proposer-pessimal (Pessimal) stable solutions.

https://doi.org/10.52202/085713-0639

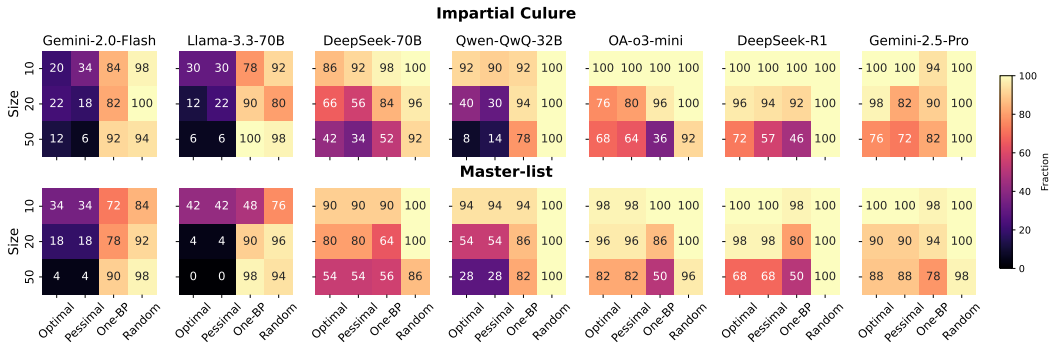

Figure 5: The fraction of responses where each model correctly detects stability or instability of a given matching.

Reasoning Models: Preferences and Blocking Pairs. Figure 5 reveals an interesting observation about reasoning models: their performance is influenced by the number of blocking pairs in the matching being evaluated—similar to the observations in Section 4. They achieve a high accuracy on identifying random matchings (which have a larger number of blocking pairs) as unstable and a significantly lower accuracy with matchings that have at most one blocking pair.

Basic Models and Hallucination. Interestingly, the non-reasoning models achieve a high accuracy (80%) with both types of unstable matchings, and extremely low (20%) accuracy with stable matchings. Note that the performance across all models is qualitatively similar in ML and IC profiles, even though each ML profile admits a unique stable solution (thus, identical reports for Optimal and Pessimal). See Appendix G for further analysis. A manual analysis of non-reasoning models uncovers frequent hallucinations about blocking pairs, resulting in a systematic bias toward classifying matchings as unstable. This can be largely attributed to misinterpretations of the input preferences.

6 Reasoning about Ranked Preferences

Many advanced reasoning paradigms, ranging from causal inference [12] and counterfactual analysis to game-theoretic decision making, depend fundamentally on the ability to compare and evaluate alternative choices. As demonstrated thus far, even advanced reasoning models often fail to execute the step-by-step procedures of combinatorial algorithms when those procedures operate over ranked preference lists. This shortcoming motivates the question of whether current LLMs can truly reason *about* preferences, as opposed to merely applying preferences in generating responses heuristically.

To investigate preference comprehension, we introduce a suite of tasks spanning three levels of inference over ranked preferences: (i) **basic retrieval (L1)**, in which models must extract individual preference relations; (ii) **comparison queries (L2)**, requiring pairwise preference judgments; and (iii) **proposal-acceptance simulations (L3)**, which combine comparison of alternatives with binary accept/reject decisions mirroring the dynamics of deferred-acceptance algorithms.

Hierarchical, level-wise reasoning evaluations have been proposed recently in domains such as causal inference of LLMs [12]. For example, an **L1** question is "*Who is agent W5's, 4th-most preferred agent?*", and an **L2** question "Would agent W5, prefer M8 over M7?"

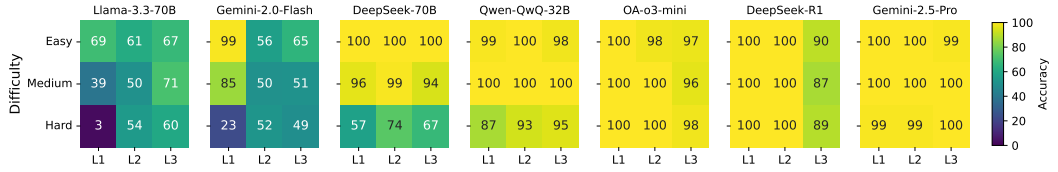

Figure 6: Accuracy on questions about provided preferences, with both IC and ML instances.

Basic models have low accuracy in all levels of difficulty even on small instances, which is probably the reason behind their inability to compute or detect stable solutions (as discussed in Section 3

and Section 5). In basic models (e.g., Llama-3.3-70B) and (non-advanced) reasoning models (e.g., DeepSeek-70B), the size of the problem has a greater impact on accuracy as compared to the level of the question, indicating their difficulty of handling larger inputs. Although advanced-reasoning models have significantly higher accuracy rates compared to other models, they still make minor errors, especially in larger profiles (Hard). Generating stable solutions for these instances often requires a larger number of reasoning steps over many preference lists, causing minor errors to compound (small errors multiply!).

7 Performance Improvement through LoRA Fine-Tuning

Supervised fine-tuning has proven to be effective in enhancing the reasoning capabilities of LLMs in a variety of tasks, including mathematical problem solving [13, 47], logical reasoning [58], and code generation [52]. Markeeva et al. [55] demonstrate how fine-tuning a small LLM (2B parameters) can significantly improve LLMs' ability to execute textbook algorithms (e.g., sorting an array, finding the shortest path between two nodes on a graph, etc.). Hence, we evaluate whether fine-tuning can be used to enhance LLMs' ability to compute stable matchings—a task requiring reasoning over ranked preferences in addition to algorithmic understanding. To this end, we perform LoRA fine-tuning [34] on three reasoning models, including Qwen-QwQ-32B and two smaller models, DeepSeek-8B and DeepSeek-14B, for Generating Stable Solutions task. Additionally, we also evaluate whether fine-tuning can mitigate errors made by LLMs in the Preference Reasoning task (Section 6).

Training. Let $\mathcal{D} = \{(\mathbf{x}^{(i)}, \mathbf{y}^{(i)})\}_{i=1}^{N}$ be the fine-tuning dataset for a given task. Each pair consists of an *input instance* $\mathbf{x}^{(i)}$ and the *desired model completion* $\mathbf{y}^{(i)}$. The input instance $\mathbf{x}^{(i)}$ is made up of four components: (i) a generic *system-prompt* s, (ii) a high-level *instruction* u, (iii) the *preference profile* $p^{(i)}$, and (iv) the *task-prompt* $t^{(i)}$. The desired completion $\mathbf{y}^{(i)}$ consists of two components, (i) a chain-of-thought *reasoning trace* $r^{(i)}$, and (ii) the *answer* $a^{(i)}$ in the desired format. Each model is fine-tuned with standard next-token cross-entropy on the concatenated sequence $\mathbf{z}^{(i)} = \mathbf{x}^{(i)}||\mathbf{y}^{(i)}$. We separately fine-tune each model for the Generating Stable solutions task ($N = 10,000$) and the Preference Reasoning task ($N = 9,000$). See Appendix H for details of the fine-tuning setup and results from experiments with different hyperparameter values.

Improvement. Fine-tuning LLMs with data containing synthetically generated reasoning traces substantially improves their performance on both tasks, as evidenced in Table 2. In fact, this approach enhances the performance of Qwen-QwQ-32B, a (non-advanced) reasoning model, to a success rate of 100% in computing stable matchings with *both* Easy and Medium instances, significantly outperforming advanced-reasoning models. Fine-tuning also clearly improves smaller models, i.e. DeepSeek-8B and DeepSeek-14B, both in terms of achieving stability and Instability Rate.[6] Similar results are obtained for the Preference Reasoning tasks, with the error-rate reducing to 0 at the Easy and Medium levels. [7]

Table 2: Improvement in performance in the Generating Stable Solutions and Preference Reasoning tasks after fine-tuning on respective datasets.

Model	Stage	Generating Stable Solutions						Preference Reasoning		
		Stable Solutions (%)			Instability Rate (↓)			Accuracy (%)		
		Easy	Med.	Hard	Easy	Med.	Hard	Easy	Med.	Hard
DeepSeek-8B	**Vanilla**	3.0	0.0	0.0	41.02	64.19	92.70	81.67	74.33	47.67
	Fine-tuned	64.0	31.0	0.0	6.00	12.13	–	**100.0**	98.33	75.00
DeepSeek-14B	**Vanilla**	19.0	0.0	0.0	20.66	55.31	94.09	97.67	91.33	72.33
	Fine-tuned	51.0	41.0	0.0	16.35	24.42	84.00	**100.0**	**100.0**	91.00
Qwen-QwQ-32B	**Vanilla**	83.0	24.0	0.0	2.35	19.27	59.07	99.00	**100.0**	91.67
	Fine-tuned	**100.0**	**100.0**	0.0	**0.00**	**0.00**	**55.19**	**100.0**	**100.0**	**99.00**

In spite of these improvements, however, there remains a distinct gap in performance at the Easy and Medium levels as compared to the Hard level. LLMs remain altogether unable to compute stable matchings for Hard instances, even after fine-tuning. A similar trend is reflected in the accuracy on the Preference Reasoning task. Hence, while fine-tuning clearly improves the reasoning capabilities of LLMs, further enhancements are required to improve their ability to handle larger inputs.

[6] Interestingly, this improvement is clearer for ML instances, where both models achieve a 100% success rate at the Easy level and $> 80\%$ success rate at the Medium level.

[7] The only exception being DeepSeek-8B at the Medium level.

https://doi.org/10.52202/085713-0639

Table 3: The performance of LLMs with different reasoning capability across all tasks requiring reasoning over ranked preferences and executing structured algorithms.

		Generating Stable Solutions			Resolving Instability			Detecting Instability	Preference Reasoning
Category	**Model**	**Stable Solutions (%)**	**Instability Rate (↓)**	**Optimality Rate (↑)**	**Stable Solutions (%)**	**Instability Rate (↓)**	**Optimality Rate (↑)**	**Accuracy (%)**	**Accuracy (%)**
Basic	**Llama-3.3-70B**	0.33	54.03	0.21	0.17	56.04	0.25	56.76	52.67
	Gemini-2.0-Flash	2.36	48.44	0.21	0.83	53.94	0.22	58.38	58.89
Reasoning	**DeepSeek-70B**	26.20	23.43	0.59	22.49	24.61	0.60	77.05	88.67
	Qwen-QwQ-32B	35.67	21.22	0.63	28.00	26.37	0.62	75.05	96.89
Advanced Reasoning	**o3-mini**	58.00	19.98	0.72	57.50	18.52	0.75	86.67	98.78
	DeepSeek-R1	64.22	12.35	0.80	55.73	14.21	**0.79**	88.19	96.22
	Gemini-2.5-Pro	**68.33**	**7.16**	**0.84**	**61.33**	**8.80**	**0.79**	**92.38**	**99.67**

8 Concluding Remarks

We summarize the performance of LLMs across all four tasks in Table 3, reflecting the clear hierarchy between advanced-reasoning models, (non-advanced) reasoning models, and basic models. The limitations in reliably reasoning about ranked preferences raise concerns about the viability of LLMs as agents acting on behalf of users in market-oriented or preference-sensitive decision-making settings, limit their capacity to negotiate complex user preferences, and hinder efforts in developing pluralistic techniques (e.g., constitutional AI [2] and social choice-theoretic [15]) for value alignment that are inherently based on aggregating rankings.

Open-Source vs. Closed-Source models. Among the models that we evaluate, Gemini-2.5-Pro (a closed-source model) emerges as the most capable across all tasks. While DeepSeek-R1 (open-source) broadly outperforms OpenAI's o3-mini (closed-source), it performs much worse with IC instances than with ML instances. While both basic models struggle on all tasks, Gemini-2.0-Flash (closed-source) marginally outperforms Llama-3.3-70B (open-source) on various metrics. Given the promising improvement yielded by fine-tuning an open-source reasoning model, i.e. Qwen-QwQ-32B, it is worth exploring strategies that also enable it to handle large inputs.

Beyond Linear Preferences. Our current evaluation paradigm considers complete and strict linear preferences. In real-world scenarios, however, preferences involve complexities such as incompleteness, indifference between alternatives, and capacity constraints [18, 46, 54]. As a result, algorithms to compute stable solutions in such settings are far more complex and solutions are often intractable [53]. While, in Appendix E, we provide some preliminary insights on preferences with ties, understanding such cases requires deeper theoretical and empirical investigation. A meaningful next step would be to examine how AI models respond to these more intricate preference structures.

Acknowledgments

This research was supported by the National Science Foundation (NSF) through CAREER Award IIS-2144413 and Award IIS-2107173. We thank the anonymous reviewers for their fruitful comments. We would also like to thank Shraddha Pathak for several useful discussions.

References

[1] Hernan Abeledo and Uriel G Rothblum. Paths to marriage stability. *Discrete Applied Mathematics*, 63(1):1–12, 1995.

[2] Yuntao Bai, Saurav Kadavath, Sandipan Kundu, Amanda Askell, Jackson Kernion, Andy Jones, Anna Chen, Anna Goldie, Azalia Mirhoseini, Cameron McKinnon, et al. Constitutional AI: Harmlessness from AI feedback. *arXiv preprint arXiv:2212.08073*, 2022.

[3] Maciej Besta, Nils Blach, Ales Kubicek, Robert Gerstenberger, Michał Podstawski, Lukas Gianinazzi, Joanna Gajda, Tomasz Lehmann, Hubert Niewiadomski, Piotr Nyczyk, and Torsten Hoefler. Graph of thoughts: solving elaborate problems with large language models. In *Proceedings of the Thirty-Eighth AAAI Conference on Artificial Intelligence and Thirty-Sixth Conference on Innovative Applications of Artificial Intelligence and Fourteenth Symposium on Educational Advances in Artificial Intelligence*, AAAI'24/IAAI'24/EAAI'24. AAAI Press,

2024. ISBN 978-1-57735-887-9. doi: 10.1609/aaai.v38i16.29720. URL https://doi.org/10.1609/aaai.v38i16.29720.

[4] Duncan Black et al. *The theory of committees and elections*. Springer, 1958.

[5] Niclas Boehmer, Piotr Faliszewski, Łukasz Janeczko, Andrzej Kaczmarczyk, Grzegorz Lisowski, Grzegorz Pierczyński, Simon Rey, Dariusz Stolicki, Stanisław Szufa, and Tomasz Wąs. Guide to numerical experiments on elections in computational social choice. *arXiv preprint arXiv:2402.11765*, 2024.

[6] Robert Bredereck, Klaus Heeger, Dušan Knop, and Rolf Niedermeier. Multidimensional stable roommates with master list, 2021. URL https://arxiv.org/abs/2009.14191.

[7] Angelina Brilliantova and Hadi Hosseini. Fair stable matching meets correlated preferences. In *Proceedings of the 21st International Conference on Autonomous Agents and Multiagent Systems*, pages 190–198, 2022.

[8] Huanqia Cai, Yijun Yang, and Zhifeng Li. System-2 mathematical reasoning via enriched instruction tuning. *CoRR*, abs/2412.16964, 2024. doi: 10.48550/ARXIV.2412.16964. URL https://doi.org/10.48550/arXiv.2412.16964.

[9] Ioannis Caragiannis and Karl Fehrs. Beyond the worst case: Distortion in impartial culture electorates, 2024. URL https://arxiv.org/abs/2307.07350.

[10] Qiguang Chen, Libo Qin, Jinhao Liu, Dengyun Peng, Jiannan Guan, Peng Wang, Mengkang Hu, Yuhang Zhou, Te Gao, and Wanxiang Che. Towards reasoning era: A survey of long chain-of-thought for reasoning large language models. *CoRR*, abs/2503.09567, 2025. doi: 10.48550/ARXIV.2503.09567. URL https://doi.org/10.48550/arXiv.2503.09567.

[11] Yiting Chen, Tracy Xiao Liu, You Shan, and Songfa Zhong. The emergence of economic rationality of gpt. *Proceedings of the National Academy of Sciences*, 120(51):e2316205120, 2023.

[12] Haoang Chi, He Li, Wenjing Yang, Feng Liu, Long Lan, Xiaoguang Ren, Tongliang Liu, and Bo Han. Unveiling causal reasoning in large language models: Reality or mirage? In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang, editors, *Advances in Neural Information Processing Systems*, volume 37, pages 96640–96670. Curran Associates, Inc., 2024. URL https://proceedings.neurips.cc/paper_files/paper/2024/file/af2bb2b2280d36f8842e440b4e275152-Paper-Conference.pdf.

[13] Hyung Won Chung, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus, Yunxuan Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, Albert Webson, Shixiang Shane Gu, Zhuyun Dai, Mirac Suzgun, Xinyun Chen, Aakanksha Chowdhery, Alex Castro-Ros, Marie Pellat, Kevin Robinson, Dasha Valter, Sharan Narang, Gaurav Mishra, Adams Yu, Vincent Y. Zhao, Yanping Huang, Andrew M. Dai, Hongkun Yu, Slav Petrov, Ed H. Chi, Jeff Dean, Jacob Devlin, Adam Roberts, Denny Zhou, Quoc V. Le, and Jason Wei. Scaling instruction-finetuned language models. *J. Mach. Learn. Res.*, 25:70:1–70:53, 2024. URL https://jmlr.org/papers/v25/23-0870.html.

[14] Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training verifiers to solve math word problems. *CoRR*, abs/2110.14168, 2021. URL https://arxiv.org/abs/2110.14168.

[15] Vincent Conitzer, Rachel Freedman, Jobst Heitzig, Wesley H Holliday, Bob M Jacobs, Nathan Lambert, Milan Mossé, Eric Pacuit, Stuart Russell, Hailey Schoelkopf, et al. Social choice should guide AI alignment in dealing with diverse human feedback. *arXiv preprint arXiv:2404.10271*, 2024.

[16] Google Deepmind. Gemini pro, Mar 2025. URL https://deepmind.google/technologies/gemini/pro/.

[17] DeepSeek-AI, Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, Xiaokang Zhang, Xingkai Yu, Yu Wu, Z. F. Wu, Zhibin Gou, Zhihong Shao, Zhuoshu Li, Ziyi Gao, Aixin Liu, Bing Xue, Bingxuan Wang, Bochao Wu, Bei Feng, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, Damai Dai, Deli Chen, Dongjie Ji, Erhang Li, Fangyun Lin, Fucong Dai, Fuli Luo, Guangbo Hao, Guanting Chen, Guowei Li, H. Zhang, Han Bao, Hanwei Xu, Haocheng Wang, Honghui Ding, Huajian Xin, Huazuo Gao, Hui Qu, Hui Li, Jianzhong Guo, Jiashi Li, Jiawei Wang, Jingchang Chen, Jingyang Yuan, Junjie Qiu, Junlong Li, J. L. Cai, Jiaqi Ni, Jian Liang, Jin Chen, Kai Dong, Kai Hu, Kaige Gao, Kang Guan, Kexin Huang, Kuai Yu, Lean Wang, Lecong Zhang, Liang Zhao, Litong Wang, Liyue Zhang, Lei Xu, Leyi Xia, Mingchuan Zhang, Minghua Zhang, Minghui Tang, Meng Li, Miaojun Wang, Mingming Li, Ning Tian, Panpan Huang, Peng Zhang, Qiancheng Wang, Qinyu Chen, Qiushi Du, Ruiqi Ge, Ruisong Zhang, Ruizhe Pan, Runji Wang, R. J. Chen, R. L. Jin, Ruyi Chen, Shanghao Lu, Shangyan Zhou, Shanhuang Chen, Shengfeng Ye, Shiyu Wang, Shuiping Yu, Shunfeng Zhou, Shuting Pan, and S. S. Li. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *CoRR*, abs/2501.12948, 2025. doi: 10.48550/ARXIV.2501.12948. URL `https://doi.org/10.48550/arXiv.2501.12948`.

[18] Maxence Delorme, Sergio García, Jacek Gondzio, Jörg Kalcsics, David F. Manlove, and William Pettersson. Mathematical models for stable matching problems with ties and incomplete lists. *Eur. J. Oper. Res.*, 277(2):426–441, 2019. doi: 10.1016/J.EJOR.2019.03.017. URL `https://doi.org/10.1016/j.ejor.2019.03.017`.

[19] 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.

[20] Jinhao Duan, Renming Zhang, James Diffenderfer, Bhavya Kailkhura, Lichao Sun, Elias Stengel-Eskin, Mohit Bansal, Tianlong Chen, and Kaidi Xu. Gtbench: Uncovering the strategic reasoning capabilities of llms via game-theoretic evaluations. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang, editors, *Advances in Neural Information Processing Systems*, volume 37, pages 28219–28253. Curran Associates, Inc., 2024. URL `https://proceedings.neurips.cc/paper_files/paper/2024/file/3191170938b6102e5c203b036b7c16dd-Paper-Conference.pdf`.

[21] 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. doi: 10.48550/ARXIV.2407.21783. URL `https://doi.org/10.48550/arXiv.2407.21783`.

[22] Ömer Eğecioğlu and Ayça E Giritligil. The impartial, anonymous, and neutral culture model: a probability model for sampling public preference structures. *The Journal of Mathematical Sociology*, 37(4):203–222, 2013.

[23] Sara Fish, Paul Gölz, David C. Parkes, Ariel D. Procaccia, Gili Rusak, Itai Shapira, and Manuel Wüthrich. Generative social choice. In Dirk Bergemann, Robert Kleinberg, and Daniela Sabán,

editors, *Proceedings of the 25th ACM Conference on Economics and Computation, EC 2024, New Haven, CT, USA, July 8-11, 2024*, page 985. ACM, 2024. doi: 10.1145/3670865.3673547. URL https://doi.org/10.1145/3670865.3673547.

[24] Sara Fish, Julia Shephard, Minkai Li, Ran I. Shorrer, and Yannai A. Gonczarowski. Econevals: Benchmarks and litmus tests for LLM agents in unknown environments. *CoRR*, abs/2503.18825, 2025. doi: 10.48550/ARXIV.2503.18825. URL https://doi.org/10.48550/arXiv.2503.18825.

[25] R. A. Fisher. On the interpretation of $\chi 2$ from contingency tables, and the calculation of p. *Journal of the Royal Statistical Society*, 85(1):87–94, 1922. ISSN 09528385. URL http://www.jstor.org/stable/2340521.

[26] David Gale and Lloyd S Shapley. College Admissions and the Stability of Marriage. *The American Mathematical Monthly*, 69(1):9–15, 1962.

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

[28] Chengguang Gan, Qinghao Zhang, and Tatsunori Mori. Application of LLM agents in recruitment: A novel framework for automated resume screening. *Journal of Information Processing*, 32:881–893, 2024.

[29] Jiayi Gui, Yiming Liu, Jiale Cheng, Xiaotao Gu, Xiao Liu, Hongning Wang, Yuxiao Dong, Jie Tang, and Minlie Huang. Logicgame: Benchmarking rule-based reasoning abilities of large language models. *CoRR*, abs/2408.15778, 2024. doi: 10.48550/ARXIV.2408.15778. URL https://doi.org/10.48550/arXiv.2408.15778.

[30] Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021. URL https://openreview.net/forum?id=d7KBjmI3GmQ.

[31] 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. In Joaquin Vanschoren and Sai-Kit Yeung, editors, *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks 1, NeurIPS Datasets and Benchmarks 2021, December 2021, virtual*, 2021. URL https://datasets-benchmarks-proceedings.neurips.cc/paper/2021/hash/be83ab3ecd0db773eb2dc1b0a17836a1-Abstract-round2.html.

[32] John J Horton. Large language models as simulated economic agents: What can we learn from homo silicus? Technical report, National Bureau of Economic Research, 2023.

[33] Hadi Hosseini and Samarth Khanna. Distributive fairness in large language models: Evaluating alignment with human values. *arXiv preprint arXiv:2502.00313*, 2025.

[34] Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net, 2022. URL https://openreview.net/forum?id=nZeVKeeFYf9.

[35] David Huang, Francisco J. Marmolejo Cossío, Edwin Lock, and David C. Parkes. Accelerated preference elicitation with llm-based proxies. *CoRR*, abs/2501.14625, 2025. doi: 10.48550/ARXIV.2501.14625. URL https://doi.org/10.48550/arXiv.2501.14625.

[36] Robert W. Irving. Stable marriage and indifference. *Discrete Applied Mathematics*, 48(3):261–272, 1994. ISSN 0166-218X. doi: https://doi.org/10.1016/0166-218X(92)00179-P. URL https://www.sciencedirect.com/science/article/pii/0166218X9200179P.

[37] Robert W. Irving, Paul Leather, and Dan Gusfield. An efficient algorithm for the "optimal" stable marriage. *Journal of the ACM*, 34(3):532–543, jul 1987. ISSN 0004-5411. doi: 10.1145/28869.28871. URL https://doi.org/10.1145/28869.28871.

[38] Robert W Irving, David F Manlove, and Sandy Scott. The stable marriage problem with master preference lists. *Discrete Applied Mathematics*, 156(15):2959–2977, 2008.

[39] Jingru Jia and Zehua Yuan. An experimental study of competitive market behavior through llms. *arXiv preprint arXiv:2409.08357*, 2024.

[40] Jingru Jia, Zehua Yuan, Junhao Pan, Paul McNamara, and Deming Chen. Large language model strategic reasoning evaluation through behavioral game theory. *CoRR*, abs/2502.20432, 2025. doi: 10.48550/ARXIV.2502.20432. URL https://doi.org/10.48550/arXiv.2502.20432.

[41] Carlos E. Jimenez, John Yang, Alexander Wettig, Shunyu Yao, Kexin Pei, Ofir Press, and Karthik R. Narasimhan. Swe-bench: Can language models resolve real-world github issues? In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net, 2024. URL https://openreview.net/forum?id=VTF8yNQM66.

[42] Enric Junqué De Fortuny. Simulating market equilibrium with large language models. In *Proceedings of the 58th Hawaii International Conference on System Sciences*, 2025.

[43] Naoyuki Kamiyama. Stable matchings with ties, master preference lists, and matroid constraints. In Martin Hoefer, editor, *Algorithmic Game Theory*, pages 3–14, Berlin, Heidelberg, 2015. Springer Berlin Heidelberg. ISBN 978-3-662-48433-3.

[44] Donald Ervin Knuth. Marriages stables. *Technical report*, 1976.

[45] Donald Ervin Knuth. *Stable marriage and its relation to other combinatorial problems: An introduction to the mathematical analysis of algorithms*, volume 10. American Mathematical Soc., 1997.

[46] Augustine Kwanashie and David F. Manlove. An integer programming approach to the hospitals/residents problem with ties. In Dennis Huisman, Ilse Louwerse, and Albert P. M. Wagelmans, editors, *Operations Research Proceedings 2013, Selected Papers of the International Conference on Operations Research, OR2013, organized by the German Operations Research Society (GOR), the Dutch Society of Operations Research (NGB) and Erasmus University Rotterdam, September 3-6, 2013*, pages 263–269. Springer, 2013. doi: 10.1007/978-3-319-07001-8_36. URL https://doi.org/10.1007/978-3-319-07001-8_36.

[47] Aitor Lewkowycz, Anders Andreassen, David Dohan, Ethan Dyer, Henryk Michalewski, Vinay V. Ramasesh, Ambrose Slone, Cem Anil, Imanol Schlag, Theo Gutman-Solo, Yuhuai Wu, Behnam Neyshabur, Guy Gur-Ari, and Vedant Misra. Solving quantitative reasoning problems with language models. In Sanmi Koyejo, S. Mohamed, A. Agarwal, Danielle Belgrave, K. Cho, and A. Oh, editors, *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*, 2022. URL http://papers.nips.cc/paper_files/paper/2022/hash/18abbeef8cfe9203fdf9053c9c4fe191-Abstract-Conference.html.

[48] Chengpeng Li, Zheng Yuan, Hongyi Yuan, Guanting Dong, Keming Lu, Jiancan Wu, Chuanqi Tan, Xiang Wang, and Chang Zhou. Mugglemath: Assessing the impact of query and response augmentation on math reasoning. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar, editors, *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2024, Bangkok, Thailand, August 11-16, 2024*, pages 10230–10258. Association for Computational Linguistics, 2024. doi: 10.18653/V1/2024.ACL-LONG.551. URL https://doi.org/10.18653/v1/2024.acl-long.551.

[49] Zhong-Zhi Li, Duzhen Zhang, Ming-Liang Zhang, Jiaxin Zhang, Zengyan Liu, Yuxuan Yao, Haotian Xu, Junhao Zheng, Pei-Jie Wang, Xiuyi Chen, Yingying Zhang, Fei Yin, Jiahua Dong, Zhijiang Guo, Le Song, and Cheng-Lin Liu. From system 1 to system 2: A survey of reasoning large language models. *CoRR*, abs/2502.17419, 2025. doi: 10.48550/ARXIV.2502.17419. URL https://doi.org/10.48550/arXiv.2502.17419.

[50] Yinhong Liu, Zhijiang Guo, Tianya Liang, Ehsan Shareghi, Ivan Vulic, and Nigel Collier. Aligning with logic: Measuring, evaluating and improving logical consistency in large language models. *CoRR*, abs/2410.02205, 2024. doi: 10.48550/ARXIV.2410.02205. URL https://doi.org/10.48550/arXiv.2410.02205.

[51] Liangchen Luo, Yinxiao Liu, Rosanne Liu, Samrat Phatale, Harsh Lara, Yunxuan Li, Lei Shu, Yun Zhu, Lei Meng, Jiao Sun, and Abhinav Rastogi. Improve mathematical reasoning in language models by automated process supervision. *CoRR*, abs/2406.06592, 2024. doi: 10.48550/ARXIV.2406.06592. URL https://doi.org/10.48550/arXiv.2406.06592.

[52] Ziyang Luo, Can Xu, Pu Zhao, Qingfeng Sun, Xiubo Geng, Wenxiang Hu, Chongyang Tao, Jing Ma, Qingwei Lin, and Daxin Jiang. Wizardcoder: Empowering code large language models with evol-instruct. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net, 2024. URL https://openreview.net/forum?id=UnUwSIgK5W.

[53] David F. Manlove. Algorithmics of matching under preferences. *Bull. EATCS*, 112, 2014. URL http://eatcs.org/beatcs/index.php/beatcs/article/view/252.

[54] David F. Manlove, Duncan Milne, and Sofiat Olaosebikan. Student-project allocation with preferences over projects: Algorithmic and experimental results. *Discret. Appl. Math.*, 308: 220–234, 2022. doi: 10.1016/J.DAM.2020.08.015. URL https://doi.org/10.1016/j.dam.2020.08.015.

[55] Larisa Markeeva, Sean McLeish, Borja Ibarz, Wilfried Bounsi, Olga Kozlova, Alex Vitvitskyi, Charles Blundell, Tom Goldstein, Avi Schwarzschild, and Petar Velickovic. The clrs-text algorithmic reasoning language benchmark. *CoRR*, abs/2406.04229, 2024. doi: 10.48550/ARXIV.2406.04229. URL https://doi.org/10.48550/arXiv.2406.04229.

[56] David G McVitie and Leslie B Wilson. The stable marriage problem. *Communications of the ACM*, 14(7):486–490, 1971.

[57] Seyed-Iman Mirzadeh, Keivan Alizadeh, Hooman Shahrokhi, Oncel Tuzel, Samy Bengio, and Mehrdad Farajtabar. Gsm-symbolic: Understanding the limitations of mathematical reasoning in large language models. *CoRR*, abs/2410.05229, 2024. doi: 10.48550/ARXIV.2410.05229. URL https://doi.org/10.48550/arXiv.2410.05229.

[58] Terufumi Morishita, Gaku Morio, Atsuki Yamaguchi, and Yasuhiro Sogawa. Enhancing reasoning capabilities of llms via principled synthetic logic corpus. In Amir Globersons, Lester Mackey, Danielle Belgrave, Angela Fan, Ulrich Paquet, Jakub M. Tomczak, and Cheng Zhang, editors, *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*, 2024. URL http://papers.nips.cc/paper_files/paper/2024/hash/8678da90126aa58326b2fc0254b33a8c-Abstract-Conference.html.

[59] OpenAI. Openai o3-mini system card, 2025. URL https://openai.com/index/o3-mini-system-card/.

[60] Sundar Pichai, Demis Hassabis, and Kouray Kavukcuoglu. Introducing gemini 2.0: our new ai model for the agentic era, Dec 2024. URL https://blog.google/technology/google-deepmind/google-gemini-ai-update-december-2024/.

[61] National Resident Matching Program. Results and data: 2024 main residency match, 2024. URL https://www.nrmp.org/match-data/2024/06/results-and-data-2024-main-residency-match/.

[62] Alvin E Roth and John H Vande Vate. Random paths to stability in two-sided matching. *Econometrica: Journal of the Econometric Society*, pages 1475–1480, 1990.

[63] Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. Deepseekmath: Pushing the limits of mathematical reasoning in open language models, 2024. URL https://arxiv.org/abs/2402.03300.

[64] Kiwon Song, James M Jennings III, and Clintin P Davis-Stober. Benchmarking the rationality of AI decision making using the transitivity axiom. *arXiv preprint arXiv:2502.10554*, 2025.

[65] Ermis Soumalias, Yanchen Jiang, Kehang Zhu, Michael J. Curry, Sven Seuken, and David C. Parkes. Llm-powered preference elicitation in combinatorial assignment. *CoRR*, abs/2502.10308, 2025. doi: 10.48550/ARXIV.2502.10308. URL https://doi.org/10.48550/arXiv.2502.10308.

[66] Wenjie Tang, Yuan Zhou, Erqiang Xu, Keyan Cheng, Minne Li, and Liquan Xiao. Dsgbench: A diverse strategic game benchmark for evaluating llm-based agents in complex decision-making environments. *CoRR*, abs/2503.06047, 2025. doi: 10.48550/ARXIV.2503.06047. URL https://doi.org/10.48550/arXiv.2503.06047.

[67] Qwen Team. Qwq: Reflect deeply on the boundaries of the unknown, Nov 2024. URL https://qwenlm.github.io/blog/qwq-32b-preview/.

[68] Ye Tian, Baolin Peng, Linfeng Song, Lifeng Jin, Dian Yu, Lei Han, Haitao Mi, and Dong Yu. Toward self-improvement of llms via imagination, searching, and criticizing. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang, editors, *Advances in Neural Information Processing Systems*, volume 37, pages 52723–52748. Curran Associates, Inc., 2024. URL https://proceedings.neurips.cc/paper_files/paper/2024/file/5e5853f35164e434015716a8c2a66543-Paper-Conference.pdf.

[69] Vince Trencsenyi, Agnieszka Mensfelt, and Kostas Stathis. Approximating human strategic reasoning with llm-enhanced recursive reasoners leveraging multi-agent hypergames. *CoRR*, abs/2502.07443, 2025. doi: 10.48550/ARXIV.2502.07443. URL https://doi.org/10.48550/arXiv.2502.07443.

[70] Ilia Tsetlin, Michel Regenwetter, and Bernard Grofman. The impartial culture maximizes the probability of majority cycles. *Social Choice and Welfare*, 21(3):387–398, 2003. ISSN 01761714, 1432217X. URL http://www.jstor.org/stable/41106568.

[71] Emily Vaillancourt and Christopher Thompson. Instruction tuning on large language models to improve reasoning performance. *Authorea Preprints*, 2024.

[72] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837, 2022.

[73] B. L. Welch. The generalization of 'student's' problem when several different population variances are involved. *Biometrika*, 34(1/2):28–35, 1947. ISSN 00063444. URL http://www.jstor.org/stable/2332510.

[74] Kai Xiong, Xiao Ding, Ting Liu, Bing Qin, Dongliang Xu, Qing Yang, Hongtao Liu, and Yixin Cao. Meaningful learning: Enhancing abstract reasoning in large language models via generic fact guidance. In Amir Globersons, Lester Mackey, Danielle Belgrave, Angela Fan, Ulrich Paquet, Jakub M. Tomczak, and Cheng Zhang, editors, *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*, 2024. URL http://papers.nips.cc/paper_files/paper/2024/hash/da5498f88193ff61f0daea1940b819da-Abstract-Conference.html.

[75] Yi Xu, Laura Ruis, Tim Rocktäschel, and Robert Kirk. Investigating non-transitivity in LLM-as-a-judge. *arXiv preprint arXiv:2502.14074*, 2025.

[76] Joshua C. Yang, Damian Dailisan, Marcin Korecki, Carina I. Hausladen, and Dirk Helbing. LLM voting: Human choices and AI collective decision-making. In Sanmay Das, Brian Patrick Green, Kush Varshney, Marianna Ganapini, and Andrea Renda, editors, *Proceedings of the Seventh AAAI/ACM Conference on AI, Ethics, and Society (AIES-24) - Full Archival Papers, October 21-23, 2024, San Jose, California, USA - Volume 1*, pages 1696–1708. AAAI Press, 2024. doi: 10.1609/AIES.V7I1.31758. URL https://doi.org/10.1609/aies.v7i1.31758.

[77] Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Thomas L. Griffiths, Yuan Cao, and Karthik Narasimhan. Tree of thoughts: deliberate problem solving with large language models. In *Proceedings of the 37th International Conference on Neural Information Processing Systems*, NIPS '23, Red Hook, NY, USA, 2023. Curran Associates Inc.

[78] Xuan Zhang, Chao Du, Tianyu Pang, Qian Liu, Wei Gao, and Min Lin. Chain of preference optimization: Improving chain-of-thought reasoning in llms. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang, editors, *Advances in Neural Information Processing Systems*, volume 37, pages 333–356. Curran Associates, Inc., 2024. URL https://proceedings.neurips.cc/paper_files/paper/2024/file/00d80722b756de0166523a87805dd00f-Paper-Conference.pdf.

[79] Heng Zhou, Hejia Geng, Xiangyuan Xue, Zhenfei Yin, and Lei Bai. Reso: A reward-driven self-organizing llm-based multi-agent system for reasoning tasks. *CoRR*, abs/2503.02390, 2025. doi: 10.48550/ARXIV.2503.02390. URL https://doi.org/10.48550/arXiv.2503.02390.

[80] Pei Zhou, Jay Pujara, Xiang Ren, Xinyun Chen, Heng-Tze Cheng, Quoc V. Le, Ed H., Denny Zhou, Swaroop Mishra, and Huaixiu Steven Zheng. Self-discover: Large language models self-compose reasoning structures. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang, editors, *Advances in Neural Information Processing Systems*, volume 37, pages 126032–126058. Curran Associates, Inc., 2024. URL https://proceedings.neurips.cc/paper_files/paper/2024/file/e41efb03e20ca3c231940a3c6917ef6f-Paper-Conference.pdf.

 https://doi.org/10.52202/085713-0639

A Limitations and Future Work

While we present a comprehensive evaluation of the practical algorithmic and economic reasoning capabilities of a series of state-of-the-art LLMs, our dataset primarily relies on synthetic data due to the challenges in obtaining real-world ordinal preference data. This calls for the collection and curation of datasets in the two-sided matching setting, and generating preference profile datasets that are better aligned with human preferences.

Additionally, while our work provides insights into the reasons behind the failure of LLMs to consistently generate stable solutions (see Section 6), there is scope for further clarity on where exactly LLMs make mistakes during algorithmic execution. A potential method to explore this is to break the algorithmic execution task into smaller steps (e.g., a single proposal-acceptance/rejection cycle) and identify which components of the *state-transition* are challenging for LLMs to understand.

Furthermore, while fine-tuning substantially enhances LLMs' performance on instances with relatively smaller input sizes, improving their performance with larger inputs requires further exploration. This can include the investigation of methods such as fine-tuning the entire set of parameters (unlike with LoRA) or reinforcement-learning methods such as *group-relative policy optimization* (GRPO) that are known to increase the reasoning capabilities of LLMs [63].

B Broader Impacts

This paper is intended to advance Machine Learning and AI research, with a special emphasis on the reasoning ability of LLMs—an essential component of autonomous AI systems. We identify key shortcomings in the reasoning capabilities of LLMs, especially in terms of aggregating individuals preferences over alternatives and algorithmic execution. We believe that the findings presented in this work can inform further research into AI systems to enhance their ability to act independently in complex decision-making environments.

C Preference-Based Algorithms in Matching Markets

C.1 Algorithm for Generating a Stable Matching

As shown in Algorithm 1, the standard deferred-acceptance algorithm runs by having the proposing side of the market make a series of proposals, and each agent that receives at least one proposal decides which proposal to accept (the proposal becomes an engagement), and which proposals to reject (or engagements to break). This continues until all agents are matched, which requires $O(n^2)$ proposal steps. The resulting solution is stable [27]. To describe the algorithm, we adopt the traditionally used setting of *stable-marriage* where the proposing side consists of *men* and the receiving side consist of *women*.

Algorithm 1 The Deferred Acceptance Algorithm

assign each agent $m \in M$ and $w \in W$ to be free
while there exists a free man m who has not proposed to every woman **do**
  $w \leftarrow$ highest-ranked woman on m's preference list to whom he has not yet proposed
  m proposes to w
  if w is free **then**
    w tentatively accepts m
  else if w prefers m to her current partner m' **then**
    w rejects m' and tentatively accepts m
    m' becomes free
  else
    w rejects m
  end if
end while
return the set of engaged pairs, these form a stable matching

Another commonly used version of the DA algorithm involves the receiver agent (in a given proposal) removing all agents (on the proposer-side) from their preference list who are ranked below the current proposer agent (who also remove the receiver from their respective preference lists). Due to the shortening of the preference lists as the algorithm progresses (see Algorithm 2), this is referred to as the version of the DA algorithm *with Shortlists* [37]. While this version terminates with at most as many (and often less) proposal steps as compared to Algorithm 1, it requires repeated updates to the original preference lists.

Algorithm 2 The Deferred-Acceptance Algorithm with Shortlists

```
assign each agent m ∈ M and w ∈ W to be free
while some man m is free do
    w = first woman on m's preference list
    m proposes and becomes engaged to w
    if some man p is engaged to w then
        break the engagement (p, w), assign p to be free
    end if
    for each m′ in w's preference list s.t. m ≻_w m′ do
        remove m′ and w from each other's preferences
    end for
end while
return the set of engaged pairs, these form a stable matching
```

C.2 Algorithm for Generating a Stable Matching w/ Master-Lists

Similar to Algorithm 1, Algorithm 3 runs in rounds of proposals. Since there is a Master-list over proposing agents in the preference lists of receiving agents, and proposing agents are selected to make proposals in the order in which they appear in the Master-list, there are no rejections (since any receiver receives the best possible proposal at any step) [38]. Hence, this algorithm terminates in n proposal steps (and is therefore easier to execute).

Algorithm 3 The Deferred-Acceptance Algorithm for Preferences w/ Master-lists on One Side

```
assign each agent m ∈ M and w ∈ W to be free
L ← Master-list over men.
for next free man m in L do
    w = first woman on m's preference list
    m proposes and becomes engaged to w
end for
return the set of engaged pairs, these form a stable matching
```

C.3 Algorithm for Resolving Instability

While we don't explicitly describe the algorithm here, the mechanism presented by Abeledo and Rothblum [1] can be applied to an unstable matching μ by resolving blocking pairs, resulting in a stable solution. Intuitively, an LLM does not have to follow a specific mechanism, rather the model can resolve instability by iteratively resolving blocking pairs as they arise (eventually, assuming all steps are correct, the model should arrive at a stable solution).

C.4 Algorithm for Detecting Instability

Intuitively, Algorithm 4 works by iteratively visiting each pair of agents (m, w) s.t. $m \in M$ and $w \in W$, and finding a pair such that either m prefers w to their current partner, or w prefers m to their current partner (when such a pair is found, output it as a blocking pair). If no pair (m, w) is found to be a blocking pair, then the solution is stable.

https://doi.org/10.52202/085713-0639

Algorithm 4 Stability Detecting Algorithm

for each $(m, w) \in \mu$, where $m \in M$ and $w \in W$
for man $m \in M$ **do**
  for man $w \in W$ **do**
    if $m \succ_w \mu(w)$ and $w \succ_m \mu(m)$ **then**
      output the identified blocking pair (m, w)
    end if
  end for
end for
output that there exist no blocking pairs

Table 4: Percentage of stable solutions returned by LLMs when provided with the DA algorithm in the prompt (With) as compared to the case when not provided (Without).

		Basic LLMs				**Reasoning LLMs**				**Advanced Reasoning LLMs**					
		Gemini-2.0-Flash		**Llama-3.3-70B**		**Qwen-QwQ-32B**		**DeepSeek-70B**		**o3-mini**		**DeepSeek-R1**		**Gemini-2.5-Pro**	
Difficulty	**Preference**	**Without**	**With**	**Without**	**With**	**Without**	**With**	**Without**	**With**	**Without**	**With**	**Without**	**With**	**Without**	**With**
Easy	**IC**	6	**10**	**2**	0	76	**84**	70	**74**	**100**	98	**100**	96	98	**100**
	ML	**8**	6	2	**6**	**90**	88	72	**94**	96	**100**	98	**100**	98	98
Medium	**IC**	0	0	0	0	**14**	2	0	0	**68**	64	42	**44**	**90**	88
	ML	0	0	0	0	34	**40**	14	12	80	**86**	**86**	82	88	**94**
Hard	**IC**	0	0	0	0	0	0	0	0	0	0	0	0	**8**	6
	ML	0	0	0	0	0	0	0	0	0	0	**54**	36	**40**	38
Average		2.33	**2.67**	0.67	**1.00**	35.67	35.67	26.00	**30.00**	57.33	**58.00**	**63.33**	59.66	68.33	**70.66**

D Prompt Engineering

Providing Algorithmic Description. In Table 4, examine LLMs' performance when provided with a prompt containing pseudocode for the DA algorithm, and compare it to the case when no algorithm is provided in the prompt. While providing the DA algorithm as part of the prompt leads to an improvement in some cases, the only case in which there is improvement is statistically significant[8] is with DeepSeek-70B at the Easy level with ML instances.

Reasoning-enhancement Prompts. For models that fail at generating stable matchings even with small instances, we evaluate whether prompt-based enhancements such as Chain-of-Thought (CoT) [72] and Few-shot prompting [19] can improve their performance. In particular, we introduce the following three types of modifications to the original prompt (used in Section 3):

- **CoT-Vanilla (CoT-V):** The steps of execution of the DA algorithm (see Algorithm 1) are provided for an example instance. Each step consists of a single (free) proposing agent making his next proposal, and the receiving agent either accepting or rejecting the proposal based on their current status.
- **CoT-Shortlist (CoT-SL):** This version of CoT uses the Shortlist version of the DA algorithm (see Algorithm 2) which reduces the overall number of proposal steps, but requires repeated updates to the original preference lists.
- **Few-shot Examples:** As opposed to the previous two cases, we provide LLMs with three examples of stable matching instances accompanied by their solutions.

To limit the context size of the prompt, we consider examples with $n = 5$ for each of these prompt modifications.

As depicted in Table 5, these prompting enhancements fail to improve the ability of LLMs to generate stable matchings. While there is a slight improvement for models like Qwen-QwQ-32B and DeepSeek-70B instances with Master-list preferences and size $n = 10$, this improvement is not statistically significant.

[8] At $p < 0.05$, using Fisher's exact test

[9] All models considered here are never able to generate stable matchings at the Hard difficulty level, with any prompting method.

Table 5: Percentage of stable solutions returned when prompt-enhancement strategies are used, as compared to the case without, for the Easy and Medium difficulty levels. [9]

	Model	Gemini-2.0-Flash				Llama-3.3-70B				Qwen-QwQ-32B				DeepSeek-70B			
Size	Culure	None	CoT-V	CoT-SL	Few-shot	None	CoT-V	CoT-SL	Few-shot	None	CoT-V	CoT-SL	Few-shot	None	CoT-V	CoT-SL	Few-shot
10	**IC**	**6**	2	0	0	0	**2**	0	0	**76**	86	84	**90**	**70**	60	60	68
	ML	**8**	4	4	2	2	0	**10**	6	90	**94**	92	**94**	72	**76**	68	**76**
20	**IC**	0	0	0	0	0	0	0	0	**14**	2	6	6	0	0	0	0
	ML	0	0	0	0	0	0	0	0	34	36	34	**42**	**14**	10	4	2

Table 6: Percentage of stable solutions from two LLMs when the task is framed as the *stable marriage* and the *task-scheduling problem*.

	Model	Gemini-2.0-F		o3-mini	
Difficulty	Preference	Stable Marriage	Task Scheduling	Stable Marriage	Task Scheduling
10	**IC**	6	8	100	98
	ML	8	6	68	50
20	**IC**	0	0	0	0
	ML	0	0	100	98
50	**IC**	0	0	80	72
	ML	0	0	0	0

Modified Problem Setting. We consider the traditionally used setting of *stable-marriage*, where the set M consists of *men* who propose to *women* in the set W [26]. To measure whether LLMs are sensitive to the nomenclature used to described the two-sided matching market, we also consider a different setting, i.e. that where a set of *workers* (W) are to be assigned a set of *tasks* (T) (and members on both sides have preferences over members of the other). We test the difference in the performance of two LLMs—Gemini-2.0-Flash and o3-mini—between the task-scheduling and stable-marriage settings. The results are provided in Table 6. While there is a slight decrease in performance for o3-mini, the change is not significant (at $p < 0.05$). This provides further evidence that LLMs understand requirements of computing stable solutions for matching markets, in general.

E Generating Stable Solutions for Preferences with Ties

As demonstrated in Section 3, reasoning-enabled models achieve significantly higher accuracy on Easy instances when compared to non-reasoning baseline models, but suffer dramatic drops in performance with Hard instances. A natural extension of this is to examine how introducing ties to preferences impacts an LLM's ability to generate stable solutions.

New notions of stability. When ties are introduced to preference profiles, additional (stronger) notions of stability exist. Namely, in addition to the standard notion of *weak stability* (where a matching does not admit any weak blocking pairs: agents who strictly prefer each other to their current partners), we have the added notions of *strong* and *super stability*, where a matching does not admit any strong or super blocking pairs, respectively. A strong blocking pair is a pair of agents in which one agent strictly prefers the other agent over their current partner, and the other agent remains indifferent or prefers the first agent and their current partner. A super blocking pair is a pair of agents in which either agent is either indifferent the other agent and their current partner, or prefers the other agent to their current partner. *Super stability*, the strongest notion of stability, implies *strong stability*, which in turn implies *weak stability*. Additionally, it is important to note that *strongly* and *super stable* solutions do not exist for all preference profiles.

Irving [36] provides three algorithms (one for each) to compute *weakly*, *strongly*, and *super stable* matchings for preference profiles with ties. The algorithm for *weak* stability takes $\mathcal{O}(n^2)$ steps (the algorithm is equivalent to running standard the DA algorithm with arbitrary tie-breaking), the algorithm for *strong* stability takes $\mathcal{O}(n^4)$ steps, and the algorithm for *super* stability takes $\mathcal{O}(n^2)$ steps (these two algorithms require more demanding steps with complex operations).

Generating matchings for each stability notion. To evaluate how LLMs handle preferences with ties, we perform the same experiment as in Section 3 with Easy preference profiles and Gemini-2.5-Pro, except we introduce multiple ties of random sizes, at randomly selected starting positions, in each preference list in the profile. Additionally, we modify the prompt listed in Appendix J.1 to get a total of three different prompts. The first prompt (referred to as the *Baseline Prompt*) is identical to the prompt in Appendix J.1. The other two prompts (referred to as the *Strong*

https://doi.org/10.52202/085713-0639

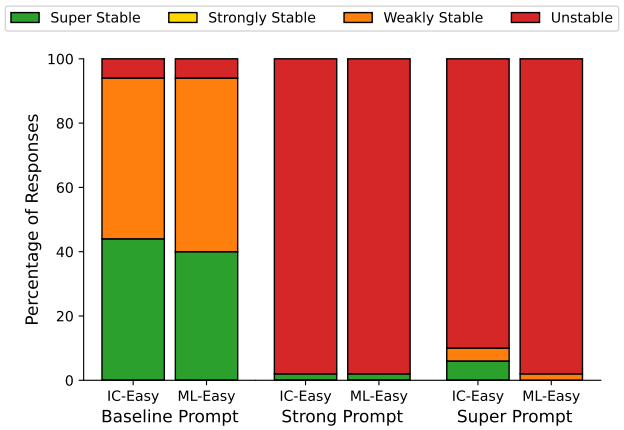

Figure 7: The generated responses by Gemini-2.5-Pro for Master-list (ML) and Impartial Culture (IC) Easy preferences with different prompts. All generated matchings were valid and complete. Again, *unstable* indicates one-to-one matchings with *weak* blocking pairs present.

Prompt and *Super Prompt*) replace the instruction to "...find the proposer-optimal stable matching..." with "...find the proposer-optimal STRONG stable matching..." and "...find the proposer-optimal SUPER stable matching...", respectively. For each prompt, we run the experiment and count the proportion of generated matchings that are *super*, *strong*, and *weakly stable*, as well as unstable matchings.

Figure 7 demonstrates the performance of Gemini-2.5-Pro in generating stable outcomes with respect to each of the three new notions of stability for preference profiles with ties. Gemini-2.5-Pro once again demonstrates a high accuracy (96%) in generating stable solutions (considering *weak*, *strong*, and *super stability*) with the Baseline Prompt, with 50% and 54% of stable solutions being *weakly stable* with the IC-Easy and ML-Easy preference profiles, respectively. However, only around 40% of generated solutions were super (and strongly) stable for both preference cultures. Gemini-2.5-Pro's reasoning traces for each instance indicate that the model essentially attempts to execute the DA algorithm when generating stable solutions even when ties are present (rather than utilizing the more complex algorithms for *strong* and *super stability*, even when explicitly prompted). Additionally, any generated *super* stable solutions are a result of an instance's *super* stable solution intersecting with its *weakly* stable solution. Interestingly, specifying the notion of stability appears to significantly hurt the model's ability to generate any stable solutions. While one potential explanation is that specifying the desired stability notion in the prompt introduces unnecessary noise, this is an interesting avenue for future study.

F Resolving Instability: Additional Material

F.1 Generating Unstable Matchings

Here we describe the procedures we use to generate the two types of unstable matchings we consider.

One-BP. We generate a matching that contains a single blocking pair, by swapping the partners of two randomly selected proposer agents in the Optimal matching. Since such a swap may lead to more than one blocking-pair (or no blocking pairs), we perform this process (for every instance) until we obtain a matching with exactly one blocking-pair. This procedure is formally described in Algorithm 5.

Random. A Random matching is simply generated by generating a random permutation of agents on one side and assigning agents such a list to the agents on the other side, one-by-one.

Algorithm 5 GENERATEONEBPMATCHING

Require:
$\Pi = (\succ_m, \succ_w)$ ▷ Preference profile for all men $m \in M$ and women $w \in W$
μ^* ▷ Men-optimal stable matching returned by Deferred Acceptance
Ensure:
A matching μ containing *exactly one* blocking pair

function GENERATEONEBPMATCHING(Π, μ^*)
  repeat ▷ Keep trying until the condition is met
    $\mu \leftarrow \text{copy}(\mu^*)$ ▷ Start from the stable matching
    $(m_a, m_b) \leftarrow$ arbitrary pair $m_a, m_b \in M$ s.t. $m_a \neq m_b$
    $w_a \leftarrow \mu(m_a)$
    $w_b \leftarrow \mu(m_b)$ ▷ Swap partners of the two men
    $\mu(m_a) \leftarrow w_b, \;\; \mu(w_b) \leftarrow m_a$
    $\mu(m_b) \leftarrow w_a, \;\; \mu(w_a) \leftarrow m_b$
  until $|\text{BLOCKINGPAIRS}(\mu, \Pi)| = 1$ ▷ Stop when exactly one blocking pair exists
  return μ
end function

function BLOCKINGPAIRS(μ, Π)
  $B \leftarrow \emptyset$
  for all $m \in M$ **do**
    for all $w \in W$ **do**
      if $w \succ_m \mu(m)$ **and** $m \succ_w \mu(w)$ **then**
        $B \leftarrow B \cup \{(m, w)\}$
      end if
    end for
  end for
  return B
end function

Table 7: Instability Rate (averaged across instances) in the (valid) matchings returned by LLMs when asked to resolve a given unstable matching of types One-BP and Random. The column "Baseline" contains the (average) Instability Rate for the provided matching of the indicated type. Numbers in bold indicate that the Instability Rate of the corrected solution is significantly worse than the provided matching. A $*$ on the number in the One-BP column indicates a that Instability Rate is significantly lower than the case when a Random matching is provided (at $p < 0.05$).[10]

	Basic LLMs				Reasoning LLMs				Advanced Reasoning LLMs							
	Gemini-2.0-Flash		Llama-3.3-70B		Qwen-QwQ-32B		DeepSeek-70B		o3-mini		DeepSeek-R1		Gemini-2.5-Pro		Baseline	
Difficulty	One-BP	Random	One-BP	Random	One-BP	Random	One-BP	Random	One-BP	Random	One-BP	Random	One-BP	Random	One-BP	Random
Easy	**45.5**	46.3	**34.25***	42.05	4.30*	10.80	8.74	9.05	0.25	0.00	0.20	0.20	0.30	0.80	10.00	77.32
Medium	**55.09**	58.81	**57.48***	64.12	**14.50***	49.20	**23.39***	43.84	3.92	3.25	**10.65**	12.58	2.17	3.5	5.00	87.95
Hard	**49.99***	74.66	**61.42***	86.79	**17.44***	86.91	**35.38***	84.76	**43.77***	59.98	**14.8***	47.81	**20.99***	25.08	2.00	94.52

F.2 Further Results

Measuring Instability after Repair. The extent to which a given matching is incorrect does influence quality of the solutions returned by LLMs after being asked to correct it. Table 7 shows that the number of blocking pairs is often significantly lower when the unstable matching (that LLMs are asked to correct) has a single blocking pair, as compared to the case with random matchings.[11] Similarly, as shown in Figure 8, matchings returned after resolving an almost stable matching have a greater overlap with the Optimal solution (especially at Medium and Hard difficulty levels).

[10] All statistical comparisons in this table are made using Welch's t-test [73].

[11] In fact, asking LLMs to resolve a random matching leads to a significantly higher number of blocking pairs in the returned solution, as compared to the case when they are asked to generate solutions from scratch.

https://doi.org/10.52202/085713-0639

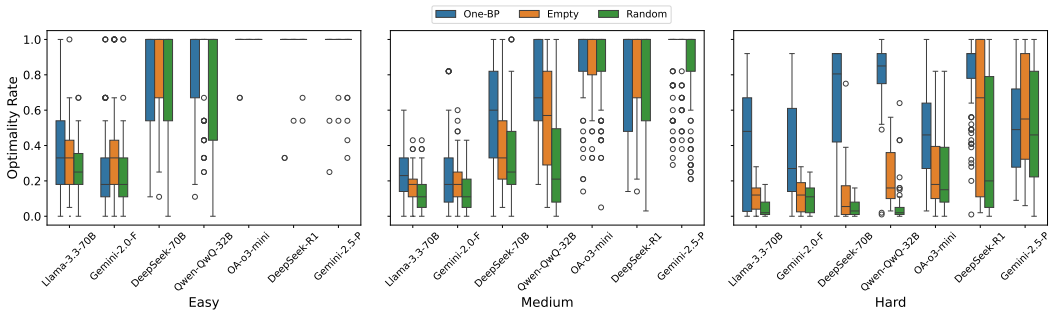

Figure 8: Optimality Rate when LLMs are asked to resolve a matching with a single blocking pair introduced into the Optimal solution (One-BP), or a randomly generated matching (Random). These are is compared to the case when they are asked to generate a stable matching from scratch (Empty).

G Additional Material about Detecting Instability

G.1 Comparing Impartial Culture and Master-List Instances

Generally, the classification of a preference profile as an impartial culture or Master-list instance has a relatively small impact on the ability of an LLM to detect instability in the instance. However, we observe some differences between the ability of certain LLMs to detect stable/unstable matchings with ML instances compared to IC instances. Models such as DeepSeek-14B, o3-mini, and DeepSeek-70B are able correctly detect stable solutions significantly more frequently with ML instances than with IC instances. A potential explanation for this is that Master-list preferences contain fewer *unique* preference lists, decreasing the chances that the model hallucinates blocking pairs. On the other hand, models such as Gemini-2.0-Flash and Llama-3.3-70B correctly identify unstable solutions significantly more often with IC preferences as compared to the case with ML preferences. The intuition for this observation is the opposite: with impartial culture preferences, there is a higher probability of having blocking pairs, therefore models that tend to predict that solutions are unstable will perform better with impartial culture instances.

H Fine-tuning Details

Models. We fine-tune four reasoning models:

- DeepSeek-8B (`deepseek-ai/DeepSeek-R1-Distill-Llama-8B`),
- DeepSeek-14B (`deepseek-ai/DeepSeek-R1-Distill-Qwen-14B`),
- Qwen-QwQ-32B (`Qwen/QwQ-32B`), and
- Qwen3-32B (`Qwen/Qwen3-32B`)

using the Unsloth[12] framework with parameter-efficient tuning (LoRA).[13]

Dataset. The dataset for the Generation task contains $N = 10000$ samples for which the reasoning trace is generated using a Python implementation of the DA algorithm. For the Preference Reasoning task, the dataset consists of $N = 9000$ samples (3000 for each question level), where the reasoning trace involves explicitly identifying the positions of agents in the concerned preferences. In both datasets, we include an equal number of IC and ML instances, with sizes ranging from $n = 5$ to $n = 50$. Detailed examples of training examples for both tasks (Generation and Preference Reasoning) are provided in Appendix K.

Model Setup. We used the `FastLanguageModel.from_pretrained` interface from Unsloth to load the base model with a maximum sequence length of 10,000 tokens. The model was loaded in full

[12] https://unsloth.ai/

[13] Since results for Qwen-QwQ-32B and Qwen3-32B are similar, we show only those for the former in Table 2.

precision (no quantization) and fine-tuned using Low-Rank Adaptation (LoRA) with the following settings:

- Rank (r): 32
- Target Modules: `q_proj`, `k_proj`, `v_proj`, `o_proj`, `gate_proj`, `up_proj`, `down_proj`
- LoRA α: 32
- LoRA Dropout: 0
- Bias: `none`
- Gradient Checkpointing: Enabled via `use_gradient_checkpointing="unsloth"`

Training Configuration. Fine-tuning was conducted using the `SFTTrainer` from the TRL library with the following training arguments:

- Epochs: 1
- Batch size per device: 2 (1, for Qwen-QwQ-32B)
- Gradient accumulation steps: 4 (2, for Qwen-QwQ-32B)
- Learning rate: 2×10^{-4} with a linear scheduler and 5 warmup steps
- Optimizer: `AdamW-8bit`
- Weight decay: 0.01
- Precision: Mixed precision (FP16 or BF16, based on hardware support)
- Seed: 3407

Hardware. Each model was fine-tuned using a single NVIDIA H100 GPU (80GB RAM) with CUDA support; model and inputs were explicitly transferred to GPU for inference and training.

Model Saving and Sharing. The resulting models were uploaded to the Hugging Face Hub and will be released upon acceptance.

Inference Setup. After fine-tuning, the model was evaluated using in-context inference. Inputs were formatted similarly to training prompts, and the model's output was parsed to extract the JSON-formatted matching solution.

Ablation tests. Table 8 illustrates how performance is sensitive to variations in data-related parameters such as the types of instances included in the data, range of instance sizes, and number of training examples, as well as training-related parameters such as the LoRA rank and the base model used. The primary observation from these tests is that easier training instances are more crucial that harder ones. For example, performance is significantly better when the training data consists of only Easy (and smaller) instances, i.e. with $n = 10$, as compared to the case with only Hard instances ($n = 50$). Similarly, performance is much better when the data consists of only ML instances, as opposed to the case where it consists of only IC instances, which are more difficult than the former.

I Inference Details

Fine-tuned Models. The models we fine-tune have been pushed to HuggingFace Hub and will be released upon acceptance.

Inference Configuration. For the task of generating stable solutions, inference was performed on each of the open-source models such as DeepSeek-8B, DeepSeek-14B, Qwen-QwQ-32B, DeepSeek-70B, and Llama-3.3-70B, with the following sampling parameters:

- Temperature: 0.5
- Maximum tokens: 30,000

https://doi.org/10.52202/085713-0639

Table 8: Performance (percentage of stable solutions generated) after fine-tuning, for DeepSeek-8B and Qwen3-32B, with different configurations of data- and training-related hyper-parameters. Configuration 1 is the default configuration. The variation in each other configuration (as compared to config. 1) is in bold.

Config.	Instance types	Instance sizes (range)	Training set size	LoRA rank	Base model	ML			IC			Total
						Easy	Medium	Hard	Easy	Medium	Hard	
1	ML, IC	[5,50]	10000	32	**DeepSeek-8B**	94	60	0	34	2	0	31.67
					Qwen3-32B	100	98	4	98	84	0	64
2	**ML**	[5,50]	10000	32	**DeepSeek-8B**	100	94	0	2	0	0	32.67
					Qwen3-32B	100	100	0	6	0	0	34.33
3	**IC**	[5,50]	10000	32	**DeepSeek-8B**	0	0	0	0	0	0	0
					Qwen3-32B	0	0	0	0	2	0	0.33
4	ML, IC	[5,50]	10000	**64**	**DeepSeek-8B**	96	82	0	2	0	0	30
					Qwen3-32B	100	100	0	100	100	0	66.67
5	ML, IC	[5,50]	**5000**	32	**DeepSeek-8B**	74	46	0	0	0	0	20
					Qwen3-32B	100	94	0	98	92	0	64
6	ML, IC	**[5,10]**	**5000**	32	**DeepSeek-8B**	100	2	0	100	8	0	35
					Qwen3-32B	100	52	0	100	26	0	46.33
7	ML, IC	**[50,50]**	**5000**	32	**DeepSeek-8B**	0	0	0	0	0	0	0
					Qwen3-32B	12	0	0	2	0	0	2.33

Default values were utilized for all other sampling parameters. We used online APIs for the following models:

- Gemini-2.0-Flash ('gemini-2.0-flash')
- o3-mini ('o3-mini')
- DeepSeek-R1 ('deepseek-reasoner')
- Gemini-2.5-Pro ('gemini-2.5-pro-preview-03-25')

Hardware. All inference experiments with open-source models were run on NVIDIA H100 GPUs (80GB RAM) with CUDA support; model and inputs were explicitly transferred to GPU for inference and training. We used a single GPU for inference involving DeepSeek-8B and DeepSeek-14B, two GPUs for inference involving Qwen-QwQ-32B, and four GPUs for inference involving Llama-3.3-70B and DeepSeek-70B.

J Prompts

J.1 Example prompt for Generating Stable Solutions

Vanilla prompt. The prompt used for generating stable solutions with LLMs follows standard prompting procedures, by first outlining the task, providing appropriate context, specifying constraints, and detailing the desired output format (a JSON object). We intentionally provide preferences in a structured, tabular format. It enables us to isolate and rigorously evaluate the LLMs' reasoning, alignment, and solution quality relative to normative axioms (e.g., stability). Natural language formulations introduce significant noise in both input and output, making it difficult to attribute performance failures to reasoning versus parsing. By grounding our analysis in tabular settings first, we can obtain clean and interpretable measurements, forming a benchmark for future extensions that incorporate naturalistic input.

Notice that despite the deferred-acceptance algorithm never being mentioned in the prompt, all models mentioned the deferred-acceptance algorithm in their responses. As mentioned in Appendix D, we use the traditional setting of stable-marriage (where men propose to women) considered by Gale and Shapley [27] while describing the problem in the prompt.

You are an intelligent assistant who is an expert in algorithms. Consider the following instance of the two-sided matching problem, where 10 men are to be matched with 10 women. Here are the preference lists for all individuals:
<preferences>
{
M: {
M1: [W10,W1,W3,W6,W2,W4,W9,W8,W7,W5],
M2: [W8,W3,W10,W6,W2,W5,W4,W7,W1,W9],
...
M10: [W2,W5,W1,W3,W7,W6,W10,W4,W9,W8],
},
W: {
W1: [M2,M8,M9,M10,M5,M7,M1,M4,M6,M3],
W2: [M2,M7,M3,M1,M8,M9,M6,M10,M5,M4],
...
W10: [M6,M4,M7,M5,M8,M9,M10,M2,M3,M1],
}}
</preferences>
Your task is to find the proposer-optimal stable matching. You can use XML tags like <scratchpad> to explain your thought process while computing the solution.
Once you have found a stable matching, please return your matching in the JSON format given below:
<answer>
{
"M1": "<woman matched with M1>",
"M2": "<woman matched with M2>",
...
"M10": "<woman matched with M10>"
}
</answer>
Make sure that each man/woman is matched with exactly ONE partner. It is mandatory that you provide a matching as a JSON object enclosed in <answer></answer> tags as described above.

https://doi.org/10.52202/085713-0639

Providing Algorithmic Description. The following is the prompt is a modification of the vanilla prompt where the steps of the DA algorithm have been described to assist the model with implementing the same.

> You are an intelligent assistant who is an expert in algorithms.
> ...
> </preferences>
> Your task is to find the proposer-optimal stable matching. For this, you can use the Deferred Acceptance algorithm. The steps of this algorithm are described below:
> 1. Initialize all men and women as unmatched.
> 2. Create a list to keep track of each man's next proposal (initially set to 0 for all men).
> 3. While there are unmatched men:
> a. Select an unmatched man (M).
> b. Find the next woman (W) on M's preference list that he hasn't proposed to yet.
> c. If W is unmatched, match M and W.
> d. If W is matched but prefers M to her current partner:
> - Unmatch W from her current partner.
> - Match M and W.
> - Set the unmatched man as W's previous partner.
> e. If W rejects M, move to the next woman on M's preference list.
> 4. Repeat step 3 until all men are matched.
> You can use XML tags like <scratchpad> to explain your thought process ...
> ...
> It is mandatory that you provide a matching as a JSON object enclosed in <answer></answer> tags as described above.

Modified Problem Setting. The following is a modification to the vanilla prompt, where the setting of *task-allocation* (assigning tasks to workers) is considered instead of the *stable-marriage* setting. We replace *men* with *workers* and *women* with *tasks*.

> You are an intelligent assistant who is an expert in algorithms. Consider the following instance of the two-sided matching problem, where 5 workers are to be assigned with 5 tasks, and each worker is assigned exactly one task.
> Here are the preference lists for all workers (W) over tasks (T) and the preferences of tasks over workers:
> <preferences>
> {
> W: {
> W1: [T5, T3, T4, T2, T1]
> ...
> }
> T: {
> T1: [W3, W5, W4, W1, W2]
> ...
> }}
> </preferences>
> Your task is to find a stable matching of workers and tasks. You can use XML tags like <scratchpad> to explain your thought process while computing the solution.
> Once you have found a stable matching, please return your matching in the JSON format given below:
> <answer>
> {
> "W1": "<task assigned to W1>",
> ...
> "W5": "<task assigned to W5>"
> }
> </answer>
> Make sure that each worker is assigned exactly ONE task. It is mandatory that you provide a matching as a JSON object enclosed in <answer></answer> tags as described above.

J.2 Example Prompts for Prompt Engineering

J.2.1 CoT-Vanilla

Chain-of-Thought methods were applied to the prompt in Appendix J.1 by additionally including an example trace of steps performed when running the deferred-acceptance algorithm on a randomly generated instance. The algorithm trace includes all proposals, all respective acceptances/rejections, and the resultant stable solution. The entire Chain-of-Thought example is enclosed within <example> XML tags.

You are an intelligent assistant who is an expert in algorithms. Your task is to find the proposer-optimal stable matching, for the two-sided matching problem. Here is an example to demonstrate how you should proceed:
<example>
<preferences>
{
M: {
M1: [W5,W1,W2,W4,W3],
M2: [W1,W2,W5,W4,W3],
M3: [W4,W2,W3,W1,W5],
M4: [W5,W1,W2,W4,W3],
M5: [W3,W5,W4,W2,W1],
},
W: {
W1: [M2,M3,M5,M4,M1],
W2: [M5,M2,M4,M3,M1],
W3: [M2,M1,M3,M5,M4],
W4: [M1,M4,M5,M3,M2],
W5: [M4,M3,M5,M2,M1],
}}
</preferences>
M1 is free. M1 proposes to W5
Since W5 is free, W5 accepts the proposal. Now M1 and W5 are matched.
M2 is free. M2 proposes to W1
Since W1 is free, W1 accepts the proposal. Now M2 and W1 are matched.
M3 is free. M3 proposes to W4
Since W4 is free, W4 accepts the proposal. Now M3 and W4 are matched.
M4 is free. M4 proposes to W5
Since W5 prefers M4 to their current partner M1, W5 accepts the proposal. Now M4 and W5 are matched, and M1 is free.
M1 is free. M1 proposes to W1
Since W1 prefers their current partner M2 to M1, W1 rejects the proposal. M2 and W1 are still matched, and M1 is still free.
M1 is free. M1 proposes to W2
Since W2 is free, W2 accepts the proposal. Now M1 and W2 are matched.
M5 is free. M5 proposes to W3
Since W3 is free, W3 accepts the proposal. Now M5 and W3 are matched.
<answer>
{
"M1": "W2",
"M2": "W1",
"M3": "W4",
"M4": "W5",
"M5": "W3"
}
</answer>
</example>
Consider the following instance of the two-sided matching problem, where 10 men are to be matched with 10 women . . .

https://doi.org/10.52202/085713-0639

J.2.2 CoT-Shortlist

The main distinction between the prompt described here and the one in Appendix J.2.1 lies in how the algorithm's execution is detailed. In the CoT-Shortlist prompt, the provided algorithm trace includes an additional step: agents remove each other from their respective shortlists if they become matched with a partner they find more desirable than the other agents on their list. All other aspects of the prompt are identical to the CoT-Vanilla prompt.

You are an intelligent assistant who is an expert in algorithms. Your task is to find the proposer-optimal stable matching, for the two-sided matching problem. Here is an example to demonstrate how you should proceed:
<example>
<preferences>
{
M: {
M1: [W4,W3,W5,W2,W1],
M2: [W5,W4,W3,W1,W2],
M3: [W5,W4,W1,W2,W3],
M4: [W5,W4,W2,W1,W3],
M5: [W2,W4,W5,W3,W1],
},
W: {
W1: [M5,M2,M3,M4,M1],
W2: [M3,M4,M5,M1,M2],
W3: [M4,M1,M2,M5,M3],
W4: [M5,M1,M4,M3,M2],
W5: [M1,M4,M5,M3,M2],
}}
</preferences>
M1 is free. M1 proposes to W4. W4 accepts the proposal. Now M1 and W4 are matched.
W1 deletes M4, M3, M2 from her list. M4, M3, M2 delete W4 from their list.
M2 is free. M2 proposes to W5. W5 accepts the proposal. Now M2 and W5 are matched.
M3 is free. M3 proposes to W5. W5 accepts the proposal. Now M3 and W5 are matched.
W5 prefers M3, so W5 breaks her engagement with M2.
W3 deletes M2 from her list. M2 delete W5 from their list.
M4 is free. M4 proposes to W5. W5 accepts the proposal. Now M4 and W5 are matched.
W5 prefers M4, so W5 breaks her engagement with M3.
W4 deletes M5, M3 from her list. M5, M3 delete W5 from their list.
M5 is free. M5 proposes to W2. W2 accepts the proposal. Now M5 and W2 are matched.
W5 deletes M1, M2 from her list. M1, M2 delete W2 from their list.
<answer>
{
"M1": "W4",
"M2": "W3",
"M3": "W1",
"M4": "W5",
"M5": "W2"
}
</answer>
</example>
Consider the following instance of the two-sided matching problem, where 10 men are to be matched with 10 women . . .

J.2.3 Few-shot Examples

Few-shot prompting was applied to the prompt in Appendix J.1 by additionally including a series of randomly generated preference/stable solution pairs. As with other few-shot prompting strategies, the model is then asked to generate a stable solution (as shown in Appendix J.1). As with the CoT methods, each sample preference/stable solution pairs is enclosed in <example> XML tags.

You are an intelligent assistant who is an expert in algorithms. Your task is to find the proposer-optimal stable matching, for the two-sided matching problem. Here is an example to demonstrate how you should proceed:
<example>
<preferences>
{
M: {
M1: [W5,W3,W4,W2,W1],
M2: [W3,W4,W1,W2,W5],
M3: [W5,W1,W4,W2,W3],
M4: [W3,W2,W5,W1,W4],
M5: [W3,W4,W2,W1,W5],
},
W: {
W1: [M1,M4,M3,M5,M2],
W2: [M2,M4,M5,M1,M3],
W3: [M1,M2,M4,M5,M3],
W4: [M3,M5,M1,M4,M2],
W5: [M5,M3,M4,M2,M1],
}}
</preferences>
<answer>
{
"M1": "W3",
"M2": "W1",
"M3": "W5",
"M4": "W2",
"M5": "W4"
}
</answer>
</example>
<example>
...
</example>
<example>
...
</example>
Consider the following instance of the two-sided matching problem, where 10 men are to be matched with 10 women . . .

J.3 Example Prompt for Evaluating Stability

The following prompt requires LLMs to determine if a given solution to a provided preference profile is stable. Unlike the prompt in Appendix J.1, the only element that the LLM must include in their response is a binary response (yes/no).

https://doi.org/10.52202/085713-0639

Consider the following instance of the two-sided matching problem, where 5 men are to be matched with 5 women.
Here are the preference lists for all individuals:
<preferences>
{
M: {
M1: [W5,W3,W4,W2,W1],
...
},
W: {
W1: [M3,M5,M4,M1,M2],
...
}}
</preferences>
Your task is to determine whether the following matching is stable or not.
<matching>
[[M1, W4],[M2, W5],[M3, W3],[M4, W1],[M5, W2],]
</matching>
Please return 'Yes' if you think the provided matching is stable and 'No' if you think it is unstable, and enclose your answer in <answer></answer> tags.

J.4 Example Prompts for Preference Comprehension

In each of the following preference comprehension prompts, models are asked to provide the name of an agent (in level-1) or to provide a binary answer (yes/no for levels 2 and 3) in response to a provided question. In addition to changing the preference profiles for each instance of a preference comprehension task, the agents and positions mentioned in the question are also changed with each instance. For details about each level of preference comprehension, view Section 6.

J.4.1 Level-1

Your goal is to correctly interpret the given preference lists and answer a specific question about agent preferences.
First, here are the preference lists for all individuals:
<preferences>
{
M: {
M1: [W5,W3,W4,W2,W1],
...
},
W: {
W1: [M3,M5,M4,M1,M2],
W2: [M1,M3,M4,M5,M2],
...
}}
</preferences>
Now, you will be asked a specific question about agent preferences:
<question>
Who is agent W2's, 1-most preferred agent?
</question>
Once you have determined the answer, provide your output in the following format:
1. The solution as a single agent name. For example, "W1"
Present your final answer within <answer> tags.
IMPORTANT: ONLY RETURN THE NAME OF THE SINGLE AGENT THAT IS THE ANSWER TO THE QUESTION. Do not include any explanations or additional information in your final answer.

J.4.2 Level-2

You are an AI assistant tasked with analyzing preference profiles in a two-sided matching problem with one-to-one solutions. Your goal is to correctly interpret the given preference lists and answer a specific question about agent preferences.
First, here are the preference lists for all individuals:
<preferences>
{
M: {
M1: [W5,W3,W4,W2,W1],
...
},
W: {
W1: [M3,M5,M4,M1,M2],
...
}}
</preferences>
Now, you will be asked a specific question about agent preferences:
<question>
Would agent W1, prefer M3 or M2 over M4?
</question>
Once you have determined the answer, provide your output in the following format:
1. The solution as a YES or a NO. For example, "NO"
Present your final answer within <answer> tags.
IMPORTANT: ONLY RETURN YES OR NO THAT IS THE ANSWER TO THE QUESTION. Do not include any explanations or additional information in your final answer.

J.4.3 Level-3

You are an AI assistant tasked with analyzing preference profiles in a two-sided matching problem with one-to-one solutions. Your goal is to correctly interpret the given preference lists and answer a specific question about agent preferences.
First, here are the preference lists for all individuals:
<preferences>
{
M: {
M1: [W5,W3,W4,W2,W1],
...
},
W: {
W1: [M3,M5,M4,M1,M2],
...
}}
</preferences>
Now, you will be asked a specific question about agent preferences:
<question>
If agent W1 is currently engaged to M4, would she accept proposals from M3 or M2?
</question>
Once you have determined the answer, provide your output in the following format:
1. The solution as a YES or a NO. For example, "NO"
Present your final answer within <answer> tags.
IMPORTANT: ONLY RETURN YES OR NO THAT IS THE ANSWER TO THE QUESTION. Do not include any explanations or additional information in your final answer.

https://doi.org/10.52202/085713-0639

J.5 Example Prompt for Resolving Instability

For the task of resolving instability in a given unstable solution, the prompt begins by providing models with the instance's preference profile (as with the prompts for the other tasks). In addition, the prompt includes an unstable matching, and asks the model to resolve the instability by outputting a stable solution (in an identical format to the prompt in Appendix J.1).

You are an intelligent assistant who is an expert in algorithms. Consider the following instance of the two-sided matching problem and respective unstable matching, where 5 men are to be matched with 5 women.
Here are the preference lists for all individuals:
<preferences>
{
M: {
M1: [W5,W3,W4,W2,W1],
M2: [W2,W3,W5,W1,W4],
M3: [W5,W3,W1,W4,W2],
M4: [W1,W3,W2,W5,W4],
M5: [W2,W3,W4,W1,W5],
},
W: {
W1: [M3,M5,M4,M1,M2],
W2: [M1,M3,M4,M5,M2],
W3: [M3,M2,M4,M1,M5],
W4: [M4,M2,M3,M5,M1],
W5: [M2,M4,M5,M1,M3],
}}
</preferences>
Here is an unstable matching.
<answer>
{
"M1": "W4",
"M2": "W5",
"M3": "W3",
"M4": "W2",
"M5": "W1"
}
</answer>
Your task is to modify the given unstable matching to make it equivalent to the proposer-optimal stable matching. You can use XML tags like <scratchpad> to explain your thought process while computing the solution.
Once you have found a stable matching, please return your matching in the JSON format given below:
<answer>
{
"M1": "<woman matched with M1>",
"M2": "<woman matched with M2>",
"M3": "<woman matched with M3>",
"M4": "<woman matched with M4>",
"M5": "<woman matched with M5>"
}
</answer>
Make sure that each man/woman is matched with exactly ONE partner. It is mandatory that you provide a matching as a JSON object enclosed in <answer></answer> tags as described above.

J.6 Example Prompt for Repeated Queries Due to Missing JSON Object

For tasks where the desired output is a JSON object (when outputting a stable solution), models are given an additional opportunity to rectify issues in their response if the original response is incorrectly formatted. The prompt below is passed to the model to help rectify issues related to missing JSON objects. Note that the <initially passed prompt> and <last 3,000 characters of LLM's first response> XML tags are replaced by the initial prompt and the tail of the models initial response, respectively.

Previously, I gave you the following task:

<initially passed prompt>

In your response, you either failed to provide me with a matching or did not adhere to the JSON format I had asked for. Here are the last few lines of your response for reference:

<last 3,000 characters of LLM's first response>

Please correct your response and provide me with the matching in the following JSON format, enclosed in <answer></answer> tags.<answer>
{
"M1": "<woman matched with M1>",
"M2": "<woman matched with M2>",
"M3": "<woman matched with M3>",
"M4": "<woman matched with M4>",
"M5": "<woman matched with M5>"
}
</answer>
Make sure that each man/woman is matched with exactly ONE partner.

J.7 Example Prompt for Repeated Queries Due to Incorrectly Formatted JSON Object

Similar to the prompt in Appendix J.6, the following prompt is passed to the model when the model's initial response contains an incorrectly formatted JSON object. Once again, the <initially passed prompt> and <last 3,000 characters of LLM's first response> XML tags are replaced by the initial prompt and the tail of the models initial response, respectively.

Previously, I gave you the following task:

<initially passed prompt>

In your response, you failed adhere to the JSON format I had asked for. Here are the last few lines of your response for reference:

<last 3,000 characters of LLM's first response>

Please correct your response and provide me with the matching in the following JSON format, enclosed in <answer></answer> tags.<answer>
{
"M1": "<woman matched with M1>",
"M2": "<woman matched with M2>",
"M3": "<woman matched with M3>",
"M4": "<woman matched with M4>",
"M5": "<woman matched with M5>"
}
</answer>
Make sure that each man/woman is matched with exactly ONE partner.

https://doi.org/10.52202/085713-0639

J.8 Example Prompt for Repeated Queries Due to Incomplete Matching

Similar to the prompt in Appendix J.6, the following prompt is passed to the model when the model's initial response contains a correctly formatted JSON object, but the matching itself is incomplete, or some agents have multiple partners. After the initially passed prompt, note that additional details are provided to assist the LLM in rectifying its response. Again, the <initially passed prompt> and <last 3,000 characters of LLM's first response> XML tags are replaced by the initial prompt and the tail of the models initial response, respectively.

> Previously, I gave you the following task:
> ---
> <initially passed prompt>
> ---
> In your response, the matching you selected involves some women being matched with multiple men, which is not allowed. For example, W2 is matched with M1, M2, and M5. Additionally, W3, and W4 are unmatched. Here are the last few lines of your response for reference:
> ---
> <last 3,000 characters of LLM's first response>
> ---
> Please correct your response and provide me with the matching in the following JSON format, enclosed in <answer></answer> tags.<answer>
> {
> "M1": "<woman matched with M1>",
> "M2": "<woman matched with M2>",
> "M3": "<woman matched with M3>",
> "M4": "<woman matched with M4>",
> "M5": "<woman matched with M5>"
> }
> </answer>
> Make sure that each man/woman is matched with exactly ONE partner.

K Training Examples for Fine-tuning

K.1 System-prompt (s)

This is the first part of the input, and is common across all tasks.

> Below is an instruction that describes a task, paired with an input that provides further context.
> Write a response that appropriately completes the request.
> Before answering, think carefully about the question and create a step-by-step chain of thoughts to ensure a logical and accurate response.

K.2 High-level instruction (u)

- **Generating:**

  > ### Instruction:
  > You are an intelligent assistant who is an expert in algorithms. Your task is to find the proposer-optimal stable matching, for the two-sided matching problem.
  > ### Question:
  > Consider the following instance of the two-sided matching problem, where 5 men are to be matched with 5 women.
  > Here are the preference lists for all individuals:

- **Comprehension:**

Instruction:
You are an intelligent assistant who is an expert in algorithms. You will be given an instance of the two-sided matching problem, and will be asked to answer a question about the preferences of the agents involved.
Question:
First, here are the preference lists for all individuals:

K.3 Preference Profile ($p^{(i)}$)

<preferences>
{
M: {
M1: [W5,W3,W4,W2,W1],
M2: [W2,W3,W5,W1,W4],
M3: [W5,W3,W1,W4,W2],
M4: [W1,W3,W2,W5,W4],
M5: [W2,W3,W4,W1,W5],
},
W: {
W1: [M3,M5,M4,M1,M2],
W2: [M1,M3,M4,M5,M2],
W3: [M3,M2,M4,M1,M5],
W4: [M4,M2,M3,M5,M1],
W5: [M2,M4,M5,M1,M3],
}}
</preferences>
</preferences>

K.4 Task-prompt ($t^{(i)}$)

- **Generating:**

Your task is to find the proposer-optimal stable matching.
Once you have found a stable matching, please return your matching in the JSON format given below:
<answer>
{
"M1": "<woman matched with M1>",
"M2": "<woman matched with M2>",
"M3": "<woman matched with M3>",
"M4": "<woman matched with M4>",
"M5": "<woman matched with M5>"
}
</answer>
Make sure that each man/woman is matched with exactly ONE partner. It is important that you enclose your JSON object in <answer></answer> tags.

- **Comprehension (Level-1):**

https://doi.org/10.52202/085713-0639

Now, you will be asked a specific question about agent preferences:
<question>
Who is agent W3's, 5-most preferred agent?
</question>
Once you have determined the answer, provide your output in the following format:
1. The solution as a single agent name. For example, "W1"
Present your final answer within <answer> tags.
IMPORTANT: ONLY RETURN THE NAME OF THE SINGLE AGENT THAT IS THE ANSWER TO THE QUESTION. Do not include any explanations or additional information in your final answer.

- **Comprehension (Level-2):**

Now, you will be asked a specific question about agent preferences:
<question>
Would agent W3, prefer M4 and M3 over M1?
</question>
Once you have determined the answer, provide your output in the following format:
1. The solution as a YES or a NO. For example, "NO"
Present your final answer within <answer> tags.
IMPORTANT: ONLY RETURN YES OR NO THAT IS THE ANSWER TO THE QUESTION. Do not include any explanations or additional information in your final answer.

- **Comprehension (Level-3):**

Now, you will be asked a specific question about agent preferences:
<question>
If agent W5 is currently engaged to M3, would she accept proposals from M1 and M4?
</question>
Once you have determined the answer, provide your output in the following format:
1. The solution as a YES or a NO. For example, "NO"
Present your final answer within <answer> tags.
IMPORTANT: ONLY RETURN YES OR NO THAT IS THE ANSWER TO THE QUESTION. Do not include any explanations or additional information in your final answer.

The input $\mathbf{x}^{(i)}$ ends with the following text, to indicate the start of the chain-of-thought reasoning trace,

Response:
<think>

K.5 Reasoning-trace ($r^{(i)}$):

- **Generating:**

Response:
<think>
Okay, so I need to find a stable matching for 5 men and 5 women. For this, I can use the Gale-Shapley algorithm. Before I compute the solution for this instance, I will list the steps of the Gale-Shapley algorithm.
1. Initialize all men and women as unmatched.
2. Create a list to keep track of each man's next proposal (initially set to 0 for all men).
3. While there are unmatched men:
a. Select an unmatched man (M).
b. Find the next woman (W) on M's preference list that he hasn't proposed to yet.
c. If W is unmatched, match M and W.
d. If W is matched but prefers M to her current partner:
- Unmatch W from her current partner.
- Match M and W.
- Set the unmatched man as W's previous partner.
e. If W rejects M, move to the next woman on M's preference list.
4. Repeat step 3 until all men are matched.
Next, I will exectute the above algorithm for the given instance. The steps are as follows:
M1 is free. M1 proposes to W3
Since W3 is free, W3 accepts the proposal. Now M1 and W3 are matched.
M2 is free. M2 proposes to W5
Since W5 is free, W5 accepts the proposal. Now M2 and W5 are matched.
M3 is free. M3 proposes to W1
Since W1 is free, W1 accepts the proposal. Now M3 and W1 are matched.
M4 is free. M4 proposes to W4
Since W4 is free, W4 accepts the proposal. Now M4 and W4 are matched.
M5 is free. M5 proposes to W4
Since W4 prefers their current partner M4 to M5, W4 rejects the proposal. M4 and W4 are still matched, and M5 is still free.
M5 is free. M5 proposes to W1
Since W1 prefers M5 to their current partner M3, W1 accepts the proposal. Now M5 and W1 are matched, and M3 is free.
M3 is free. M3 proposes to W4
Since W4 prefers their current partner M4 to M3, W4 rejects the proposal. M4 and W4 are still matched, and M3 is still free.
M3 is free. M3 proposes to W3
Since W3 prefers their current partner M1 to M3, W3 rejects the proposal. M1 and W3 are still matched, and M3 is still free.
M3 is free. M3 proposes to W2
Since W2 is free, W2 accepts the proposal. Now M3 and W2 are matched.
</think>

- **Comprehension (Level-1):**

Alright, I have been given the task of determining W3's 5-most preferred agent. For this, let me look at her preference list in order:
position 1: M4
position 2: M5
position 3: M1
position 4: M3
position 5: M2
Hence, W3's 5-most preferred agent is M2.
</think>

- **Comprehension (Level-2):**

https://doi.org/10.52202/085713-0639

Alright, I have been asked to determine whether agent W3 would prefer M4 and M3 over M1. For this, let me look at W3's preference list in order:
position 1: M4
position 2: M5
position 3: M1
position 4: M3
position 5: M2
So, M4 is W3's 1-most preferred partner, M3 is W3's 4-most preferred partner, and M1 is W3's 3-most preferred partner.
Hence, agent W3 prefers M4 over M1, and agent W3 does not prefer M3 over M1. This means that the order of preference is M4 $>$ M1 $>$ M3.
Therefore, since the question was whether agent W3 would prefer BOTH M4 AND M3, given that it DOES NOT prefer M3 over M1, the answer is 'NO'.
</think>

- **Comprehension (Level-3):**

Alright, I have been asked to determine whether agent W5 would accept a proposal from M1 or M4, if she is currently matched with M3. For this, let me look at W5's preference list in order:
position 1: M5
position 2: M4
position 3: M2
position 4: M3
position 5: M1
So, M1 is W5's 5-most preferred partner, M4 is W5's 2-most preferred partner, and M3 is W5's 4-most preferred partner.
Hence, agent W5 does not prefer M1 over their current partner M3, and agent W5 prefers M4 over their current partner M3. This means that the order of preference is M4 $>$ M3 $>$ M1.
Therefore, since the question was whether agent W5 would prefer EITHER M1 OR M4, given that it prefers M4 over M3, the answer is 'YES'.
</think>

K.6 Answer ($a^{(i)}$):

- **Generating:**

<answer>
{
"M1": "W3",
...
"M5": "W1"
}
</answer>

- **Comprehension (Level-1):**

<answer>M2</answer>

- **Comprehension (Level-2):**

<answer>NO</answer>

- **Comprehension (Level-3):**

<answer>YES</answer>

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The claims made in the abstract are an accurate representation of the paper's contributions and scope; a more detailed summary of results and contributions is present in Section 1.1, and detailed discussion for each result is present in subsequent sections in the main paper.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: The limitations of the work presented in this paper is included in the supplemental material (present in the appendix)

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

https://doi.org/10.52202/085713-0639

Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [NA]

Justification: [NA]

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [Yes]

Justification: All code and parameters required to run the experiments are present in the main paper and the supplemental material (the supplemental code).

Guidelines:

- The answer NA means that the paper does not include experiments.
- If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: All code used to run the experiments described in the main paper and appendix is available as a part of the supplemental material

   Guidelines:

   - The answer NA means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

   Answer: [Yes]

   Justification: All experimental settings and details (including model parameters) are detailed in the supplemental material (in the appendix, as well as the supplemental code).

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: The statistical significance tests used throughout the paper have been mentioned on Page 5 (footnote 4).

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.

https://doi.org/10.52202/085713-0639

- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: All details about computational resources used are mentioned in the supplemental material (in the appendix)

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: The research conducted in this paper conforms in every respect with the NeurIPS Code of Ethics

   Guidelines:

   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: the broader positive and negative societal impacts of this work are discussed in the supplemental work (in the appendix).

    Guidelines:

- The answer NA means that there is no societal impact of the work performed.
- If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

    Answer: [NA]

    Justification: While pretrained language models are used in this paper, all datasets and models mentioned in this paper use randomly-generated and openly available data.

    Guidelines:

    - The answer NA means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: All original owners and creators of assets (code and models) have been properly credited and respected in both the main paper and supplemental materials.

    Guidelines:

    - The answer NA means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.

https://doi.org/10.52202/085713-0639

- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: All assets introduced in the paper are well documented and openly available through the supplemental materials.

    Guidelines:

    - The answer NA means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [NA]

    Justification: [NA]

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [NA]

    Justification: [NA]

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.

- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does not impact the core methodology, scientific rigorousness, or originality of the research, declaration is not required.

    Answer: [NA]

    Justification: This paper does not involve LLMs as any important, original, or non-standard components.

    Guidelines:

    - The answer NA means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy (`https://neurips.cc/Conferences/2025/LLM`) for what should or should not be described.

https://doi.org/10.52202/085713-0639