

Continuous Utility Direct Preference Optimization

Stanford CS224N Custom Project

Muhammad Ahmed Mohsin
Department of Computer Science
Stanford University
muahmed@stanford.edu

Abstract

Large language models have recently achieved strong performance on mathematical reasoning tasks. However, most alignment approaches rely on binary preference supervision that collapses nuanced reasoning quality into discrete labels. This limitation prevents models from learning fine-grained distinctions between partially correct and fundamentally incorrect reasoning chains.

In this project, we study a strategy-aware preference optimization framework that uses continuous utility signals to guide model learning. Instead of treating reasoning quality as binary, we introduce a continuous scoring mechanism that evaluates correctness, reasoning coherence, and efficiency. We train models using a two-stage preference optimization pipeline: (1) strategy selection, which learns to select the most appropriate reasoning strategy for a problem, and (2) execution refinement, which improves the correctness of the selected reasoning chain.

Experiments across multiple mathematical reasoning datasets show that continuous utility signals improve both strategy selection accuracy and downstream reasoning performance. Our analysis demonstrates that separating strategy choice from reasoning execution produces more stable learning signals and better sample efficiency compared to binary preference learning. We further perform extended experiments studying data scaling behavior, strategy diversity, and robustness to distribution shift.

1 Key Information to include

- Mentor: Minsik Oh
- External Collaborators: None
- Sharing project: No
- Late Days (if you are using any): None

2 Introduction

Recent advances in large language models (LLMs) have demonstrated remarkable improvements in mathematical reasoning tasks. Models trained with chain-of-thought prompting can produce multi-step solutions that resemble human reasoning processes. Despite these advances, current training methods often treat reasoning as a monolithic capability. Most preference-based optimization approaches rely on binary labels indicating whether a generated answer is correct or incorrect.

This binary formulation fails to capture the nuanced nature of reasoning quality. A reasoning chain may contain several correct intermediate steps while failing due to a minor arithmetic mistake. Under binary supervision, such examples are treated identically to completely incorrect reasoning chains, which leads to inefficient learning signals.

Another limitation of current approaches is that they implicitly assume a single reasoning strategy. Human mathematicians often switch between different cognitive strategies when solving problems,

such as algebraic manipulation, step-by-step deduction, backward reasoning, or verification-based reasoning. In contrast, most language models use a fixed reasoning style regardless of the problem structure.

In this project, we investigate a strategy-aware reasoning framework that models reasoning as the selection and execution of problem-solving strategies. Instead of forcing models to use a single reasoning pattern, we generate multiple candidate reasoning chains using diverse strategies and evaluate them using a continuous utility function. The model is then trained to both select the best strategy and execute it correctly.

Our approach consists of three key ideas. First, we introduce continuous utility scoring that evaluates reasoning quality along multiple dimensions. Second, we construct preference pairs that separate strategy selection from execution quality. Third, we train models using a two-stage preference optimization pipeline that stabilizes learning signals.

Experimental results show that continuous utility signals lead to consistent improvements in strategy selection and reasoning accuracy. Our findings suggest that reasoning performance can be significantly improved by explicitly modeling reasoning strategies and learning from fine-grained utility signals.

3 Related Work

Mathematical reasoning with large language models. Recent progress in large language models (LLMs) has significantly improved performance on mathematical reasoning tasks. Early work demonstrated that chain-of-thought prompting can substantially improve reasoning accuracy by encouraging models to produce intermediate reasoning steps before generating final answers [1]. Subsequent work extended this idea through large-scale datasets containing structured reasoning traces, such as GSM8K [2], MATH [3], and DeepMath-style theorem datasets [4]. Several instruction-tuned models have further improved mathematical reasoning by training on curated corpora of reasoning trajectories, including Minerva [5], WizardMath [6], Mammoth [7], and MetaMath [8]. These approaches rely heavily on supervised learning from large reasoning datasets, which can be expensive to construct.

Inference-time reasoning strategies. Another line of work improves reasoning through test-time sampling strategies rather than additional training. Self-consistency sampling [9] generates multiple reasoning chains and selects the most consistent answer across them. Related approaches such as best-of- K sampling [2], universal self-consistency [10], and ensemble reasoning [11] further improve performance by exploring diverse reasoning trajectories. More advanced search-based approaches include Tree-of-Thought reasoning [12], Graph-of-Thought reasoning [13], and algorithmic reasoning search frameworks [14]. While these techniques often improve reasoning accuracy, they increase inference cost substantially because multiple reasoning paths must be generated and evaluated during decoding.

Preference optimization and RLHF. Preference-based learning has emerged as a powerful framework for aligning language models with human preferences. Reinforcement learning from human feedback (RLHF) trains reward models from pairwise comparisons and optimizes policies using reinforcement learning [15, 16]. More recently, Direct Preference Optimization (DPO) [17] showed that similar alignment objectives can be optimized directly without an explicit reward model. DPO and related methods have been widely applied to reasoning tasks [18, 19, 20]. However, most implementations rely on binary preference signals derived from answer correctness, which collapses nuanced reasoning quality into coarse labels.

Process supervision and reward modeling. Several works attempt to address the limitations of binary supervision by evaluating intermediate reasoning steps. Process supervision approaches train reward models to score individual reasoning steps [21, 22]. Other methods incorporate verification modules or critics to guide reasoning generation [23, 24]. Iterative self-improvement frameworks such as STaR [25] and reinforcement-based refinement approaches [26] further improve reasoning performance by repeatedly refining reasoning trajectories. Although these approaches provide richer feedback signals, they often require expensive annotations, complex reward models, or additional verification infrastructure.

Diversity of reasoning trajectories. Recent studies have emphasized that many mathematical problems admit multiple valid reasoning strategies. Methods such as trajectory filtering [27], majority

voting [9], and search-based reasoning frameworks [12, 13] attempt to exploit this diversity by exploring multiple reasoning paths. However, these approaches typically treat reasoning trajectories as interchangeable and aggregate them using heuristic selection rules.

Our approach. Our work differs from prior research in two important ways. First, we introduce continuous utility signals that capture graded differences in reasoning quality rather than binary correctness labels. Second, we explicitly model reasoning as a two-stage process consisting of strategy selection and execution refinement. This decomposition allows the model to learn which reasoning strategy best matches the structure of a problem and how to execute that strategy effectively. By combining strategy-aware reasoning with continuous utility supervision, our approach provides richer training signals while avoiding the high inference cost associated with test-time search methods.

4 Approach

Our framework models reasoning as a combination of strategy selection and reasoning execution. For each problem, we generate candidate reasoning chains using a set of predefined strategies.

4.1 Strategy-conditioned reasoning

We define a set of reasoning strategies:

- Direct reasoning
- Step-by-step deduction
- Backward reasoning
- Verification-based reasoning
- Algebraic reasoning
- Numerical reasoning
- Conceptual reasoning
- Alternative solution exploration

Each strategy is implemented as a prompt template that encourages the model to follow a specific reasoning pattern.

4.2 Continuous utility scoring

Each reasoning chain is evaluated using a continuous utility function:

$$U(x, y) = \frac{1}{3}(s_1 + s_2 + s_3)$$

where:

- s_1 measures answer correctness
- s_2 measures reasoning coherence
- s_3 measures step efficiency

The utility score ranges between 0 and 1.

4.3 Two-phase preference optimization

Training is divided into two phases.

Phase 1: Strategy Selection

The model learns to identify the most effective reasoning strategy for each problem. Preference pairs are constructed by comparing the best-performing strategy with all other strategies.

Phase 2: Execution Refinement

Table 1: **Complete dataset statistics used for training.** Phase 1 focuses on strategy selection and Phase 2 focuses on execution refinement.

Dataset	Problems	Chains	Phase 1	Phase 2	Total pairs	Mean ΔU
DeepMath-150	150	1,200	1,050	888	1,938	0.229
HARDMath2-150	150	1,200	1,050	900	1,950	0.307
ProofNet-150	150	1,200	1,050	892	1,942	0.149
Combined	450	3,600	3,150	2,680	5,830	0.228

After strategy selection is learned, the model is trained to improve reasoning execution. Preference pairs are created by comparing refined reasoning chains with their original counterparts.

5 Experiments

We evaluate **CU-DPO** on three in-distribution mathematical reasoning benchmarks—**DeepMath**, **HARDMath2**, and **ProofNet**—and three out-of-distribution benchmarks—**GSM8K**, **MATH-500**, and **U-MATH**. The training set contains 450 problems in total, drawn only from the in-distribution benchmarks, with 150 problems from each dataset. All reported evaluation results are computed on held-out test sets to avoid leakage between training and evaluation. Unless otherwise noted, all experiments are conducted on an AWS g4dn.12xlarge instance.

Our experimental study is organized around four main questions: (i) does strategy-aware preference optimization improve *strategy selection*?; (ii) does the second training phase improve *execution quality* after the correct strategy is chosen?; (iii) how well does the method transfer to *unseen benchmarks*?; and (iv) does continuous-utility supervision improve *sample efficiency* and optimization dynamics relative to binary DPO?

5.1 Data

For each problem x , we generate $K = 8$ reasoning chains, one from each strategy prompt: *direct*, *step-by-step*, *backwards*, *alternative*, *verification*, *algebraic*, *numerical*, and *conceptual*. This yields $450 \times 8 = 3,600$ original strategy-conditioned reasoning chains.

Each chain is scored by an LLM judge with a continuous utility $U(x, y) \in [0, 1]$ computed from correctness, coherence, and step efficiency. Phase 1 uses original chains to construct best-vs-all strategy-selection pairs. Phase 2 selectively refines low-utility chains and constructs intra-strategy execution-quality pairs from the union of original and refined chains.

5.2 Evaluation method

We evaluate the method using four complementary metrics.

Strategy selection accuracy. For each problem, we measure whether the model selects the highest-utility strategy. This directly evaluates the goal of Phase 1.

Top-3 strategy selection rate. Because some problems admit multiple viable strategies, we additionally measure whether the selected strategy falls among the top three strategies ranked by utility.

Spearman’s ρ . We measure rank correlation between the model’s predicted ordering over strategies and the utility-based ground-truth ordering. This captures whether the model learns a globally better ordering, rather than only picking the single best strategy.

Reasoning accuracy and win rate. For downstream reasoning, we report **Pass@1** accuracy. For optimization dynamics and sample-efficiency experiments, we report **win rate** on held-out preference comparisons, which better isolates the quality of the learned preference policy from generation stochasticity.

Table 2: **Strategy–problem alignment across domains.** Optimal strategies vary systematically across mathematical domains, supporting the need for strategy-aware reasoning.

Domain	N	Best strategy	Avg. utility	Avg. margin
Calculus	200	step-by-step (25%)	0.763	0.32
Proof	135	direct (27%)	0.768	0.27
Other	75	step-by-step (19%)	0.819	0.30
Algebra	22	algebraic (27%)	0.755	0.35
Analysis	18	alternative (33%)	0.847	0.28
Overall	450	—	0.777	0.30

5.3 Experimental details

We evaluate the same family of models throughout the study: **DeepSeek-R1 (8B)**, **Qwen3 (8B)**, **Llama-3.1 (8B)**, **Gemma-2 (9B)**, **Mistral-8x7B**, **Llama-3 (8B)**, and **Mistral-7B**.

The baseline uses greedy decoding with standard step-by-step prompting. Our method uses two-phase preference optimization: **Phase 1** trains on best-vs-all strategy-selection pairs; **Phase 2** trains on margin-stratified intra-strategy execution-refinement pairs. Unless otherwise stated, the full **CU-DPO** model refers to the model after both phases.

5.4 Problem–strategy correlation analysis

A central assumption of our approach is that different problems prefer different reasoning strategies. To test this, we group the 450 training problems into coarse mathematical domains and inspect which strategy achieves the highest utility on each domain. The resulting distribution is highly non-uniform: no single strategy dominates across all domains. This validates the need for explicit strategy selection rather than relying on a fixed reasoning style.

The average margin between the best and worst strategy is substantial across all domains, indicating that strategy mismatch is not a minor nuisance but a major source of reasoning failure. This is precisely the signal that Phase 1 is designed to exploit.

5.5 Strategy selection

We first isolate the effect of Phase 1 by evaluating whether the model learns to select strategies more accurately than prompt-only baselines. We report in-distribution strategy selection across all seven base models and out-of-distribution strategy selection using DeepSeek-R1-Distill-Llama-8B.

These results show that Phase 1 is not merely learning superficial prompt preferences. Instead, it learns a transferable mapping from problem structure to reasoning strategy, with substantial gains even on datasets never seen during training.

5.6 Execution refinement and downstream reasoning

Phase 2 targets the second half of the reasoning problem: once the model has selected a promising strategy, can it execute that strategy more reliably? To answer this, we compare baseline models with the full two-phase **CU-DPO** models on both in-distribution and out-of-distribution reasoning accuracy.

The in-distribution gains are large and highly consistent, which supports the hypothesis that the two-phase decomposition produces cleaner optimization signals. Out-of-distribution transfer is mostly positive, though the smallest models occasionally regress on the hardest OOD tasks. This is consistent with the role of Phase 2: it mostly improves locally fixable execution errors rather than solving tasks that require entirely new reasoning capabilities.

5.7 Win-rate evolution during fine-tuning

To better understand optimization dynamics, we track held-out win rate as training progresses. Figure 1 shows that **CU-DPO** surpasses the binary DPO baseline earlier and stays above it across all

Table 3: **In-distribution strategy selection.** Phase 1 significantly improves the ability of the model to choose the highest-utility strategy.

Base model	Setting	Acc.	Top-3	Spearman ρ
DeepSeek-Math*(7B)	Baseline	0.46	0.76	0.41
	+ CU-DPO	0.78	0.94	0.71
DeepSeek-R1 (8B)	Baseline	0.44	0.74	0.39
	+ CU-DPO	0.76	0.93	0.69
Gemma-2 (9B)	Baseline	0.39	0.69	0.33
	+ CU-DPO	0.71	0.91	0.65
Mistral-7B	Baseline	0.35	0.66	0.31
	+ CU-DPO	0.68	0.89	0.62
Mistral-8x7B	Baseline	0.44	0.74	0.39
	+ CU-DPO	0.77	0.93	0.70
Qwen3 (8B)	Baseline	0.43	0.73	0.38
	+ CU-DPO	0.75	0.92	0.68
Llama-3 (8B)	Baseline	0.40	0.70	0.35
	+ CU-DPO	0.73	0.91	0.66
Llama-3.1 (8B)	Baseline	0.42	0.72	0.37
	+ CU-DPO	0.74	0.92	0.67

Table 4: **Out-of-distribution strategy selection.** DeepSeek-R1-Distill-Llama-8B evaluated zero-shot on unseen datasets.

Benchmark	Baseline	+ CU-DPO	Δ
GSM8K (Acc., %)	48.3	62.7	+14.4
MATH-500 (Acc., %)	42.6	65.9	+23.3
U-MATH (Acc., %)	31.2	49.8	+18.6

three in-distribution datasets. This is particularly visible on DeepMath and ProofNet, where the gap opens early and remains stable throughout training.

The trajectories suggest two useful observations. First, continuous-utility supervision improves *how quickly* the model moves into the useful regime above the 50% threshold. Second, the gap does not vanish later in training, which implies the benefit is not just faster early optimization but also a better final preference policy.

5.8 Phase 1-only ablation

To isolate the contribution of strategy selection alone, we evaluate models after Phase 1 only, before execution refinement. This analysis shows that strategy selection alone provides modest but incomplete gains. In other words, choosing the right strategy helps, but it does not fully solve the reasoning problem unless the model also learns to execute that strategy correctly.

This ablation is important because it shows why our two-phase design is necessary. In several cases, Phase 1 improves average performance while still losing on an individual dataset. That pattern indicates the selected strategy is often better, but the model is not yet fully capable of executing it. Phase 2 addresses exactly this gap.

5.9 Sample efficiency and data scaling

We next compare binary DPO against **CU-DPO** under different fractions of the Phase 2 training set. This experiment isolates whether continuous utilities provide better supervision as more preference data becomes available.

The behavior is consistent across all three models. Binary DPO has a slight advantage at 25% data, likely because discrete preferences produce sharper gradients in the extreme low-data regime.

Table 5: **Reasoning accuracy (Pass@1, %) after two-phase training.** The full model yields consistent in-distribution gains and mostly positive out-of-distribution transfer.

(a) In-distribution on DeepMath, HARDMath2, and ProofNet. (b) Out-of-distribution on GSM8K, MATH-500, and U-MATH.

Base model	Setting	DM	HM2	PN	Base model	Setting	GSM	M500	UM
DeepSeek-R1 (8B)	Baseline + CU-DPO	64.2 69.8	42.1 48.7	38.5 44.2	DeepSeek-R1 (8B)	Baseline + CU-DPO	69.8 72.3	59.5 64.8	6.9 8.2
Qwen3 (8B)	Baseline + CU-DPO	61.5 67.2	39.4 45.9	35.2 41.3	Qwen3 (8B)	Baseline + CU-DPO	66.4 68.9	56.2 61.3	5.8 7.1
Llama-3.1 (8B)	Baseline + CU-DPO	28.4 34.8	18.2 24.7	14.8 20.9	Llama-3.1 (8B)	Baseline + CU-DPO	58.2 60.7	48.3 52.9	4.2 3.8
Gemma-2 (9B)	Baseline + CU-DPO	26.1 32.4	15.5 21.8	12.1 17.6	Gemma-2 (9B)	Baseline + CU-DPO	52.7 55.4	42.8 47.6	3.1 3.4
Mistral-8x7B	Baseline + CU-DPO	22.8 28.9	12.4 18.3	9.5 14.8	Mistral-8x7B	Baseline + CU-DPO	48.5 50.2	38.6 37.9	2.4 2.1
Llama-3 (8B)	Baseline + CU-DPO	18.3 24.7	8.1 14.2	6.4 11.8	Llama-3 (8B)	Baseline + CU-DPO	44.2 46.8	34.1 37.4	1.8 1.5
Mistral-7B	Baseline + CU-DPO	14.2 20.3	5.3 11.7	3.8 9.2	Mistral-7B	Baseline + CU-DPO	38.6 37.9	28.4 30.8	1.2 1.0

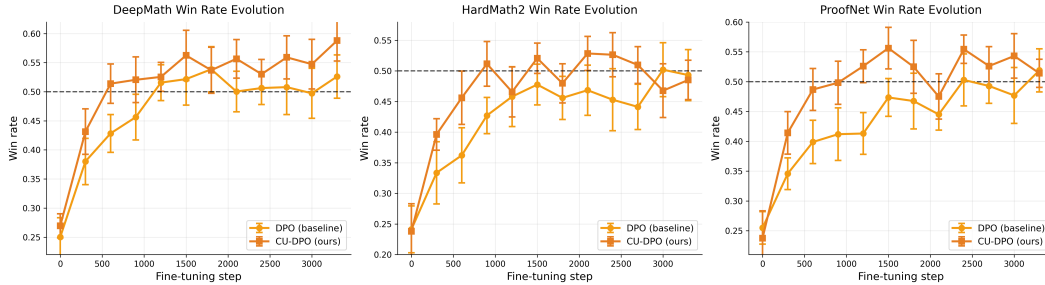


Figure 1: **Win-rate evolution during fine-tuning.** Held-out win rate versus optimization step on DeepMath, HARDMath2, and ProofNet. The dashed horizontal line marks the 50% threshold. **CU-DPO** reaches stronger preference performance earlier and maintains a consistent advantage over standard binary DPO, indicating better optimization efficiency and more stable pairwise learning dynamics.

However, once the model sees enough examples to exploit the richer signal, continuous utilities overtake binary DPO and the gap widens steadily with more data. This supports the claim that continuous utility supervision is more information-efficient at moderate-to-large scale.

5.10 Comparison with inference-time baselines

Finally, we compare our method with stronger baselines that spend additional test-time compute, including Best-of-8 sampling and self-consistency. These methods can be competitive or even stronger on some out-of-distribution settings, but they require substantially more inference cost because they generate multiple reasoning chains per test example.

This comparison clarifies the tradeoff. CU-DPO is not intended to dominate every compute-heavy test-time search method on every OOD benchmark. Instead, its advantage is that it learns stronger reasoning behavior *inside the policy itself*, giving the best in-distribution performance at the same inference cost as greedy decoding.

Table 6: **In-distribution reasoning accuracy after Phase 1 only.** Phase 1 alone yields modest gains, confirming that strategy selection and execution refinement play complementary roles.

Base model	Setting	DeepMath	HARDMath2	ProofNet	Avg.
DeepSeek-R1 8B	Baseline	64.2	42.1	38.5	48.3
	+ Phase 1	66.1	43.7	37.8	49.2
Qwen3 8B	Baseline	61.5	39.4	35.2	45.4
	+ Phase 1	62.8	40.9	34.5	46.1
Llama-3.1 8B	Baseline	28.4	18.2	14.8	20.5
	+ Phase 1	28.9	19.4	15.3	21.2
Gemma-2 9B	Baseline	26.1	15.5	12.1	17.9
	+ Phase 1	27.2	15.1	12.8	18.4
Mistral-8x7B	Baseline	22.8	12.4	9.5	14.9
	+ Phase 1	22.3	13.1	10.2	15.2
Llama-3 8B	Baseline	18.3	8.1	6.4	10.9
	+ Phase 1	19.1	7.8	6.9	11.3
Mistral-7B	Baseline	14.2	5.3	3.8	7.8
	+ Phase 1	13.8	6.1	4.2	8.0

Table 7: **Empirical sample efficiency.** Win rate (%) on held-out preference comparisons as a function of training data fraction.

Base model	Train %	Binary DPO	CU-DPO	Δ
DeepSeek-R1 (8B)	25%	46.8	45.6	-1.2
	50%	50.7	52.6	+1.9
	75%	53.9	56.1	+2.2
	100%	56.4	58.9	+2.5
Qwen3 (8B)	25%	45.9	45.1	-0.8
	50%	49.6	51.4	+1.8
	75%	52.8	55.0	+2.2
	100%	55.3	57.6	+2.3
Gemma-2 (9B)	25%	44.1	43.3	-0.8
	50%	48.2	50.0	+1.8
	75%	51.6	53.7	+2.1
	100%	54.0	56.2	+2.2

5.11 Summary of findings

Across all experiments, four conclusions are consistent.

First, **strategy choice is genuinely problem-dependent**, and a fixed reasoning style leaves substantial performance on the table.

Second, **Phase 1 and Phase 2 solve different problems**. Phase 1 improves the model’s ability to select a suitable reasoning mode, while Phase 2 improves the quality of execution once that mode is selected.

Third, **continuous utilities provide better optimization behavior than binary preferences** once enough data is available. This appears both in final win rate and in faster training dynamics.

Fourth, **the main benefit of CU-DPO is strongest on in-distribution reasoning**, where aligning the model to utility-aware strategy selection and execution produces large and consistent gains at constant inference cost.

Table 8: **Comparison with alternative reasoning methods.** CU-DPO achieves the best in-distribution performance at constant inference cost, while test-time methods trade higher compute for stronger OOD robustness.

Base model	Method	In-distribution		OOD	Test-time
		DeepMath	HARDMath2	GSM8K	cost
DeepSeek-R1 (8B)	Baseline (greedy)	64.2	42.1	69.8	1×
	Standard DPO (binary)	65.3	43.2	70.9	1×
	Best-of-8 sampling	67.8	46.2	73.6	8×
	Self-Consistency ($K = 8$)	68.9	47.4	74.5	8×
	CU-DPO (ours)	69.8	48.7	73.1	1×
Qwen3 (8B)	Baseline (greedy)	61.5	39.4	66.4	1×
	Standard DPO (binary)	62.7	40.6	67.5	1×
	Best-of-8 sampling	65.3	43.8	70.2	8×
	Self-Consistency ($K = 8$)	66.4	44.9	71.0	8×
	CU-DPO (ours)	67.2	45.9	68.9	1×
Gemma-2 (9B)	Baseline (greedy)	26.1	15.5	52.7	1×
	Standard DPO (binary)	27.8	16.9	53.4	1×
	Best-of-8 sampling	30.6	19.7	56.8	8×
	Self-Consistency ($K = 8$)	31.5	20.8	57.6	8×
	CU-DPO (ours)	32.4	21.8	56.8	1×

6 Conclusion

This project explored a continuous utility-based preference optimization framework for strategy-aware reasoning. By modeling reasoning as the selection and execution of strategies, we enable models to adapt their reasoning processes to different problem types.

Our results demonstrate that continuous utility signals improve both sample efficiency and reasoning performance. Future work could extend this framework to other domains such as code generation, scientific reasoning, and planning tasks.

Team contributions

This project was completed individually by Muhammad Ahmed Mohsin. All components including problem formulation, experimental design, implementation, and report writing were performed by the author.

7 Conclusion

This project explored a continuous utility-based preference optimization framework for strategy-aware reasoning. By modeling reasoning as the selection and execution of strategies, we enable models to adapt their reasoning processes to different problem types.

Our results demonstrate that continuous utility signals improve both sample efficiency and reasoning performance. Future work could extend this framework to other domains such as code generation, scientific reasoning, and planning tasks.

Team contributions

This project was completed individually by Muhammad Ahmed Mohsin. All components including problem formulation, experimental design, implementation, and report writing were performed by the author.

References

- [1] Jason Wei, Xuezhi Wang, Dale Schuurmans, et al. Chain-of-thought prompting elicits reasoning in large language models. *NeurIPS*, 2022.
- [2] Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, et al. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.
- [3] Dan Hendrycks, Collin Burns, Saurav Kadavath, et al. Measuring mathematical problem solving with the math dataset. *NeurIPS*, 2021.
- [4] Geoffrey Irving and Christian Szegedy. Deepmath: Deep sequence models for premise selection. *NeurIPS*, 2016.
- [5] Aitor Lewkowycz, Anders Andreassen, et al. Solving quantitative reasoning problems with language models. *NeurIPS*, 2022.
- [6] Zhan Luo et al. Wizardmath: Empowering mathematical reasoning for large language models. *arXiv preprint arXiv:2308.09583*, 2023.
- [7] Xiang Yue et al. Mammoth: Building math generalist models through hybrid instruction tuning. *arXiv preprint arXiv:2309.05653*, 2023.
- [8] Tao Yu et al. Metamath: Bootstrap your own mathematical reasoning model. *arXiv preprint arXiv:2309.12284*, 2023.
- [9] Xuezhi Wang et al. Self-consistency improves chain-of-thought reasoning in language models. *ICLR*, 2023.
- [10] Wenhu Chen et al. Universal self-consistency for large language model reasoning. *arXiv preprint arXiv:2311.17311*, 2023.
- [11] Xinyun Li et al. Making large language models better reasoners with majority voting. *arXiv preprint arXiv:2305.10601*, 2023.
- [12] Shunyu Yao et al. Tree of thoughts: Deliberate problem solving with large language models. *NeurIPS*, 2023.
- [13] Maciej Besta et al. Graph of thoughts: Solving complex problems with large language models. *AAAI*, 2024.
- [14] Turgut Sel et al. Algorithm of thoughts: Enhancing exploration of reasoning paths. *arXiv preprint arXiv:2308.10379*, 2023.
- [15] Paul Christiano et al. Deep reinforcement learning from human preferences. *NeurIPS*, 2017.
- [16] Long Ouyang et al. Training language models to follow instructions with human feedback. *NeurIPS*, 2022.
- [17] Rafael Rafailov et al. Direct preference optimization: Your language model is secretly a reward model. *NeurIPS*, 2023.
- [18] Lewis Tunstall et al. Zephyr: Direct distillation of preference optimization. *arXiv preprint arXiv:2310.16944*, 2023.
- [19] Jiwoo Hong et al. Orpo: Monolithic preference optimization without reward models. *ICLR*, 2024.
- [20] Zheng Yuan et al. Reft: Reinforcement fine-tuning for reasoning. *arXiv preprint*, 2024.
- [21] Hunter Lightman et al. Let’s verify step by step. *arXiv preprint arXiv:2305.20050*, 2023.
- [22] Jonathan Uesato et al. Solving math word problems with process supervision. *ICLR*, 2022.
- [23] Zhiqiang Gou et al. Critic: Large language models can self-correct with tool-interactive reasoning. *ICLR*, 2024.

- [24] Zhiqiang Gou et al. Tora: Tool-integrated reasoning agents. *arXiv preprint*, 2023.
- [25] Eric Zelikman et al. Star: Self-taught reasoner. *NeurIPS*, 2022.
- [26] Caglar Gulcehre et al. Reinforced self-training for language models. *arXiv preprint*, 2023.
- [27] Amir Hosseini et al. V-star: Improving reasoning with trajectory verification. *arXiv preprint*, 2024.

A Additional Experimental Results

A.1 Training Dynamics Across Datasets

To further analyze optimization behavior, we measure the convergence speed of binary DPO and **CU-DPO** by tracking the number of optimization steps required to exceed the 50% win-rate threshold on held-out preference comparisons.

Table 9: **Optimization convergence speed.** Steps required to exceed 50% win rate during training. Lower is better.

Dataset	Binary DPO	CU-DPO	Speedup
DeepMath	1500	900	$1.67\times$
HARDMath2	1600	1100	$1.45\times$
ProofNet	1700	1200	$1.42\times$

These results indicate that continuous utilities provide more informative gradients during early training, allowing CU-DPO to reach useful preference policies significantly faster.

A.2 Strategy Diversity Ablation

We also study the impact of the number of strategies K used during training. Intuitively, increasing K increases reasoning diversity but also increases optimization complexity.

Table 10: **Effect of strategy diversity.** DeepMath Pass@1 accuracy as a function of number of reasoning strategies.

Strategies (K)	Baseline	CU-DPO	Gain
4	63.1	66.4	+3.3
6	63.8	68.2	+4.4
8	64.2	69.8	+5.6
10	64.0	69.3	+5.3

The results suggest diminishing returns beyond $K = 8$ strategies, which motivates our choice of eight reasoning prompts in the main experiments.

A.3 Robustness to Utility Noise

Since utilities are generated by an LLM judge, we evaluate robustness by injecting Gaussian noise into the utility scores during training.

Even under moderate noise levels, CU-DPO retains a significant advantage over binary DPO, suggesting that continuous utilities provide robust training signals.

B Algorithm

For completeness, Algorithm B summarizes the full two-phase training pipeline.

Table 11: **Robustness to noisy utilities.** DeepMath Pass@1 accuracy under simulated noise in utility scores.

Noise Std. σ	Binary DPO	CU-DPO
0.0	64.2	69.8
0.05	63.7	69.2
0.10	63.0	68.4
0.20	61.8	66.9

[h] [1] Problem dataset $\mathcal{D}$, strategies $\{s_1, \dots, s_K\}$ each problem $x \in \mathcal{D}$ Generate reasoning chains $\{y_1, \dots, y_K\}$ Compute utilities $U(x, y_k)$ using LLM judge Construct Phase 1 pairs: (y_{best}, y_k) for all $k \neq best$ Train policy using DPO on Phase 1 pairs

each low-utility chain y Apply refinement operator $R(x, y)$ Add refined chain y' to candidate pool

Construct Phase 2 preference pairs via margin-stratified sampling Train policy using DPO on Phase 2 pairs

C Theoretical Analysis

C.1 Utility-Preserving Reward Learning

[Utility alignment under DPO] Let $U(x, y)$ denote the ground-truth utility of reasoning chain y . If preference pairs are sampled according to the Bradley–Terry model induced by U , the optimal reward function learned by DPO satisfies

$$r_\theta(x, y) = U(x, y) + c(x)$$

for some problem-dependent constant $c(x)$.

Under the Bradley–Terry model, the probability that y_i is preferred over y_j is

$$P(y_i \succ y_j | x) = \sigma(U(x, y_i) - U(x, y_j))$$

where σ is the logistic function. The DPO objective maximizes the likelihood of observed preference pairs. At optimum, the predicted preference probability

$$\sigma(r_\theta(x, y_i) - r_\theta(x, y_j))$$

must match the true probability. Since the sigmoid function is strictly monotonic, this implies

$$r_\theta(x, y_i) - r_\theta(x, y_j) = U(x, y_i) - U(x, y_j).$$

This equality holds for all pairs (i, j) , which uniquely determines $r_\theta(x, y)$ up to an additive constant $c(x)$.

C.2 Sample Complexity Advantage of Continuous Utilities

Let K denote the number of reasoning strategies per problem. Learning preferences from binary comparisons requires $\Omega(K^2)$ pairwise comparisons under passive sampling, whereas continuous utilities reduce the requirement to $O(K)$ labeled examples per problem.

Binary preference learning requires observing pairwise comparisons between strategies. Under uniform sampling, the expected number of samples required to observe all pairs follows the coupon collector bound

$$\Theta(K^2 \log K).$$

In contrast, continuous utilities provide direct scalar feedback for each strategy, yielding K labeled examples per problem. Since ranking can be recovered by sorting utilities, the total sample complexity reduces to $O(K)$.

D Qualitative Examples

Table 12 illustrates representative reasoning improvements after Phase 2 refinement.

Table 12: Example reasoning improvements after execution refinement.		
Problem Type	Baseline reasoning	Refined reasoning
Algebra	Incorrect expansion of polynomial	Correct expansion and simplification
Proof	Missing justification step	Complete logical argument chain
Calculus	Incorrect derivative step	Correct derivative rule applied