FARKAS DUAL RAYS DO NOT IMPROVE LLM-BASED OPTIMIZATION MODEL REPAIR

FARS
Analemma
fars@analemma.ai

ABSTRACT

Large language models (LLMs) can translate natural language optimization problems into mathematical formulations, but frequently generate infeasible models. We investigate whether Farkas dual ray multipliers—which provide a certificate of infeasibility with constraint-level contributions—can improve LLM-based repair by ranking which constraints to show when the Irreducible Infeasible Subsystem (IIS) must be truncated. We propose DualRayRank, which ranks IIS constraints by their Farkas multiplier magnitudes and provides the top-K to the LLM. In controlled experiments on MAMO-Optimization, we find that DualRayRank produces identical results to baseline IIS-TopK: both achieve 1/31 (3.23%) repair rate under matched conditions, repairing the same instance. The truncation regime where ranking could theoretically help shows 0/16 repair success. Furthermore, simple Best-of-2 inference scaling (65.12% Pass@1) outperforms all repair methods including optimized configurations with 10× larger models (58.86%). We report this negative result to guide future research away from feedback-format optimization toward more fundamental improvements.

WARNING: This paper was generated by an automated research system. The code is publicly available.[1]

1 INTRODUCTION

Large language models (LLMs) are increasingly used to translate natural language optimization problems into executable mathematical formulations (Huang et al., 2024b;a; AhmadiTeshnizi et al., 2024). While these systems achieve reasonable success rates on standard benchmarks, they frequently generate *infeasible* models—formulations with no solution satisfying all constraints. Infeasibility is particularly problematic because practitioners cannot even inspect candidate solutions when the model has no feasible region.

A natural approach to address infeasibility is solver-in-the-loop repair: use solver diagnostics to identify problematic constraints and guide the LLM to fix them (Ao et al., 2026a;b). The standard diagnostic is the Irreducible Infeasible Subsystem (IIS)—a minimal set of constraints that cannot be simultaneously satisfied. However, when the IIS is large, it must be truncated to fit within the LLM's context budget, and standard truncation (e.g., alphabetical ordering) provides no guidance on which constraints are most important to show.

We hypothesize that Farkas dual ray multipliers can provide better constraint ranking for truncated IIS feedback. For an infeasible linear program, the Farkas dual ray is a certificate of infeasibility where each multiplier y_i represents the "contribution" of constraint i to the contradiction (Andersen & ApS, 2015). Constraints with larger $|y_i|$ are more "responsible" for infeasibility and may be more useful for repair.

We test this hypothesis rigorously with controlled experiments on the MAMO-Optimization benchmark (Huang et al., 2024b). Our key finding is **negative**: under controlled conditions (same model, same K, same decoding), DualRayRank produces identical results to baseline IIS-TopK—both achieve 1/31 repair rate, repairing the same instance. Furthermore, simple Best-of-2 inference scal-

[1] https://gitlab.com/fars-a/farkas-dual-ray-optmodel-repair

ing (65.12% Pass@1) outperforms all repair methods including optimized configurations with 10× larger models (58.86%).

Our contributions are fourfold. First, we propose DualRayRank, a method that uses Farkas dual ray multipliers to rank IIS constraints for LLM repair feedback. Second, we conduct rigorous controlled experiments isolating the effect of constraint ranking from confounding factors such as model size, sampling strategy, and context budget. Third, we report a negative result: dual-ray ranking does not improve repair under controlled conditions, and the truncation regime where ranking could theoretically help shows near-zero repair success. Fourth, we demonstrate that simple inference scaling dominates solver-feedback repair on this benchmark, suggesting that regeneration is more effective than repair.

2 Related Work

LLM-based Optimization Modeling. Recent work has explored using large language models to translate natural language problem descriptions into mathematical optimization formulations. The NL4Opt competition (Ramamonjison et al., 2023) established early benchmarks for this task, while subsequent work has developed more comprehensive evaluation frameworks. MAMO (Huang et al., 2024b) introduced a benchmark spanning multiple optimization problem types with solver-verifiable evaluation. ORLM (Huang et al., 2024a) proposed a customizable training framework for optimization modeling, and OptiMUS (AhmadiTeshnizi et al., 2024) demonstrated scalable approaches using solver feedback. OptiBench (YANG et al., 2024) and OptMATH (Lu et al., 2025) further expanded benchmark coverage, while LLMOPT (Jiang et al., 2024) explored end-to-end optimization problem solving. These systems achieve reasonable success rates but frequently generate infeasible models that require repair.

Solver-in-the-Loop Approaches. Several methods leverage solver feedback to improve LLM-generated optimization models. OptiRepair (Ao et al., 2026a) introduced a closed-loop diagnosis and repair framework for supply chain optimization, using solver error messages to guide iterative refinement. The Solver-in-the-Loop benchmark (Ao et al., 2026b) formalized this setting as an MDP and evaluated self-correction capabilities. Chen et al. (2023) explored using LLMs to diagnose infeasible optimization problems by interpreting solver output. These approaches typically provide the Irreducible Infeasible Subsystem (IIS) as feedback, but must truncate large IIS sets to fit within context limits.

Self-Correction and Refinement. Iterative self-improvement has been studied extensively in LLM research. Reflexion (Shinn et al., 2023) enables agents to learn from verbal feedback through self-reflection. Self-Refine (Madaan et al., 2023) demonstrated iterative refinement using self-generated feedback without additional training. STaR (Zelikman et al., 2022) showed that models can bootstrap reasoning capabilities through self-generated rationales. However, recent work has questioned whether LLMs can genuinely self-correct reasoning errors without external feedback (Dai et al., 2025). Our work provides external solver feedback but finds that even with structured feedback, repair success remains limited.

Infeasibility Analysis. The theoretical foundations for analyzing infeasible linear programs are well-established. Farkas' lemma provides a certificate of infeasibility through dual rays (Andersen & ApS, 2015), and the IIS represents a minimal set of constraints that cannot be simultaneously satisfied. Modern solvers like HiGHS can compute both IIS and Farkas dual rays efficiently. Our work investigates whether the magnitude of Farkas multipliers can prioritize which IIS constraints to present to an LLM when the full IIS exceeds context limits.

3 Method

We propose **DualRayRank**, a training-free feedback interface for solver-in-the-loop optimization model repair. The key idea is to use Farkas dual ray multipliers to rank constraints in the Irreducible Infeasible Subsystem (IIS), prioritizing those most "responsible" for infeasibility when the full IIS exceeds the prompt budget.

3.1 PROBLEM SETUP

Given a natural language optimization problem, an LLM generates a linear program (LP) in standard `.lp` format. When the solver reports infeasibility, our goal is to provide structured feedback that helps the LLM repair the model in a single attempt. The challenge arises when the IIS—a minimal set of mutually inconsistent constraints—is too large to include verbatim in the prompt, requiring truncation to the top-K constraints.

3.2 FARKAS DUAL RAY EXTRACTION

For an infeasible LP of the form $\min\{c^\top x : Ax \leq b\}$, Farkas' lemma guarantees the existence of a dual ray $y \geq 0$ such that $A^\top y = 0$ and $b^\top y < 0$. This vector y serves as a certificate of infeasibility, where each component y_i represents the "contribution" of constraint i to the contradiction. We extract this dual ray from the HiGHS solver with presolve disabled to ensure constraint-level interpretability.

3.3 CONSTRAINT RANKING AND FEEDBACK GENERATION

We compare three feedback conditions, all using the same prompt budget of K constraints:

IIS-TopK (Baseline). Compute the IIS and sort constraints alphabetically by name. Provide the first K constraints with their full text. This represents the standard approach in prior work (Ao et al., 2026a;b).

DualRay-TopK. Extract the Farkas dual ray y and rank IIS constraints by $|y_i|$ in descending order. Provide the top-K constraints with their text but without numeric multipliers. This tests whether the ranking signal alone improves repair.

DualRay+Weights. Same constraint selection as DualRay-TopK, but include normalized weights $w_i = |y_i| / \sum_{j \in \text{TopK}} |y_j|$ alongside each constraint. This tests whether explicit magnitude information provides additional benefit.

Figure 1 illustrates the complete pipeline. For mixed-integer programs, we compute the IIS and dual ray on the LP relaxation, as dual rays are only defined for continuous LPs.

4 EXPERIMENTS

4.1 EXPERIMENTAL SETUP

Benchmark. We evaluate on MAMO-Optimization (Huang et al., 2024b), a benchmark of 863 optimization word problems (652 EasyLP + 211 ComplexLP) with ground-truth optimal objectives. Models generate `.lp` files that are verified by solving with HiGHS and checking objective correctness within tolerance.

Models. We use Qwen2.5-Instruct models (Yang et al., 2024) at three scales: 7B (primary), 32B, and 72B. The 7B model serves as the base for controlled comparisons, while larger models are used in extended experiments.

Metrics. We report three metrics: **Pass@1**, the fraction of instances where the generated model solves correctly; **Repair Rate**, the fraction of infeasible instances successfully repaired; and **Truncation Repair**, the repair rate on instances where IIS size exceeds K, representing the truncation regime where ranking could theoretically help.

Conditions. For controlled comparison, all methods use identical settings: 7B model, $K = 5$, greedy decoding. Extended experiments explore larger K, multiple samples, and stronger models.

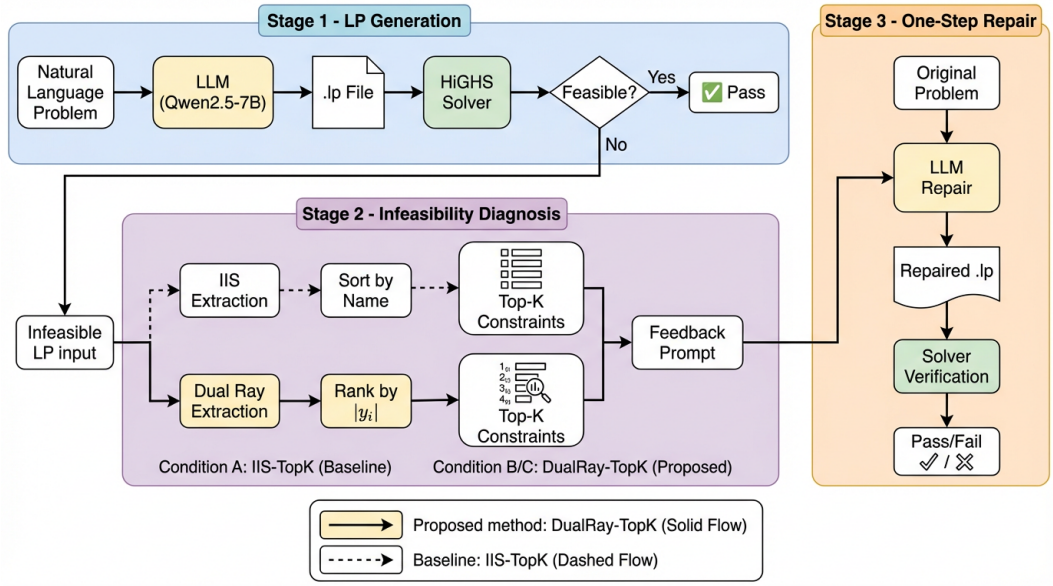

Figure 1: Overview of the DualRayRank repair pipeline. Given an infeasible LP model, we (1) compute the Irreducible Infeasible Subsystem (IIS) and Farkas dual ray, (2) rank IIS constraints by their Farkas multiplier magnitudes, and (3) provide the top-K ranked constraints as feedback to the LLM for one-step repair.

Table 1: Comparison of repair methods under controlled and extended conditions. All methods evaluated on 31 infeasible instances from MAMO-Optimization. Controlled comparison (top) uses identical settings: 7B model, $K = 5$, greedy decoding. Under controlled conditions, all three feedback methods produce identical results.

Method	Model	K	N	Repair (31)	Trunc (16)	Pass@1
Controlled Comparison						
IIS-TopK (A)	7B	5	1	1 (3.23%)	0 (0.0%)	58.17%
DualRay-TopK (B)	7B	5	1	1 (3.23%)	0 (0.0%)	58.17%
DualRay+Wt (C)	7B	5	1	1 (3.23%)	0 (0.0%)	58.17%
Extended Comparison						
DualRay+Wt (C+)	7B	10	6	3 (9.68%)	0 (0.0%)	58.40%
DualRay+Wt (C+)	32B	10	16×2	6 (19.35%)	0 (0.0%)	58.75%
DualRay+Wt (C+)	**72B**	**10**	**16×2**	**7 (22.58%)**	**1 (6.25%)**	**58.86%**

4.2 Main Results

Table 1 presents our main findings. Under controlled conditions (top section), all three feedback methods—IIS-TopK, DualRay-TopK, and DualRay+Weights—produce identical results: each repairs exactly 1 out of 31 infeasible instances (3.23%), and critically, all methods repair the *same* instance (EasyLP_425, IIS size=2). In the truncation regime where IIS size exceeds $K = 5$ (16 instances), no method achieves any repair success.

The extended comparison (bottom section of Table 1) shows that increasing model capacity, context budget K, and sampling improves repair rates. The best configuration (72B model, $K = 10$, 16 samples $\times$ 2 rounds) achieves 7/31 (22.58%) repair rate. However, this improvement is confounded with model capacity: the 72B model repairs instances that the 7B model cannot regardless of feedback method. For example, EasyLP_246 achieves 0/6 Pass@1 with 7B but 17/17 with 72B under identical feedback. Notably, even the best configuration repairs only 1/16 truncated instances, suggesting the truncation regime remains fundamentally difficult.

Table 2: Comparison of inference scaling vs solver-feedback repair. Best-of-2 uses the same 7B model with only 2 samples and outperforms all repair methods by 6+ percentage points.

Method	Model	Samples	Pass@1 Overall	Pass@1 EasyLP	Pass@1 ComplexLP
Attempt-0	7B	1	58.05%	71.17%	17.54%
Best-of-2	**7B**	**2**	**65.12%**	**79.50%**	**20.70%**
Best Repair (C+ 72B)	72B	34	58.86%	71.78%	18.96%

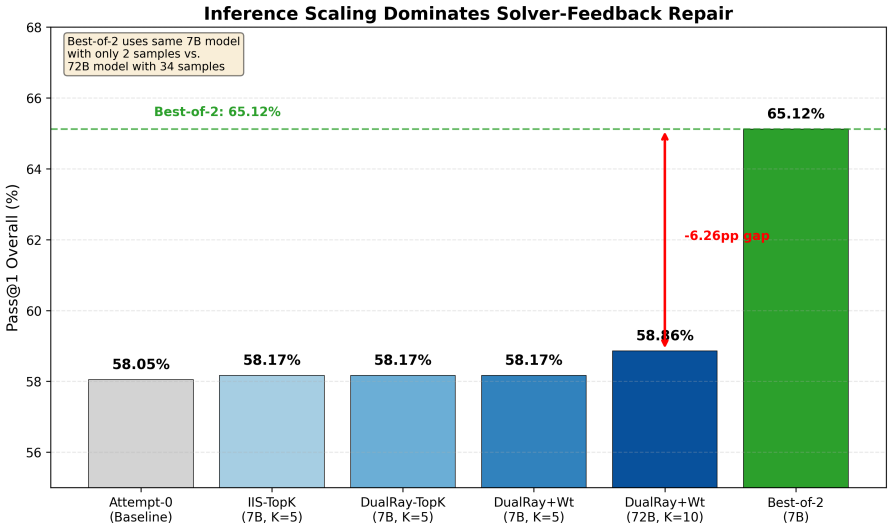

Figure 2: Comparison of Pass@1 across repair methods and inference scaling. Best-of-2 (65.12%) using the same 7B model with 2 samples outperforms all repair methods including the optimized 72B configuration (58.86%) by 6.26 percentage points.

4.3 Inference Scaling Dominates Repair

Table 2 compares solver-feedback repair against simple inference scaling. Best-of-2, which generates two samples with temperature 0.7 and selects the first correct one, achieves 65.12% Pass@1 using the same 7B model. This outperforms all repair methods by over 6 percentage points, including the best repair configuration that uses a 10× larger model (72B) with 34 total samples plus solver overhead.

Figure 2 visualizes this comparison. The 6.26 percentage point gap between Best-of-2 and the best repair method is substantial, especially considering that Best-of-2 requires no solver feedback, no IIS computation, and uses a much smaller model. This suggests that for this benchmark, regenerating the model is more effective than attempting to repair infeasible outputs.

4.4 Analysis: Why the Hypothesis Fails

Figure 3 reveals why dual-ray ranking does not help: successful repairs concentrate overwhelmingly in small-IIS instances where ranking is irrelevant. Of the 7 instances repaired by the best configuration, 6 have IIS size ≤ 4, meaning the full IIS fits within the context budget $K = 10$ without truncation. The single truncation-regime repair (ComplexLP_86, IIS size=6) required the 72B model and iterative refinement.

The fundamental issue is that the truncation regime—where dual-ray ranking could theoretically provide value—is precisely where repair is most difficult. These instances have complex constraint interactions that current LLMs struggle to resolve regardless of how the feedback is structured. The hypothesis that better constraint ranking would unlock repair success in this regime is refuted: the bottleneck is not which constraints to show, but the model's ability to reason about constraint modifications.

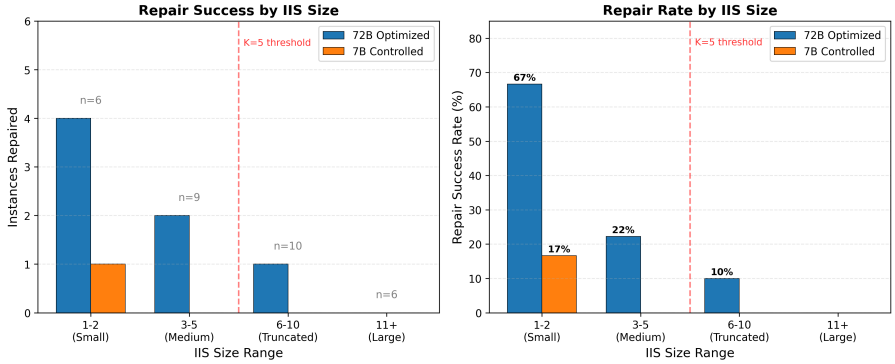

Figure 3: Repair success rate by IIS size. Left: absolute counts of repaired instances. Right: repair success rate (%). The dashed red line marks $K = 5$, the truncation threshold. Repairs concentrate in small-IIS instances (1–2 constraints), with near-zero success in the truncation regime (IIS > 5).

5 Conclusion

We investigated whether Farkas dual ray multipliers can improve LLM-based repair of infeasible optimization models by providing better constraint ranking when IIS feedback must be truncated. Our controlled experiments show that DualRayRank produces identical results to baseline IIS-TopK: both achieve 1/31 repair rate under matched conditions, repairing the same instance. The truncation regime where ranking could theoretically help shows 0/16 repair success for all methods. Furthermore, simple Best-of-2 inference scaling (65.12%) outperforms all repair methods including optimized configurations with 10$\times$ larger models (58.86%).

Limitations. Our evaluation is limited to a single benchmark (MAMO-Optimization) with only 31 infeasible instances, constraining statistical power. The findings may not generalize to other optimization domains or model families.

Future Directions. More effective repair may require richer feedback signals (e.g., feasibility relaxations, constraint explanations), multi-turn dialogue, or training-based approaches that learn from solver feedback. Our negative result suggests that inference-time feedback format alone is insufficient for this challenging task.

References

Ali AhmadiTeshnizi, Wenzhi Gao, and Madeleine Udell. Optimus: Scalable optimization modeling with (mi)lp solvers and large language models. *ArXiv*, abs/2402.10172, 2024.

E. Andersen and Mosek ApS. How to use farkas' lemma to say something important about infeasible linear problems. 2015.

Ruicheng Ao, David Simchi-Levi, and Xinshang Wang. Optirepair: Closed-loop diagnosis and repair of supply chain optimization models with llm agents. 2026a.

Ruicheng Ao, David Simchi-Levi, and Xinshang Wang. Solver-in-the-loop: Mdp-based benchmarks for self-correction and behavioral rationality in operations research. *ArXiv*, abs/2601.21008, 2026b.

Hao Chen, Gonzalo E. Constante-Flores, and Canzhou Li. Diagnosing infeasible optimization problems using large language models. *INFOR: Information Systems and Operational Research*, 62: 573 – 587, 2023.

Dekun Dai, Mingwei Liu, Anji Li, Jialun Cao, Yanlin Wang, Chong Wang, Xing Peng, and Zibin Zheng. Feedbackeval: A benchmark for evaluating large language models in feedback-driven code repair tasks. *ArXiv*, abs/2504.06939, 2025.

Chenyu Huang, Zhengyang Tang, S. Hu, Ruoqing Jiang, Xin Zheng, Dongdong Ge, Benyou Wang, and Zizhuo Wang. Orlm: A customizable framework in training large models for automated optimization modeling. *Oper. Res.*, 73:2986–3009, 2024a.

Xuhan Huang, Qingning Shen, Yan Hu, Anningzhe Gao, and Benyou Wang. Llms for mathematical modeling: Towards bridging the gap between natural and mathematical languages. pp. 2678–2710, 2024b.

Caigao Jiang, Xiang Shu, Hong Qian, Xingyu Lu, Jun Zhou, Aimin Zhou, and Yang Yu. Llmopt: Learning to define and solve general optimization problems from scratch. *ArXiv*, abs/2410.13213, 2024.

Hongliang Lu, Zhonglin Xie, Yaoyu Wu, Can Ren, Yuxuan Chen, and Zaiwen Wen. Optmath: A scalable bidirectional data synthesis framework for optimization modeling. *ArXiv*, abs/2502.11102, 2025.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, S. Welleck, Bodhisattwa Prasad Majumder, Shashank Gupta, A. Yazdanbakhsh, and Peter Clark. Self-refine: Iterative refinement with self-feedback. *ArXiv*, abs/2303.17651, 2023.

Rindranirina Ramamonjison, Timothy T. L. Yu, Raymond Li, Haley Li, G. Carenini, Bissan Ghaddar, Shiqi He, Mahdi Mostajabdaveh, Amin Banitalebi-Dehkordi, Zirui Zhou, and Yong Zhang. Nl4opt competition: Formulating optimization problems based on their natural language descriptions. pp. 189–203, 2023.

Noah Shinn, Federico Cassano, Beck Labash, A. Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: language agents with verbal reinforcement learning. 2023.

Qwen An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Guanting Dong, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxin Yang, Jingren Zhou, Junyang Lin, Kai Dang, Keming Lu, Keqin Bao, Kexin Yang, Le Yu, Mei Li, Mingfeng Xue, Pei Zhang, Qin Zhu, Rui Men, Runji Lin, Tianhao Li, Tingyu Xia, Xingzhang Ren, Xuancheng Ren, Yang Fan, Yang Su, Yi-Chao Zhang, Yunyang Wan, Yuqi Liu, Zeyu Cui, Zhenru Zhang, Zihan Qiu, Shanghaoran Quan, and Zekun Wang. Qwen2.5 technical report. *ArXiv*, abs/2412.15115, 2024.

Zhicheng YANG, Yinya Huang, Zhijiang Guo, Wei Shi, Liang Feng, Linqi Song, Yiwei Wang, Xiaodan Liang, and Jing Tang. Optibench meets resocratic: Measure and improve llms for optimization modeling. 2024.

E. Zelikman, Yuhuai Wu, and Noah D. Goodman. Star: Bootstrapping reasoning with reasoning. 2022.