How Much Can Large Language Models Help Solve MIPs?

Yicheng Huang[*1,*], Wenzhi Gao[†2,*], Dongdong Ge[‡3], Madeleine Udell[§2], and Yinyu Ye[¶2]

[1]Shanghai University of Finance and Economics
[2]Stanford University
[3]Cardinal Operations

November 26, 2026

Abstract

Large language models (LLMs) have recently made dramatic progress on problems in theoretical mathematics, including many that were previously open. Can they also help solve open instances of mixed-integer programming (MIP)? Solving a difficult MIP typically requires a combination of mathematical reasoning, numerical computation, and optimization software. Although LLMs have been studied for a range of operations research tasks, relatively little work has directly evaluated their ability to solve MIP instances. In this technical report, we introduce `LLM4MIP`, a systematic benchmark for existing LLMs on 217 open instances from `MIPLIB` 2017. Our results show that LLMs can accelerate the solution process by identifying and exploiting instance-specific structure. They use this structure to design heuristics, derive cutting planes, and even sometimes construct mathematical proofs. These results suggest that LLMs have the potential to significantly lower the barrier to developing customized solution methods for difficult MIPs.

1 Introduction

Mixed integer programming [44, 39] is arguably the most widely used mathematical optimization framework, with applications throught production planning [34], combinatorial optimization [45], energy management [46], and revenue management [6]. Although MIP is a powerful modeling tool, they are often hard to solve even for state-of-the-art commercial solvers [12, 15, 9, 32]. To solve an instance to optimality, a solver must find a feasible solution and certify that no better one exists, and a general-purpose solver implements a number of heuristics and cutting planes to adapt to instance-specific structures. Once a new structure is understood, it can be internalized by the solver and generalized to future instances upon detection. Modern commercial solvers have built up a pool of such structures through years of exposure to diverse customer instances. Despite years of development, solvers can still struggle when facing new MIP instances with unseen structures. In the pre-LLM era, structure investigation requires multiple areas of expertise, including mathematical reasoning, algorithm design, and familiarity with optimization software. Consequently, even for a single instance, identifying a useful structure and turning it into a method that speeds up the solve can take considerable human effort.

Recently, large language models have demonstrated striking capabilities in mathematical reasoning. Many open problems across theoretical and applied mathematics have seen progress with the help of LLMs [31, 30, 3, 2, 4]. On the other hand, LLMs have also long been recognized as powerful tools for coding and algorithm design [7, 37, 29]. Given that some recent studies have also demonstrated LLMs' potential on operations research tasks

[*]huangyc.opt@gmail.com, the first two authors co-lead the project
[†]gwz@stanford.edu
[‡]dongdong@shanshu.ai
[§]udell@stanford.edu
[¶]yyye@stanford.edu

[22, 19], modern LLMs appear to have most of the expertise needed to identify useful structure in MIPs. Given the limited research directly benchmarking LLMs on solving MIP instances, a natural question arises:

Can LLMs help solve MIPs?

We answer this question in the affirmative through the `LLM4MIP` project.

- We conduct a systematic benchmark of LLMs' ability to solve real-world MIPs, covering 217 open `MIPLIB` instances [13] (10/7/2026). Our results show that LLMs can identify useful instance-specific structure and exploit it to design heuristics, derive cutting planes, and even develop proofs tailored to individual instances.
- We distill reusable skills from our work on the 217 `MIPLIB` instances and openly release all artifacts (`https://llm4mip.github.io`) produced during the LLM-assisted solution process. An ablation study on the test set shows that these skills can transfer to other MIP instances.

1.1 Related work

There is a growing body of research on LLMs for operations research and optimization. Due to space limits, we focus on closely related work and refer interested readers to more comprehensive surveys [48, 14, 41, 24, 10, 49].

LLMs for operations research. Much recent work on LLMs for OR focuses on optimization modeling, with agent-based frameworks [1, 47], fine-tuning approaches [17], and reinforcement learning approaches [40]. Other works use LLMs to propose candidate solutions or design heuristics for combinatorial optimization [50, 23, 52]. Another notable recent example is FrontierOR, which evaluates the solution quality and runtime of LLM-designed algorithms [19, 43].

LLMs for optimization solvers. A particularly related line of research investigates how LLMs can assist existing solver-based workflows. Machine learning has been used to guide branching [18, 11], find primal solutions [28, 56], and select cutting planes [33]. Recent work uses LLMs to design solver procedures such as branching policies [16, 55, 58]. LLMs have also been used to configure cutting-plane separators [21] and to tune parameters for individual instances, often using solver logs [53, 27, 26, 57]. Other applications include MIP re-optimization and SMT solver tuning [25, 42]. Some works use LLMs to generate families of inequalities or to search for solver-efficient MIP formulations [51, 54]. A few recent projects also use LLMs to write numerical optimization solvers from scratch [20, 35], each achieving promising performance compared to open source solvers.

Concurrent results. After our project website was online, Anthropic applied similar ideas to open `MIPLIB` instances with Claude agents and reported improvements to best-known feasible solutions [8]. `Gurobi` also benchmarked this approach on `MIPLIB` instances and provided insights on how LLMs can help MIP solving [38].

Structure of the paper. The rest of this technical report is organized as follows. In **Section 2**, we discuss our prompting approaches and the skill-distillation process. **Section 3** covers the detailed experiment results and representative case studies. We discuss the potential impact of LLMs on future MIP solvers in **Section 4**.

2 Large language model assisted MIP solving

This section introduces the LLM-based workflow that is used throughout our experiments.

2.1 Initial screening and preprocessing

We start by extracting the detailed statistics for 217 open `MIPLIB` instances from `https://miplib.zib.de/tag_open.html` in `csv` format. Then we use a single prompt to let the LLM go through these instance statistics and determine

- a ranking of open instances by the confidence (out of 100) for solving each of them, and
- the possible entry point of the improvement (primal or dual).

LLM is allowed to access the `MIPLIB` website during the initial screening process. After the initial screening, 17 instances are considered to have a high confidence score (≥ 60), 57 to have a medium score (45–59), and the remaining 143 to have a low score (< 45). **Figure 1** shows the distribution of the confidence scores. **Appendix A** lists the scores for individual instances. The screening procedure is limited to 3600 seconds.

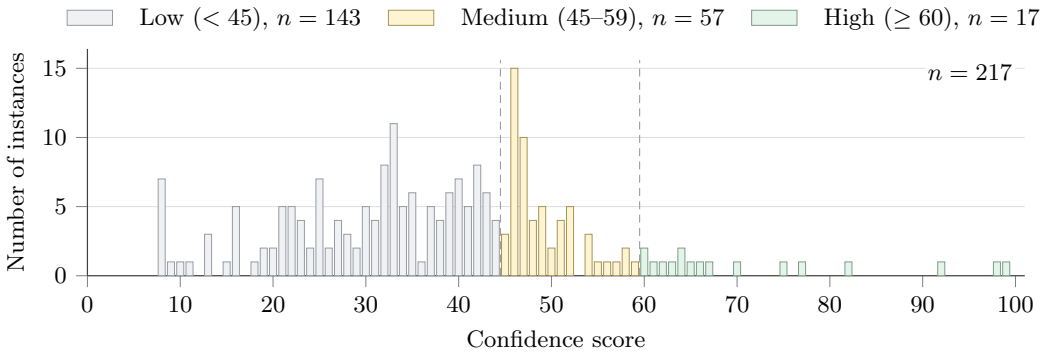


Figure 1: Distribution of heuristic confidence scores for the 217 open `MIPLIB` 2017 instances after initial screening. Each bar shows the number of instances at one integer score.

2.2 Per-instance prompting and tools

Given a MIP instance in `mps` format [13], detailed instance statistics are collected from `MIPLIB` (**Figure 2**). After that, the LLM is asked to solve the instance based on all available information and tools, including

Results from previous screening. LLM is given the available best-known solution from `MIPLIB`. It also has access to the confidence score and the entry point information obtained from the screening process.

Internet and literature search. LLM is allowed to search online for additional details on the problem background and previous attempts, including web forum discussions and related research papers.

Numerical solvers. LLM is allowed to invoke optimization solvers including `Gurobi` and `COPT`, and is also has the free to install additional open-source (e.g., `OR-tools`) in the workspace.

The prompt is simple:

```
Please solve the open MIPLIB instance {NAME}.
The previous screening result is given in screening.csv.
Feel free to look for additional information online and use solvers. {SOLVER} is
installed locally. Try exploiting the instance structure and use solvers sparingly.
```

To prevent an LLM from calling a solver directly without exploiting structure, we explicitly ask it to use solvers sparingly. Each LLM-based solve is allowed a time limit of 7200 seconds.

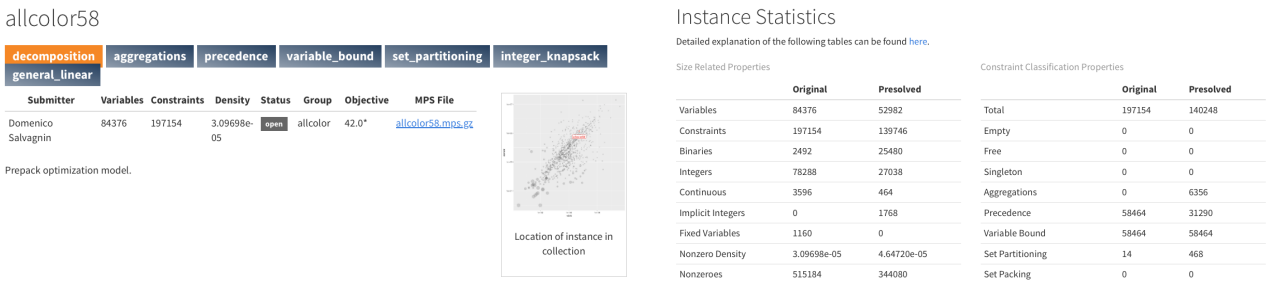


Figure 2: `MIPLIB` instance statistics for `allcolor58`.

2.3 Result validation

After attempts are made on an instance, there are following possible cases of claimed improvements. The first two kinds of improvements are made on the upper bound, and they can be mechanically verified.

Primal solution improvement. Validating a primal solution can be done by plugging in the claimed solution and verifying that no constraint is violated. We use `Gurobi` and `COPT` for validation: a solution is considered valid when both solvers can accept the solution as a valid warm start. Then the new primal solution is recorded.

Unbounded instance. This case is rare (only on `supportcase35`), and LLM provides an improving ray for the problem. Concretely, given an instance with objective vector c, the LLM finds a direction d and a feasible solution x such that $x + \alpha d$ is feasible for all $\alpha \geq 0$ and $\langle c, d \rangle < 0$. Then objective tends to $-\infty$ as $\alpha \to +\infty$.

The last two types of improvements are made on the lower bound, and they are much more challenging to verify.

Dual bound improvement. LLMs can report the dual bound progress arising from different sources, including reformulation, customized cutting plane/valid constraint generation, logical reasoning, and theorem proof.

The trustworthy case is when a dual improvement can be certified by a solver running on the original instance. Unfortunately, it is almost never the case, and most of the dual bound claims require human verification.

To reduce the effort of manual verification, we

- cross-check the claimed dual bound improvement independently with different LLMs.
- ask LLM to generate `LEAN` proof for the proposed cutting planes/theorems.

Humans then manually verify these results. See the experiment section for the progress of manual verification.

Infeasible instance. Infeasible instances have similar verification barriers as dual bound improvements.

2.4 (Iterative) skill distillation

After solving the instances, we save the detailed reasoning process of LLMs and distill structures that can be used on other instances. This distillation process yields skill packages that can be conveniently loaded into LLMs. We distill two skill sets for primal and dual bound improvements, respectively; each skill tells the LLM what structure to look for, how to exploit it, and how to verify the result on the original instance.

Iterative distillation. Whenever a few more instances are solved, the skills can be iteratively updated. We mock the process of solver development by adding useful structures and recording failed attempts from the new solves. This iterative process makes it easier for the LLM to solve future instances with similar structures (**Figure 3**). In our experiments, we split the 217 `MIPLIB` instances into a first group of 132 and a second group of 85. The group of 132 instances contains all instances with high-to-medium confidence scores. We then use the skills from the first 132 instances to help solve the remaining 85, which have lower confidence scores overall.

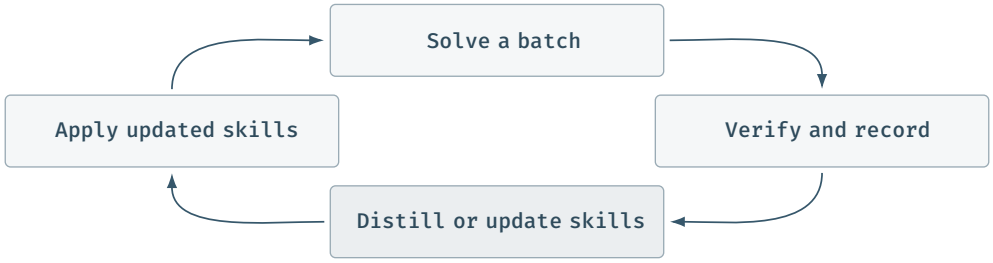


Figure 3: Iterative skill distillation.

3 Experiments

Experimental setup. We use `Codex GPT-5.6-Sol` as the base language model and run a single `Codex` session per instance. Each instance has a wall-clock budget of 7200 seconds, including time for structural analysis, solver calls, and solution validation. For MIP solver calls, `COPT` or `Gurobi` is used by default.

The primal baseline is the objective value of the best-known solution reported in `MIPLIB` v36 (see **Figure 2**). For dual improvements, the baseline is the final dual bound reported by `COPT` with a 10-hour time limit per instance. Following **Section 2.3**, we certify optimality through `LEAN`-based mathematical proofs or solver-based numerical verification. In numerical verification, a solution is classified as optimal within tolerance if the relative optimality gap is below 10^{-6}, where the gap between primal bound P and dual bound D is defined as

$$\mathsf{Gap}(P, D) := \frac{P - D}{\max\{1, |P|, |D|\}} \tag{1}$$

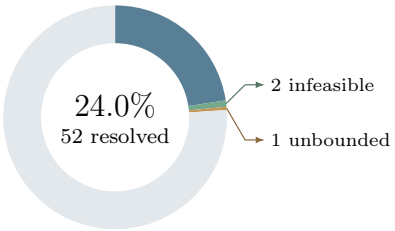
for a minimization problem.

3.1 Results on open `MIPLIB` instances

Table 1 summarizes the results on `MIPLIB`*. The overall performance of the workflow is surprising: LLMs achieve primal-bound improvement on 59 instances and dual-bound improvement on 139. Together, these improvements resolved 49 open `MIPLIB` instances, with three additional infeasible and unbounded instances. For the instances proven optimal, 25 already have optimal solutions in `MIPLIB`; therefore, their bottleneck is on the dual side.

Table 1: Results on 217 `MIPLIB` instances

Result	Count
Instances studied	217
Optimal	49
Infeasible	2
Unbounded	1
Unresolved	165
New or improved solutions	59
Improved dual bounds	136



Sources of primal and dual improvements. **Table 2** groups the improvements by the main method used to find the final feasible solution or establish the final dual bound. The detailed explanation for each solution method is summarized below. Perhaps the most surprising fact is that LLMs demonstrate strong capability of approaching MIP instances from very different perspectives, ranging from common techniques such as local

*The detailed instance results are available at `https://llm4mip.github.io/`.

search and warmstarting, to more complicated techniques such as reformulation and providing customized cutting planes. Some instances are solved with direct mathematical reasoning without even calling a solver. The experiments demonstrate LLMs' superior performance in exploiting instance-dependent structures.

Table 2: Source of improvement on `MIPLIB` instances

Method (click for explanation)	Instances	Representative instances (click `instance` to download)
Primal improvements (59 instances)		
Local search	31	`nj1`, `allcolor58`
Reformulate or decompose the model	13	`liu`, `bharat`
Repair a known solution	8	`circ10-3`, `fhnw-schedule-paira200`
Direct logical reasoning	3	`bmocbd3`, `neos-3208254-reiu`
Warmstarting a standard MIP solver	2	`nag`, `ns1905797`
Look for solution from literature	2	`fastxgemm-n3r21s3t6`, `fastxgemm-n3r22s4t6`
Dual-bound improvements (136 instances)		
Bounds from relaxations or subproblems	31	`ns930473`, `fhnw-schedule-paira200`
Add valid inequalities (cuts)	25	`fastxgemm-n3r21s3t6`, `eva1aprime6x6opt`
Prove a bound by counting or arithmetic	24	`liu`, `sorrell7`
Reformulate the model	21	`momentum3`, `cmflsp60-36-2-6`
Branch on the objective value	21	`graph40-40-1rand`, `bppc6-06`
Global bound from a standard solver	12	`cmflsp40-36-2-10`, `ger50-17-trans-pop-3t`
Use a bound from prior work	2	`circ10-3`, `a2864-99blp`

Primal improvements.

Local search. Keep most binary fixed and try changes to a selected subset and solve a sub-MIP.

Reformulate or decompose the model. Solve a more compact formulation of the original problem or a set of smaller problems when the problem can be decomposed.

Repair a known solution. Extrapolate a known solution over from a related model and repair violations. For example, the solution for `fhnw-schedule-paira200` holds for other instances from the `fhnw-schedule` family.

Direct logical reasoning. Reason what the constraints describe and use those rules to put together a feasible solution. For `neos-3208254-reiu`, a binary tree was built, and its parent and ancestor relationships were translated into values for the original variables.

Warmstarting a standard MIP solver. Use the feasible solution found by the solver's own search. It can start from an existing candidate and continue looking for improvements.

Look for solution from literature. Modify an existing construction from the literature to fit the instance.

Dual-bound improvements.

Bounds from relaxations or subproblems. Solve a relaxation of the original problem or a few subproblems.

Add valid inequalities (cuts). Add constraints that every feasible solution already satisfies.

Prove a bound by counting or arithmetic. Use the background of the original problem to reason how good any feasible solution can be. For `liu`, the total area to be packed and the possible side lengths imply a lower bound on the size of the enclosing square.

Reformulate the model. Rewrite the model and establish that its best objective value is preserved. A bound obtained from the new formulation then applies to the original problem.

Branch on the objective value. Add a constraint of the form $\langle c, x \rangle \leq z$ and prove the resulting model infeasible.

Global bound from a standard solver. Use the bound a MIP solver reports for the original instance.

Use a bound from prior work. Match the instance to a problem already studied in the literature.

Can LLMs predict whether an instance can be closed? We compare the initial confidence scores with the final outcomes for the same 217 instances. As shown in **Figure 4**, the solve rate rises from 17.5% (25 of 143) in the low-confidence group to 29.8% (17 of 57) in the medium-confidence group and 58.8% (10 of 17) in the high-confidence group. The low-confidence group accounts for nearly half of all resolutions (25 of 52) since it contains many more instances. Therefore, the experiment suggests that even with a few minutes of study, LLMs can already roughly judge the possibility of closing a hard MIP instance.

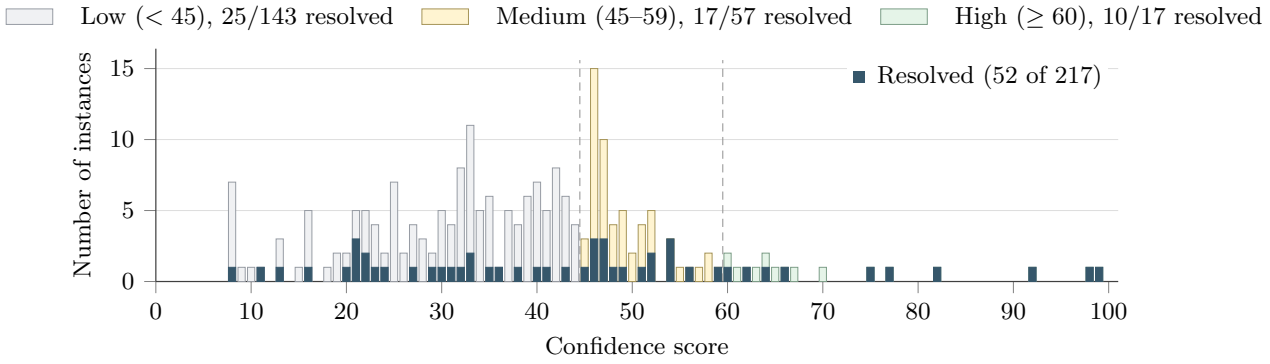


Figure 4: Total and resolved instances by initial confidence score. The dark bars show how many of them were resolved, and the dashed lines separate the confidence bands.

3.2 Distilled skill comparison

After distilling the initial versions of the primal and dual skills, we evaluate them on 20 unseen instances from `MIPLIB` collection dataset (that are not open instances).[†] The setup is the same as before, except for a longer time limit of 9000 seconds.

Table 3 summarizes the the comparison. Compared with the LLM without skills, the primal skill improves the objective on 10 instances, ties on nine, and loses on one. The dual skill wins 17, ties 1, and loses 2; compared to direct solving with `Gurobi`, the overall gap wins are 17 of the instances.

Table 3: Effect of skills on 20 instances.

Compared with	Comparison results (wins/ties/losses)			Mean gap reduction (pp)
	Primal	Dual	Gap	
LLM without skills	9/10/1	17/1/2	18/0/2	15.8079
`Gurobi` (150 minutes)	—	—	17/1/2	19.2762

Our results preliminarily show that the skills often generalize to new MIP instances.

[†]Further details on the skill comparison are available at `https://llm4mip.github.io/skill/`.

4 Discussions and conclusions

We present `LLM4MIP`, the first systematic benchmark of LLMs' ability to exploit structures MIP instances. Our findings are encouraging: LLMs can already handle complex MIPs and effectively identify useful instance-specific structures. However, we must also point out several limitations of the current LLM-based approaches.

Distribution of open instances. As noted in [38], open instances may represent only part of the range of MIPs solved in practice. In our experiments, many dual-bound improvements rely on specialized methods that may apply only to a narrow family of instances. This limited scope can make the cost of implementing such methods in a commercial solver hard to justify.

Hardness of dual certificate. Unlike primal bound improvements, which are easy to verify, most dual-side improvements cannot be trivially justified. Unlike systematic frameworks such as branch-and-cut and other relaxation-based approaches, obtaining a certificate by mathematical reasoning or theorem proof is uncommon and still requires massive human intervention. We envision that future MIP solving may finally integrate these non-numerical certificates, but doing it reliably still requires more effort. Some recent attempts have made progress in this direction [36, 5].

Privacy. A thorough investigation of a MIP instance can require sharing its data or formulation with an LLM. When the data are sensitive, or the formulation is proprietary, sharing them with a commercial LLM provider can raise privacy and confidentiality concerns.

Finally, we discuss the possible implications of LLMs for future solver development. For users, LLMs are likely to significantly lower the barrier for non-experts to solving complicated real-world MIPs. In some cases, an LLM could take on work that would otherwise require a team of algorithm engineers. Users can easily customize an open-source solver to their needs, or even write a solver from scratch.

From a solver-development perspective, solver development is likely to be more effective. Solver development is similar to the skill distillation process used in the experiments, where methods discovered while solving individual instances can be refined and reused on related problems. A key challenge is to coordinate these methods within a solver while keeping it reliable and easy to use. For example, solvers may develop more callback interfaces compatible with complex structure exploitation.

Can LLMs help solve MIPs? Our answer is "Yes, a lot".

References

[1] Ali AhmadiTeshnizi, Wenzhi Gao, and Madeleine Udell. Optimus: Scalable optimization modeling with (mi) lp solvers and large language models. *arXiv preprint arXiv:2402.10172*, 2024. (cited on 2)

[2] Levent Alpöge. [announcement of a proof of the Riemann hypothesis]. Author's website, 2026. URL https://alpo.ge/remark.pdf. November 20. (cited on 1)

[3] Anthropic. Claude solves two open problems in the physics of black holes. Anthropic Research, 2026. URL https://www.anthropic.com/research/claude-black-hole-physics. November 17. (cited on 1)

[4] Anthropic. Learning more about Claude's mathematical capabilities. Anthropic Research, 2026. URL https://www.anthropic.com/research/riemann-zeta. August 10. (cited on 1)

[5] Haozhen Bai, Wenjie Zhu, Qian Hu, and Caihua Chen. RPX: Certifying agent-generated MILP reformulations at scale. *arXiv preprint arXiv:2611.15035*, 2026. URL https://arxiv.org/abs/2611.15035. (cited on 8)

[6] Stefano Benati and Romeo Rizzi. A mixed integer linear programming formulation of the optimal mean/value-at-risk portfolio problem. *European Journal of Operational Research*, 176(1):423–434, 2007. (cited on 1)

[7] Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde De Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021. (cited on 1)

[8] Claude. MIPLIB is a library of optimization problems, such as how best to schedule airline crews and delivery trucks. X post by claudeai, November 2026. URL https://x.com/claudeai/status/2119899477855379913. Posted November 9, 2026. Accessed November 24, 2026. (cited on 2)

[9] IBM ILOG Cplex. V12. 1: User's manual for cplex. *International Business Machines Corporation*, 46(53): 157, 2009. (cited on 1)

[10] Francesca Da Ros, Michael Soprano, Luca Di Gaspero, and Kevin Roitero. Large language models for combinatorial optimization: A systematic review. *ACM Computing Surveys*, 58(11):1–53, 2026. (cited on 2)

[11] M. Gasse, D. Chételat, N. Ferroni, L. Charlin, and A. Lodi. Exact combinatorial optimization with graph convolutional neural networks. In *Advances in Neural Information Processing Systems*, volume 32, 2019. URL https://proceedings.neurips.cc/paper/2019/hash/d14c2267d848abeb81fd590f371d39bd-Abstract.html. (cited on 2)

[12] D. Ge, Q. Huangfu, Z. Wang, J. Wu, and Y. Ye. *Cardinal Optimizer (COPT) User Guide*, 2023. URL https://guide.coap.online/copt/en-doc/intro.html. (cited on 1)

[13] Ambros Gleixner, Gregor Hendel, Gerald Gamrath, Tobias Achterberg, Michael Bastubbe, Timo Berthold, Philipp Christophel, Kati Jarck, Thorsten Koch, Jeff Linderoth, et al. MIPLIB 2017: data-driven compilation of the 6th mixed-integer programming library. *Mathematical Programming Computation*, 13(3): 443–490, 2021. (cited on 2, 3)

[14] Tong Guo, Yi Mei, Yaoxin Wu, Chin Sheng Tan, Huey Yuen Ng, and Yew Soon Ong. From natural language to optimization models: A survey of llm-based autoformulation. *Available at SSRN 7204798*, 2026. (cited on 2)

[15] Gurobi Optimization, LLC. *Gurobi Optimizer Reference Manual*, 2026. URL https://docs.gurobi.com/projects/optimizer/en/current/index.html. (cited on 1)

[16] Zhinan Hou, Xingchen Li, Yankai Zhang, Tianxun Li, and Keyou You. LLM4branch: Large language model for discovering efficient branching policies of integer programs. In *Forty-third International Conference on Machine Learning*, 2026. URL https://openreview.net/forum?id=YIzUTHEvG7. (cited on 2)

[17] Chenyu Huang, Zhengyang Tang, Shixi Hu, Ruoqing Jiang, Xin Zheng, Dongdong Ge, Benyou Wang, and Zizhuo Wang. Orlm: A customizable framework in training large models for automated optimization modeling. *Operations Research*, 73(6):2986–3009, 2025. (cited on 2)

[18] Elias Khalil, Pierre Le Bodic, Le Song, George Nemhauser, and Bistra Dilkina. Learning to branch in mixed integer programming. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 30, pages 724–731, 2016. doi: 10.1609/aaai.v30i1.10080. (cited on 2)

[19] Minwei Kong, Chonghe Jiang, Ao Qu, Wenbin Ouyang, Zhaoming Zeng, Xiaotong Guo, Zhekai Li, Junyi Li, Yi Fan, Xinshou Zheng, et al. Frontieror: Benchmarking llms' capacity for efficient algorithm design in large-scale optimization. *arXiv preprint arXiv:2605.25246*, 2026. (cited on 2)

[20] Milan Korda. BRISK: An interior-point solver for conic programming. https://homepages.laas.fr/mkorda/brisk.html, 2026. Source code available at https://github.com/MilanKorda/brisk. (cited on 2)

[21] Connor Lawless, Yingxi Li, Anders Wikum, Madeleine Udell, and Ellen Vitercik. Llms for cold-start cutting plane separator configuration. In *International Conference on the Integration of Constraint Programming, Artificial Intelligence, and Operations Research*, pages 51–69. Springer, 2025. (cited on 2)

[22] Zhong Li, Hongliang Lu, Tao Wei, Yuxuan Chen, Wenyu Liu, Yuan Lan, Fan Zhang, and Zaiwen Wen. Constructing industrial-scale optimization modeling benchmark. *arXiv preprint arXiv:2602.10450*, 2026. (cited on 2)

[23] F. Liu, X. Tong, M. Yuan, X. Lin, F. Luo, Z. Wang, Z. Lu, and Q. Zhang. Evolution of heuristics: Towards efficient automatic algorithm design using large language model. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pages 32201–32223, 2024. URL https://proceedings.mlr.press/v235/liu24bs.html. (cited on 2)

[24] Fei Liu, Yiming Yao, Ping Guo, Zhiyuan Yang, Xi Lin, Zhe Zhao, Xialiang Tong, Kun Mao, Zhichao Lu, Zhenkun Wang, et al. A systematic survey on large language models for algorithm design. *ACM Computing Surveys*, 58(8):1–32, 2026. (cited on 2)

[25] Ruohong Liu, Xuan He, Yuxin Pan, and Jack Umenberger. Online algorithm configuration for MILP re-optimization with LLM guidance. OpenReview, 2026. URL https://openreview.net/forum?id=xbyebbS1ZF. Submission to ICLR 2026. (cited on 2)

[26] Yidong Luo, Chenguang Wang, Fangzhou Zhu, Tao Zhong, and Tianshu Yu. Grimip: A general framework for instance-specific configuration of mip solvers using llms. *arXiv preprint arXiv:2606.23299*, 2026. (cited on 2)

[27] Yidong Luo, Weihuang Wen, Xuanhao Pan, Yinya Wang, Shanwen Pu, Chenghao Lan, Dongdong Ge, and Tianshu Yu. Beyond the black box: A diagnosis-driven agent for interpretable mip solver tuning. *arXiv preprint arXiv:2611.30670*, 2026. URL https://arxiv.org/abs/2611.30670. (cited on 2)

[28] Vinod Nair, Sergey Bartunov, Felix Gimeno, Ingrid Von Glehn, Pawel Lichocki, Ivan Lobov, Brendan O'Donoghue, Nicolas Sonnerat, Christian Tjandraatmadja, Pengming Wang, et al. Solving mixed integer programs using neural networks. *arXiv preprint arXiv:2012.13349*, 2020. (cited on 2)

[29] A. Novikov, N. Vũ, M. Eisenberger, E. Dupont, P.-S. Huang, A. Z. Wagner, et al. AlphaEvolve: A coding agent for scientific and algorithmic discovery. arXiv preprint arXiv:2506.13131, 2025. URL https://arxiv.org/abs/2506.13131. (cited on 1)

[30] OpenAI. Ten advances in mathematics and theoretical computer science. OpenAI Research, 2026. URL https://openai.com/index/ten-advances-in-mathematics/. August 1. (cited on 1)

[31] OpenAI. An OpenAI model has disproved a central conjecture in discrete geometry. OpenAI Research, 2026. URL https://openai.com/index/model-disproves-discrete-geometry-conjecture/. May 20. (cited on 1)

[32] FICO Xpress Optimizer. v8. 11 reference manual, 2021. (cited on 1)

[33] Max B Paulus, Giulia Zarpellon, Andreas Krause, Laurent Charlin, and Chris Maddison. Learning to cut by looking ahead: Cutting plane selection via imitation learning. In *International conference on machine learning*, pages 17584–17600. PMLR, 2022. (cited on 2)

[34] Yves Pochet and Laurence A Wolsey. *Production planning by mixed integer programming*, volume 149. Springer, 2006. (cited on 1)

[35] Marcin Polkowski. popso: Polkowski's optimization solver, 2026. URL https://github.com/gozwei/popso. (cited on 2)

[36] Henry Robbins, Connor Lawless, Madeleine Udell, and Ellen Vitercik. Flare: Verifying milp reformulations with llm-based theorem proving. *arXiv preprint arXiv:2608.25220*, 2026. (cited on 8)

[37] Bernardino Romera-Paredes, Mohammadamin Barekatain, Alexander Novikov, Matej Balog, M Pawan Kumar, Emilien Dupont, Francisco JR Ruiz, Jordan S Ellenberg, Pengming Wang, Omar Fawzi, et al. Mathematical discoveries from program search with large language models. *Nature*, 625(7995):468–475, 2024. (cited on 1)

[38] Edward Rothberg. Recent advances in the use of large language models to solve mixed-integer programming models. Gurobi Optimization Blog, November 2026. URL https://www.gurobi.com/resources/blog/recent-advances-in-the-use-of-large-language-models-to-solve-mixed-integer-programming-models. Published November 17, 2026. Accessed November 24, 2026. (cited on 2, 8)

[39] Nikolaos V Sahinidis. Mixed-integer nonlinear programming 2018, 2019. (cited on 1)

[40] Xiang Shu, Hong Qian, Xingyu Lu, JUN ZHOU, Aimin Zhou, Yang Yu, et al. Llmopt: Learning to define and solve general optimization problems from scratch. In *International Conference on Learning Representations*, volume 2025, pages 101580–101606, 2025. (cited on 2)

[41] Chuyue Tian, Ling Wang, Jingfang Chen, Rui Li, and Jianmai Shi. Advances in llm-assisted combinatorial optimization. *Complex System Modeling and Simulation*, 6(3):227–252, 2026. (cited on 2)

[42] Cong Wang, Qiaoyi Zheng, Zhaoguo Wang, and Fan Long. Agent-guided configuration tuning for smt solvers. *arXiv preprint arXiv:2611.20484*, 2026. (cited on 2)

[43] Hai Wang, Bretislav Hajek, Chonghe Jiang, Minwei Kong, Ao Qu, Huatian Gong, Bo Dong, Cynthia Barnhart, and Jinhua Zhao. The frontieror challenge: Agentic algorithm design for operations research. *arXiv preprint arXiv:2611.26171*, 2026. (cited on 2)

[44] Laurence A Wolsey. *Integer programming*. John Wiley & Sons, 2020. (cited on 1)

[45] Laurence A Wolsey and George L Nemhauser. *Integer and combinatorial optimization*, volume 55. John Wiley & Sons, 1999. (cited on 1)

[46] Allen J Wood, Bruce F Wollenberg, and Gerald B Sheblé. *Power generation, operation, and control*. John Wiley & Sons, 2013. (cited on 1)

[47] Ziyang Xiao, Dongxiang Zhang, Yangjun Wu, Lilin Xu, Yuan Wang, Xiongwei Han, Xiaojin Fu, Tao Zhong, Jia Zeng, Mingli Song, et al. Chain-of-experts: When llms meet complex operations research problems. In *International Conference on Learning Representations*, volume 2024, pages 48519–48537, 2024. (cited on 2)

[48] Ziyang Xiao, Jingrong Xie, Lilin Xu, Shisi Guan, Jingyan Zhu, Xiongwei Han, Xiaojin Fu, WingYin Yu, Han Wu, Wei Shi, et al. A survey of optimization modeling meets llms: Progress and future directions. *arXiv preprint arXiv:2508.10047*, 2025. (cited on 2)

[49] Xianchao Xiu, Jianhao Li, Jun Fan, and Wanquan Liu. Large language models for operations research: A comprehensive survey. *arXiv preprint arXiv:2605.20849*, 2026. (cited on 2)

[50] Chengrun Yang, Xuezhi Wang, Yifeng Lu, Hanxiao Liu, Quoc V. Le, Denny Zhou, and Xinyun Chen. Large language models as optimizers. In *International Conference on Learning Representations*, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/hash/3339f19c5fcee3ad74502947a32be9e6-Abstract-Conference.html. (cited on 2)

[51] Milad Yazdani, Mahdi Mostajabdaveh, Samin Aref, and Zirui Zhou. LLM-guided evolution of symbolic acceleration inequality families for MILP. *Transactions on Machine Learning Research*, 2026. URL https://arxiv.org/abs/2508.11850v3. (cited on 2)

[52] H. Ye, J. Wang, Z. Cao, F. Berto, C. Hua, H. Kim, J. Park, and G. Song. ReEvo: Large language models as hyper-heuristics with reflective evolution. In *Advances in Neural Information Processing Systems*, volume 37, 2024. URL https://papers.nips.cc/paper_files/paper/2024/hash/4ced59d480e07d290b6f29fc8798f195-Abstract-Conference.html. (cited on 2)

[53] Zhengzhong Ricky You, Kaizhao Sun, Zeping Min, Wotao Yin, and Xinshang Wang. Log-driven semantic tuning for mixed integer programming solvers. *arXiv preprint arXiv:2611.00621*, 2026. (cited on 2)

[54] H. Yuan, J. Peng, J. Bi, N. Zhang, S. Song, and Z. Cao. FormuEvo: LLM-guided evolution for discovering solver-efficient mixed-integer programming formulations. arXiv preprint arXiv:2608.23353, 2026. URL https://arxiv.org/abs/2608.23353v2. Version 2, 3 October 2026; to appear in EMNLP 2026. (cited on 2)

[55] Ce Zhang, Bin Zhang, Zhiwei Xu, Hao Chen, Xinyue Lu, Shanwei Fan, Yingxuan Teng, and Guoliang Fan. Bife: Search-efficient discovery of cpu-only branching policies via llm-based bi-fidelity evolution. *arXiv preprint arXiv:2609.36735*, 2026. (cited on 2)

[56] Zherong Zhang, Guanlin Li, Chengrui Gao, Haopu Shang, Ke Xue, Jixiang Lu, Weiyong Yang, and Chao Qian. Shsp: Structure-aware hierarchical solution prediction for mixed-integer linear programming. *arXiv preprint arXiv:2608.25282*, 2026. (cited on 2)

[57] Yifan Zhao, Qianyang Zhao, Xinyu Guan, Kai Wei, and Yuming Deng. Nominate-then-adjudicate: Llm-assisted one-pass and per-instance mip solver tuning. *arXiv preprint arXiv:2610.29391*, 2026. (cited on 2)

[58] Yuyan Zhou and Jie Wang. Large language model for efficient algorithm design of combinatorial optimization solver. In *2026 International Joint Conference on Neural Networks (IJCNN)*, pages 1–7. IEEE, 2026. (cited on 2)

A Confidence score after initial screening

Table 4: Ranking of 217 open MIPLIB 2017 instances. Green marks scores ≥ 60, yellow 45–59, and gray < 45.

Instance	Score	Primal	Dual
neos-2629914-sudost	99	15	99
pythago7825	98	5	99
a2864-99blp	92	20	96
allcolor58	82	92	68
genus-sym-g31-8	77	35	82
genus-g31-8	75	35	80
graph40-20-1rand	70	35	76
supportcase39	67	60	66
circ10-3	66	40	72
ex1010-pi	65	42	70
f2000	64	38	70
graph40-40-1rand	64	35	70
ramos3	63	38	69
genus-sym-grafo5708-48	62	35	69
genus-g61-25	61	35	68
bppc6-06	60	61	57
genus-sym-g62-2	60	35	67
assign1-10-4	59	55	60
d20200	58	42	63
comp12-2idx	58	34	59
milo-v13-4-3d-4-0	57	57	55
fhnw-binpack4-58	56	66	42
neos-3603137-hoteo	55	65	40
graph40-80-1rand	54	34	62
dfn-bwin-DBE	54	56	53
fhnw-binschedule2	54	34	49
neos-3691541-lonja	52	24	49
neos-4360552-sangro	52	24	59
neos-5266653-tugela	52	24	49
ns1631475	52	24	49
scpj4scip	52	24	53
graphdraw-mainerd	51	24	43
neos-1420790	51	28	53
neos-3594536-henty	51	24	54
sorrell7	51	44	58
xmas10	50	42	57
scpk4	50	20	49
xmas10-2	49	42	56
neos-4533806-waima	49	24	49
neos-4555749-wards	49	24	49
t1717	49	34	53
t1722	49	34	53
bab3	48	30	55
lr1dr02vc05v8a-t360	48	34	43
neos-3355120-tarago	48	29	51
fhnw-schedule-paira200	48	76	34
core2586-950	47	24	43
core4284-1064	47	24	43
core4872-1529	47	24	43
fhnw-binschedule0	47	30	45
pb-gfrd-pnc	47	24	48
pb-grow22	47	24	48
scpl4	47	20	49
tokyometro	47	28	57
fhnw-schedule-paira400	47	76	34
fhnw-schedule-pairb200	47	76	34
neos-3355323-arnon	46	58	35
dws012-01	46	34	49
fhnw-schedule-paira100	46	34	38
ger50-17-ptp-pop-3t	46	38	42
ger50-17-trans-dfn-3t	46	34	38
ger50-17-trans-pop-3t	46	34	38
graphdraw-grafo2	46	20	39
graphdraw-opmanager	46	24	43
liu	46	38	42
lr2-22dr3-333vc4v17a-t60	46	30	45
milo-v12-6-r1-75-1	46	34	44
neos-1420546	46	24	49
rmine11	46	24	48
sing5	46	20	45
fhnw-schedule-pairb400	46	76	34
gmut-76-40	45	34	43
neos-3009394-lami	45	28	47
neos-3695882-vesdre	45	20	45
nj1	44	61	33
rmine13	44	20	44
sing11	44	20	45
bley_xs1	44	63	43
nj2	43	60	33
lr1dr04vc05v17a-t360	43	34	43
ns1456591	43	28	47
sct1	43	24	44
z26	43	30	49
bley_xs1noM	43	63	43
nj3	42	59	32
ds-big	42	30	49
eva1aprime5x5opt	42	24	43
gmut-76-50	42	34	43
neos-4531126-vouga	42	25	47
scpm1	42	20	49
scpn2	42	16	45
supportcase22	42	20	49
nag	41	28	47
neos-2991472-kalu	41	24	43
neos-5221106-oparau	41	24	43
ns1856153	41	24	43
rmine15	41	20	44
hgms30	40	30	39
neos-3682128-sandon	40	24	53
ns1828997	40	24	48
sct5	40	24	44
set3-09	40	28	42
set3-16	40	28	42
shipsched	40	34	38
ns1905797	39	61	27
eva1aprime6x6opt	39	24	43
ivu06	39	30	49
neos-5045105-creuse	39	28	42
rocII-10-11	39	24	43
rocII-8-11	39	24	43
lr1dr12vc10v70b-t360	38	30	39
ns2122698	38	20	45
ns930473	38	24	43
supportcase23	38	24	44
dano3mip	37	24	38
dc1l	37	34	38
neos-848198	37	24	38
nsr8k	37	34	38
r4l4-02-tree-bounds-50	37	34	38
neos-4562542-watut	36	20	39
cmflsp40-36-2-10	35	24	38
neos-1423785	35	24	38
neos-3634244-kauru	35	20	39
ns2124243	35	20	45
polygonpack4-10	35	39	35
polygonpack4-7	35	39	35
ns1690781	34	24	43
ns1904248	34	20	39
siena1	34	39	35
n3705	34	35	44
n3707	34	35	44
dws012-02	33	34	43
dws012-03	33	34	43
ivu06-big	33	26	45
ivu59	33	26	45
neos-2978205-isar	33	20	34
neos-4292145-piako	33	24	38
neos-4647032-veleka	33	24	43
seqsolve2short4288	33	20	39
seqsolve3. . . excess384	33	20	39
n3709	33	35	44
n370b	33	35	44
cdc7-4-3-2	32	96	28
cdma	32	33	39
cmflsp60-36-2-6	32	24	38
hgms62	32	30	39
neos-4290317-perth	32	29	36
neos-5189128-totara	32	20	39
rmine21	32	16	40
zeil	32	34	38
neos-4355351-swalm	31	24	38
seqsolve1	31	20	39
stockholm	31	24	38
supportcase41	31	24	38
nb10tb	30	25	37
neos-4358725-tagus	30	29	35
polygonpack3-15	30	35	31
polygonpack4-15	30	35	31
polygonpack5-15	30	35	31
cvrpp-n16k8vrpi	29	29	35
neos-4545615-waita	29	20	39
ns1849932	28	20	39
shs1014	28	30	39
shs1042	28	30	39
neos-4230265-orari	27	29	35
rmine25	27	16	40
supportcase31	27	25	37
n3700	27	39	41
neos-5044663-wairoa	26	24	38
sing17	26	20	39
kottenpark09	25	16	45
momentum3	25	29	35
neos-2974461-ibar	25	25	41
neos-4232544-orira	25	25	31
neos-5041822-cockle	25	20	39
gsvm2rl11	25	37	25
supportcase30	25	22	45
gasprod1-1	24	29	30
gsvm2rl12	24	37	25
fhnw-binschedule1	23	26	35
neos-4535459-waipa	23	20	39
rwth-timetable	23	20	39
uccase10	23	20	34
neos-4264598-oueme	22	29	30
pizza27i	22	25	41
pizza78i	22	25	41
fastxgemm-n3r22s4t6	22	20	35
gsvm2rl9	22	35	25
cvrpb-n45k5vrpi	21	25	31
minutedispatchstrategy	21	29	30
snp-04-052-052	21	25	26
supportcase35	21	29	30
fastxgemm-n3r23s5t6	21	20	35
proteindesign. . . p19	20	21	37
tpl-tub-ws1617	20	16	35
neos-3754224-navua	19	25	31
neos-5151569-mologa	19	25	31
fastxgemm-n3r21s3t6	18	20	35
bmocbd3	16	25	31
neos-3615091-sutlej	16	25	31
snp-06-004-052	16	25	26
snp-10-004-052	16	25	26
snp-10-052-052	16	25	26
s82	15	31	27
cvrpa-n64k9vrpi	13	25	31
mining	13	25	26
ns1679495	13	25	31
neos-3208254-reiu	11	16	35
datt256	10	25	12
neos-4285819-pedja	9	21	27
bharat	8	21	22
dlr1	8	16	30
dlr2	8	17	18
rfds-4-days	8	31	22
splan1	8	21	27
supportcase38	8	21	22
nucorsav	8	24	9

B Instance bounds and improvements

Primal improvement | Dual improvement | Reported optimal

Table 5: Primal and dual bounds for all 217 MIPLIB instances.

Instance	Best-known primal MIPLIB v36	COPT dual 10-hour baseline	LLM4MIP primal	LLM4MIP dual
a2864-99blp	−257	−282.960474	−257	−257
allcolor58	258	42	42	42
assign1-10-4	422	377.444386	422	422
bab3	−656214.9542	−657093.783	−656214.9542	—
bharat	4039608.5067	3783727.85	4012280.0882	3850713.4697
bley_xs1	3855895.86	3696540.8	3855895.86	3532608.7406
bley_xs1noM	3873690.77	3713869.16	3855895.86	3344924.0428
bmocbd3	−373568021.47	−373603745	−373587239.24	−373603474.46
bppc6-06	208	201	208	208
cdc7-4-3-2	−307	−381	−307	−381
cdma	-2.48×10^{16}	—	$-2.47784784 \times 10^{16}$	$-2.700669263 \times 10^{16}$
circ10-3	256	141	242	242
cmflsp40-36-2-10	66452234.495	64134678.7	66452236.454	65928415.307
cmflsp60-36-2-6	73891245.385	72075510.1	73891244.222	73054818.854
comp12-2idx	277	180.108268	277	165
core2586-950	944	937.490655	944	937
core4284-1064	1062	1055.87389	1062	1055
core4872-1529	1530	1514.10482	1530	932
cvrpa-n64k9vrpi	1617	0	1401	1401
cvrpb-n45k5vrpi	775	0	751	751
cvrpp-n16k8vrpi	450	282.375839	450	450
d20200	12237	12232.0097	12237	12232
dano3mip	664.02083333	578.852969	664.02083333	578.21718558
datt256	—	256	—	—
dc1l	1758647.6801	1751049.95	1758647.6801	1748926.0308
dfn-bwin-DBE	73623.787887	56406.0914	73623.79	73623.79
dlr1	2708064.137	—	2708064.1319	2707908.7434
dlr2	18311711.897	—	18265839.45	11601762.346
ds-big	194.58674798	87.2719112	194.49841415	178.14457882
dws012-01	82030.530143	47506.9179	82030.552222	44641.723169
dws012-02	119893.31071	58131.4504	119443.31072	23459.584922
dws012-03	140473.57649	63290.0397	140473.62939	114206.30519
eva1aprime5x5opt	−17.948075604	−52.0397187	−17.948075604	−458.91907206
eva1aprime6x6opt	−18.10146979	−140.895641	−18.101002981	−96.703805397
ex1010-pi	233	224.057416	233	224
f2000	1810	1342.95881	1810	1332
fastxgemm-n3r21s3t6	4110.9980012	84	2147	93
fastxgemm-n3r22s4t6	3101.9981984	84	1150	90
fastxgemm-n3r23s5t6	3089.997998	84	153	90
fhnw-binpack4-58[e]	—	0	—	—
fhnw-binschedule0	16088	15642	15958	15958
fhnw-binschedule1	55198	52153	55158	55158
fhnw-binschedule2	2428	2273.5	2428	2428
fhnw-schedule-paira100	−15.113122668	−23.8016842	−15.113116513	−15.113116513
fhnw-schedule-paira200	−19.229856822	−46.9081453	−19.369961205	−19.369961205
fhnw-schedule-paira400	−35.546804653	−95.8844179	−36.276966742	−36.276966742
fhnw-schedule-pairb200	−19.240943843	−51.5610835	−19.369961205	−19.369961205
fhnw-schedule-pairb400	−35.4584	−98.8384308	−36.276966742	−36.276966742

Continued on next page

Table 5 (continued)

Instance	Best-known primal MIPLIB v36	COPT dual 10-hour baseline	LLM4MIP primal	LLM4MIP dual
gasprod1-1	291.17532062	253.271446	291.52653003	291.52653003
genus-g31-8[a]	−23	−27	−23	−23
genus-g61-25	−40	−58	−40	−42
genus-sym-g31-8[a]	−23	−27	−23	−23
genus-sym-g62-2	−38	−54	−38	−40
genus-sym-grafo5708-48	−21	−27	−21	−21
ger50-17-ptp-pop-3t	5224.5144	4123.52638	5224.5144	5183.9359259
ger50-17-trans-dfn-3t	3969.4334	3831.36744	3969.4334	3934.3152
ger50-17-trans-pop-3t	4038.441	3884.84515	4038.441	4038.441
gmut-76-40	−14169477.796	−14170137.1	−14169477.786	−14171086.316
gmut-76-50	−14171893.779	−14173526.5	−14171397.828	—
graph40-20-1rand	−15	−134.42884	−15	—
graph40-40-1rand	−9	−68.9559386	−9	−9
graph40-80-1rand	−7	−495.450445	−7	−7
graphdraw-grafo2	68613.5	36314.1643	68590.5	53513.5
graphdraw-mainerd	38842	31493.2257	38842	31882
graphdraw-opmanager	99445.5	56741.4457	99445.5	58950
gsvm2rl11	18121.637644	3169.43538	18539.127162	2096.7554578
gsvm2rl12	21.903856359	3.11194162	21.903856359	12.099305681
gsvm2rl9	7438.1810212	2889.25545	7438.1811678	2289.6595304
hgms30	−44338.744803	−46112.9057	−44338.744813	−44339.653641
hgms62	−44837.245343	−46639.631	−44837.245347	−44844.328684
ivu06	142.86	135.944621	142.86433158	135.90024232
ivu06-big	140.74	135.430283	140.41233208	135.44891253
ivu59	927.89475182	884.788371	911.94105059	885
kottenpark09	1715	183.5	1715	230
liu	1083.984782	560	1082	1074
lr1dr02vc05v8a-t360	123046.81494	120969.134	123046.81612	—
lr1dr04vc05v17a-t360	252463	247825.753	251302.15104	247852.13666
lr1dr12vc10v70b-t360	1000790.9797	984474.155	999429.32495	985808.99169
lr2-22dr3-333vc4v17a-t60	−27262.589176	−71836.9293	−27640.919482	−30627.159226
milo-v12-6-r1-75-1	1153756.3982	1135852.03	1153756.3982	1125907.0416
milo-v13-4-3d-4-0	358152.26219	253283.451	358152.26219	221230.41434
mining	−833589149.27	−933493233	−833589149.27	−835361895.59
minutedispatchstrategy	3109.9034778	3109.78686	3109.9034778	3109.9034778
momentum3	173202.17622	96179.1231	173202.18557	107995.45322
n3700	1218975	1065546	1247295	1063103.1471
n3705	1212657	1064197.11	1208356	1129725
n3707	1186691	1035948.62	1185209	1102426
n3709	1205801	1054892.55	1205801	1146484
n370b	1220708	1074178.91	1212433	1166799
nag	945	465	930	550.54054054
nb10tb	$1.3796452423 \times 10^{10}$	$1.33301713 \times 10^{10}$	$1.3796451833 \times 10^{10}$	$1.3794494287 \times 10^{10}$
neos-1420546	22948.103198	22223.7073	22936.284978	22629.329177
neos-1420790	3119.8050577	3003.05309	3119.8060016	3067.0672911
neos-1423785	21893.26	20327.177	21893.333343	20707.764318
neos-2629914-sudost	48180	42387.0653	48180	48180
neos-2974461-ibar	468906174.77	429197921	468906174.77	443920829.19
neos-2978205-isar	−11.92580869	−11.9872276	−11.925808687	−11.925808687
neos-2991472-kalu	12	0	12	12
neos-3009394-lami	5.5	1.92857143	5.5	5.5
neos-3208254-reiu	−34897.000525	−38606.2553	−34902	−34902
neos-3355120-tarago	−11115971.494	−11712060.9	−11110906.689	−11123832.677

Continued on next page

Table 5 (continued)

Instance	Best-known primal MIPLIB v36	COPT dual 10-hour baseline	LLM4MIP primal	LLM4MIP dual
neos-3355323-arnon	—	0	—	—
neos-3594536-henty	401223	397821.689	401092	401092
neos-3603137-hoteo	—	0	—	—
neos-3615091-sutlej	−1048730.9379	−1404604.89	−1048730.8	−1048730.8
neos-3634244-kauru	1398.0146705	239.044636	1398.0156859	—
neos-3682128-sandon	34666767.474	21531270.9	34666770	34666770
neos-3691541-lonja	−0.00744145058	−0.019851233	0.0918239856	—
neos-3695882-vesdre	518.35054802	−7522.4774	518.5	517.4
neos-3754224-navua	157909.06776	56251.4076	166345.74719	56490.877344
neos-4230265-orari	73755	21013.257	72572	31528
neos-4232544-orira	5538597.4	5070682.61	5547858.8	5099038.4143
neos-4264598-oueme	5997393.4777	5388920.94	6207553.6	5468401.5318
neos-4285819-pedja	88.499336306	63.3949254	88.499336306	62.376611387
neos-4290317-perth	3017215.8714	3017032.16	3017171.1908	3017171.1908
neos-4292145-piako	19229.998369	15702.1569	27660	27570.1
neos-4355351-swalm	33.45757454	33.45763	33.45763	33.45763
neos-4358725-tagus[b]	4206.1401	2378.09139	32225.700099	32225.700099
neos-4360552-sangro	−8	−12.0644768	−8	−8
neos-4531126-vouga	13216.633968	412238.281	525030.88462	524657.38188
neos-4533806-waima	4870	4249.09466	4870	3956
neos-4535459-waipa	26040609	0	23331046	15474160.819
neos-4545615-waita	6311	3753.09803	6311	6021
neos-4555749-wards	3759	3187.97075	3759	3759
neos-4562542-watut	5684	4706.80045	5684	5684
neos-4647032-veleka	27214.4801	27213.9842	27214.4801	27214.4801
neos-5041822-cockle	5724245.7045	5697141.9	5724245.654	5701917.8171
neos-5044663-wairoa	4654484.5553	4654134.39	4654500	4654300
neos-5045105-creuse	20.571247075	20.1974827	20.571410409	19.932229271
neos-5151569-mologa	686748326.49	686479073	686721918.74	686532411.38
neos-5189128-totara	6459876.1385	501198.807	2901877.8002	2494433.1727
neos-5221106-oparau	52.67	0	52.07	42.54
neos-5266653-tugela	65505.200575	40074.8618	65257.57399	—
neos-848198	51797	50021.1801	51797	49843
nj1	—	78.882279	382.0606208	—
nj2	—	47.1019816	382.0606208	—
nj3	—	26.6759647	382.0606208	—
ns1456591	988.14054407	459.574181	988.14128344	988.14128344
ns1631475	11100	1659	11100	11100
ns1679495	2642035.1987	2556399.7	2642035.1987	2626815.6662
ns1690781	−928.08534703	−1022.68453	−928.18882865	−932.72130546
ns1828997	8	0	8	—
ns1849932	632	546	624	548
ns1856153	34.1634618	0	34.016973126	32.697165488
ns1904248	37.017626111	0	35.262535127	23.6999997
ns1905797	—	12.38858	13.98258	12.38858
ns2122698	68749.999885	65899.0548	68750	62785.565316
ns2124243	77459.991691	75440.7297	77460	77460
ns930473	821466	1899.50067	821466	821466
nsr8k	17974887.998	17525904.2	17974887.998	17512125.696
nucorsav	—	29.0461	73.61064226	73.610642246
pb-gfrd-pnc	8844	8521.83408	8844	8566
pb-grow22	−449536	−449587.303	−449536	−449546
pizza27i	701882	652380.8	701882	701882

Continued on next page

Table 5 (continued)

Instance	Best-known primal MIPLIB v36	COPT dual 10-hour baseline	LLM4MIP primal	LLM4MIP dual
pizza78i	564039	508093	564039	564039
polygonpack3-15	−55055635.466	−114051303	−55657792.808	−58365098.476
polygonpack4-10	−53594508.708	−93676573.4	−53594508.708	−76521127.377
polygonpack4-15	−63613643.562	−114052544	—	—
polygonpack4-7	−51837758.962	−75324457.9	−51837707.95	—
polygonpack5-15	−55494687.648	−66029891.1	—	—
proteindesign121hz512p19	3382	3392[c]	3382	3382
pythago7825[e]	—	0	—	—
r4l4-02-tree-bounds-50	499132179	478217938	499061405	483755247.89
ramos3	186	146.482178	186	156
rfds-4-days	−1129915.8646	−1157062.2	−1129915.8646	−1146278.6542[d]
rmine11	−2508.404144	−2509.7041	−2508.404144	−2521.3532743
rmine13	−3495.3706624	−3496.99083	−3495.370661	−3504.6899591
rmine15	−5018.819991	−5026.70481	−5018.823005	−5042.0362334
rmine21	−10618.750838	−10674.8943	−10618.750727	−10679.232567
rmine25	−15541.669287	−15667.8428	−15541.669277	−15667.918863
rocII-10-11	−11.793992743	−11.8947324	−11.794324502	−11.849392208
rocII-8-11	−8.7398453503	−11.889881	−8.739845027	−11.82
rwth-timetable	772998	754720.911	772258	755608.65978
s82	−33.797057624	−33.8093843	−33.797057624	—
scpj4scip	128	108.947237	128	107
scpk4	318	287.057116	318	283
scpl4	259	214.35471	259	214
scpm1	540	415.555435	537	416
scpn2	485	350.409358	485	352
sct1	−187.52765084	−202.560756	−187.52765084	−199.81870652
sct5	−228.12919207	−228.154186	−228.12919207	−228.14788057
seqsolve1	4277	4268.52039	4277	4275
seqsolve2short4288	84	33.1143818	83.999999835	17
seqsolve3short4288excess384	−40015	−42615.0595	−40035	−41168
set3-09	176497.15	134902.999	—	—
set3-16	134040.40975	104243.075	134040.40975	101848.66004
shipsched	111919.83373	67029.7019	111919.83373	85777.682467
shs1014	22671.197427	23116.6794[c]	22671.197442	22555.877512
shs1042	11070.70158	11218.5138[c]	11070.701662	10905.739776
siena1	10341586.2	10214130.1	10449008.9	10219502.195
sing11	19108639.439	18776564.1	19108639.46	18853382.032
sing17	36161699.379	35599182.4	36161699.378	35556219.48
sing5	18778435.608	18605207.3	18778435.608	18462676.779
snp-04-052-052	881796490.75	843534432	881740229.2	880540652.11
snp-06-004-052	1869531794.3	1869529780	1869531794.2	1869530183.6
snp-10-004-052	5906642759.1	5906557160	5906642753.4	5906641320.2
snp-10-052-052	6059860690.3	6011665050	6059234264.5	6055563163.9
sorrell7	−198	−223.02692	−198	−210
splan1	−194377.63065	−201881.851	−194385.95334	−201513.1968
stockholm	120.99999502	93.5364881	122	106
supportcase22	110	0.88353308	110	6.4
supportcase23	−12160.659357	−12633.1586	−12160.659354	−12345.870234
supportcase30	—	0	—	—
supportcase31	−3720096.597	−3721177.76	−3720097.298	−3720097.298
supportcase35[e]	−164208.6431	—	—	—
supportcase38	32873.968346	32476.0327	32564.500163	32476.032696
supportcase39	−1085080.9069	−1085222.66	−1085083.8568	−1085325.2307

Continued on next page

Table 5 (continued)

Instance	Best-known primal MIPLIB v36	COPT dual 10-hour baseline	LLM4MIP primal	LLM4MIP dual
supportcase41	32.000768321	28	32.000768325	31.81114547
t1717	158260	137668.992	158260	137323.70815
t1722	108953	103989.69	108953	—
tokyometro	8263.1	6817.12644	8263.1	7117.5
tpl-tub-ws1617	121066	120987.401	121065	120885
uccase10	39133.58411	39113.5635	39133.58411	39122.060112
xmas10	−497	−516.336258	−497	−507
xmas10-2	−497	−515.926191	−497	−515
z26	−1198	−1273.9627	−1198	−1646
zeil	1081.3585	834.502247	1081.3585	833.83125413

Notes. [a]Optimal within the reported tolerance. [b]The optimum applies to the verified model, and the public baseline comparison is excluded because the records could not be reconciled. [c]The historical lower bound exceeds the accepted feasible objective. [d]The numerical dual bound carries a validation qualification and is excluded from improvement counts. [e]The study reports infeasibility for fhnw-binpack4-58 and pythago7825, and unboundedness for supportcase35.