

# CO-Bench: Benchmarking Language Model Agents in Algorithm Search for Combinatorial Optimization

Weiwei Sun\* Shengyu Feng\* Shanda Li Yiming Yang  
Carnegie Mellon University  
{weiweis, shengyuf, shandal, yiming}@cs.cmu.edu

## Abstract

Although LLM-based agents have attracted significant attention in domains such as software engineering and machine learning research, their role in advancing combinatorial optimization (CO) remains relatively underexplored. This gap underscores the need for a deeper understanding of their potential in tackling structured, constraint-intensive problems—a pursuit currently limited by the absence of comprehensive benchmarks for systematic investigation. To address this, we introduce CO-Bench, a benchmark suite featuring 36 real-world CO problems drawn from a broad range of domains and complexity levels. CO-Bench includes structured problem formulations and curated data to support rigorous investigation of LLM agents. We evaluate multiple agent frameworks against established human-designed algorithms, revealing key strengths and limitations of current approaches and identifying promising directions for future research. CO-Bench is publicly available at <https://github.com/sunnweiwei/CO-Bench>.

## 1 Introduction

Combinatorial Optimization (CO) is a foundational problem class in computer science and operations research, focused on finding optimal solutions in discrete, structured, and constraint-rich domains. It underpins a wide range of real-world applications, including logistics (Vogiatzis & Pardalos, 2013), production planning (Crama, 1997), and bioinformatics (Gusfield, 1997). Many CO problems are computationally intractable and classified as NP-hard, making exact solutions impractical at scale. As a result, developing effective algorithms often demands significant domain expertise and manual effort—posing a long-standing challenge in both academic research and industrial applications.

Recent advancements in Large Language Models (LLMs) (OpenAI, 2024b; DeepSeek-AI, 2025b) have positioned LLM-based agents as increasingly promising tools for a variety of predictive and decision-making tasks (Jimenez et al., 2023; Chan et al., 2024; Gottweis et al., 2025). In particular, there is growing interest in applying LLMs to CO problems. Early studies have primarily focused on solution correctness, evaluated on small-scale test instances (Ramamonjison et al., 2023; Yang et al., 2024; Xiao et al., 2024a), aimed at addressing the everyday needs of general users. More recent work has begun to explore autonomous LLM agents capable of conducting research and designing more efficient algorithms for complex scientific and industrial challenges. For example, FunSearch (Romera-Paredes et al., 2023) combines LLM prompting with evolutionary search to discover heuristics that outperform human-designed counterparts in the Cap Set and Bin Packing problems. Subsequent methods (Liu et al., 2024; Ye et al., 2024) further improve computational efficiency and broaden applicability to domains such as routing and scheduling.

Despite these advancements, most existing efforts focus on narrow components (e.g., priority functions) within established algorithms, across a limited set of tasks (typically 4–7 problems), and often rely on heavily handcrafted, problem-specific prompts and templates (Romera-Paredes et al., 2023; Ye et al., 2024). Furthermore, there remains a lack of

---

\*Equal contribution.

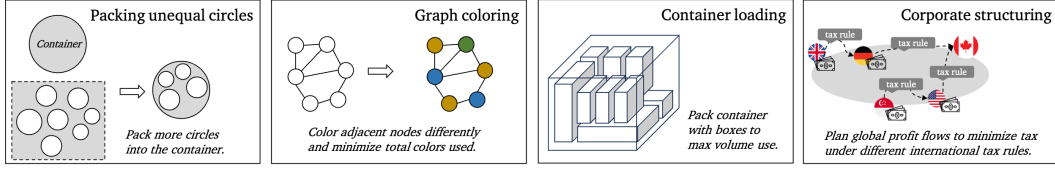


Figure 1: Four problem examples from CO-Bench. CO-Bench includes 36 problem types and aims to evaluate LLM agents’ ability to develop effective and efficient algorithms for solving real-world combinatorial optimization problems.

systematic evaluation of how these agents perform across a broader and more diverse collection of real-world CO problems.

To address this gap, we introduce **CO-Bench**, the first benchmark designed to evaluate LLM agents in the context of efficient CO algorithm development. CO-Bench comprises real-world CO problems spanning a wide range of domains and complexities. Figure 1 illustrates four example problems, while Table 1 compares CO-Bench with existing CO benchmarks. Compared to prior benchmarks, CO-Bench offers broader problem coverage and emphasizes end-to-end evaluation of LLM-based research agents—focusing on their ability to design efficient, potentially novel algorithms from *abstract problem descriptions*. This design enables reproducible and scalable evaluation of agent performance, including comparisons with human expert-designed baselines under equivalent time constraints. In doing so, CO-Bench introduces new challenges for LLM agent development, such as the discovery of algorithms that extend beyond current human knowledge of CO.

Using CO-Bench, we benchmark several LLMs and agent frameworks, comparing their performance against both expert-designed algorithms developed under similar time constraints and the best-known solutions reported in the literature. Our results show that reasoning-focused models (e.g., o3-mini and Claude-3.7-sonnet-thinking) consistently outperform standard base models. When integrated into agent frameworks like FunSearch, LLMs further improve through trial-and-error exploration. Notably, on 25 problems, AI-generated algorithms outperformed expert baselines within a 30-minute research budget, and on 3 problems, they surpassed the best-known solutions. However, our analysis also reveals current limitations, such as limited algorithmic novelty and insufficient handling of feasibility constraints. These findings highlight both the promise and challenges of LLM-driven research in CO and suggest key directions for advancing autonomous algorithm design.

In summary, this paper makes the following contributions:

- (i) We introduce CO-Bench, the first benchmark to evaluate the capability of LLMs to develop algorithms for diverse and challenging real-world CO problems
- (ii) We benchmark multiple LLMs and agent frameworks, analyzing their performance relative to expert-designed pipelines. Our results highlight the strengths of agent-generated algorithms, while also revealing limitations in planning, feasibility checking, and the generation of novel solutions.

## 2 Preliminary

### 2.1 Combinatorial Optimization

For each CO problem  $c$  (for example, Traveling salesman problem), we follow Papadimitriou & Steiglitz (1982) to formulate it as a constrained optimization problem in the discrete space. Consider a instance  $p$ , the optimization problem could be expressed as

$$\min_{x \in S_c(p)} f_c(x; p) + g_c(x; p), \quad (1)$$

where  $S_c(p)$  represents the solution space, e.g.,  $\mathbf{Z}^m \times \mathbb{R}^n$  for  $d$  discrete variables and  $c$  continuous variables,  $f_c(x; p)$  corresponds to the objective function, and  $g_c(x; p)$  stands for

Dataset	CO-Bench	NPHardEval	NL4OPT	OptiBench	ComplexOR	ReEvo
Algorithm Development	✓	✗	✗	✗	✗	✓
# of problem types	36	9	5	4	20*	7
# of test-set instances	6,482	900	289	605	100	597
# of variables (max)	11,000	24	3	18	9*	1,000

Table 1: Data statistics for CO-Bench and related CO benchmarks, including the indicator for algorithm development support, the number of problem types, the number of test-set problem instances, and the maximum number of test-set variables (e.g., the number of nodes in the largest graph). \* numbers are based on processed public data on Github.

the constraint violation, which is 0 for feasible solutions and  $+\infty$  otherwise. To avoid the clutter, we simply denote  $h_c(x; p) = f_c(x; p) + g_c(x; p)$  in the following text and omit  $c$  if the context is clear.

Given an algorithm set  $\mathcal{A}$  and a problem instance distribution  $D$ , the algorithm search problem could be defined as

$$\min_{A \in \mathcal{A}} \mathbb{E}_{p \sim D, x \sim A(p)} [h(x; p)]. \quad (2)$$

Different from previous neural CO solvers (Bengio et al., 2020) that directly parameterize  $A$  with a neural network, we focus on symbolic searching space where  $\mathcal{A}$  consists of all algorithms that could be represented by a Python Program, with a maximum number of  $d$  tokens, where  $d$  is typically decided by the output length limit of an LLM. Considering the popularity of randomized algorithms (Motwani & Raghavan, 2013) for CO, we treat the output of an algorithm  $A(p)$  as a distribution of solutions, while deterministic algorithms would correspond to the point distributions.

The main endeavor of this work is focused on the shaping of the algorithm set  $\mathcal{A}$ , the curation of the data distribution  $D$  and the definition of  $h$  on our collected CO problems.

## 2.2 LLM Agents

Given a CO problem  $c$ , a candidate algorithm could be generated by an LLM as

$$A \sim M(\text{textify}(c); \theta), \quad (3)$$

where  $M$  denotes an LLM with parameters  $\theta$ . However, one-time generation usually leads to inexecutable code or suboptimal algorithms (Madaan et al., 2023), and *agent frameworks* address this by enabling iterative refinement through structured interactions with external tools (e.g., a coding environment). Formally, an agent performs reasoning-action iterations (Yao et al., 2022):

$$r_{t+1} \sim M(\text{textify}_r(c, A_t, H_t); \theta), \quad a_{t+1} \sim M(\text{textify}_a(r_{t+1}, H_t); \theta), \quad (4)$$

where  $r_t$  is the reasoning step,  $a_t$  is the action step (e.g., executing code, evaluating results), and  $H_t = (r_i, a_i, \text{result}(a_i))_{i=1}^{t-1}$  maintains the interaction history. Thus, an *LLM agent* is formally defined as an LLM  $M(\cdot; \theta)$  guided by a structured workflow specifying iterative external interactions to enhance its outputs.

## 3 CO-Bench

We introduce CO-Bench, a comprehensive benchmark designed to evaluate the algorithm development ability of LLM agents on combinatorial optimization (CO) problems. The benchmark consists of 36 problems mainly sourced from OR-Library (Beasley, 1990), an established archive containing datasets accumulated by researchers across over 30 years of operations research. These problems span a wide range of realistic CO challenges in either academia research or industrial applications.

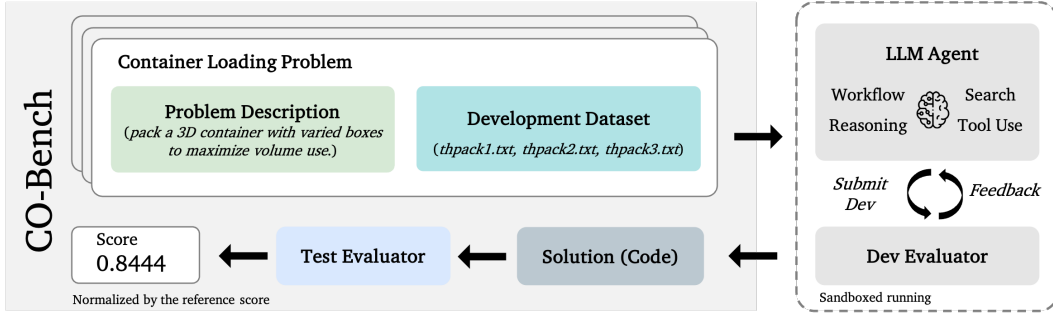


Figure 2: CO-bench is an evaluation environment for AI agents. Each problem has an associated description and a development dataset. Following the setup in Chan et al. (2024), the agent-generated code implements an algorithm design, which is further graded and compared against the best-known solution and human expert solution.

### 3.1 Data Curation

**Problem Selection** We first perform rigorous filtering and cleaning, and select 36 CO problems that cover diverse domains and complexities, including:

*Packing problems:* Bin packing (Falkenauer, 1996), Multi-Demand Multidimensional Knapsack problem (Cappanera & Trubian, 2001), Multidimensional knapsack problem (Petersen, 1967), Container loading (Bischoff & Ratcliff, 1995; Ivancic, 1988), Container loading with weight restrictions (Ratcliff & Bischoff, 1998; Bischoff, 2006), Packing unequal circles (López & Beasley, 2016), Packing unequal rectangles and squares number / area (López & Beasley, 2018).

*Cutting problems:* Assortment problem (Beasley, 1985c), Constrained / unconstrained guillotine cutting (Christofides & Whitlock, 1977; Beasley, 1985b), Constrained non-guillotine cutting (Beasley, 1985d; 2004).

*Facility location problems:* Capacitated / Uncapacitated warehouse location (Beasley, 1988; 1993), Capacitated / Uncapacitated p-median problem (Beasley, 1985a; Osman & Christofides, 1994).

*Scheduling problems:* Aircraft landing (Beasley et al., 2000; 2004), Crew scheduling (Beasley & Cao, 1996), Common due date scheduling (Biskup & Feldmann, 2001), Flow shop scheduling (Taillard, 1993), Hybrid Reentrant Shop Scheduling (Chakhlevitch & Glass, 2009), Job shop scheduling (Taillard, 1993), Open shop scheduling (Taillard, 1993).

*Routing problems:* Traveling salesman problem (Laporte, 1992), Period vehicle routing problem (Christofides & Beasley, 1984), Resource constrained shortest path (Beasley & Christofides, 1989).

*Assignment problems:* Constrained / unconstrained assignment (Osman, 1995; and, 1990).

*Tree problems:* Euclidean Steiner (Beasley, 1992), Corporate structuring (Anken & Beasley, 2012)

*Graph and set problems:* Maximal Independent Set (Erdos & Rényi, 1984), Graph colouring (Fleurent & Ferland, 1996), Equitable partitioning (Mingers & O’Brien, 1995), Set partitioning (Chu & Beasley, 1998), Set covering (Beasley & Jörnsten, 1992).

**Data Annotation** For each problem, we manually annotate the following components: (1) *Problem description*: a formal definition of the optimization problem in natural language, accompanied by a clearly specified solve function as the starter code; (2) *Data loading function*: a load\_data function to load and preprocess raw data from the test files; (3) *Evaluation function*: an eval\_func function that rigorously and robustly evaluates the quality of a solution. Additionally, each problem comprises a *development* set and a *test* set, each containing several problem instances.

**Evaluation Framework** We develop a rigorous and efficient evaluation framework to assess the performance of LLM agents in simulated, time-constrained competition scenarios (Chan et al., 2024). Specifically, LLM agents operate within a sandbox environment with access to a Linux machine. For each problem, agents are provided with a problem description, development datasets, and an API endpoint for submitting their solutions (i.e. codebases) to receive evaluation feedback. An independent evaluation system, which is

protected by built-in safeguards, scores the submitted solutions on the development set in parallel. After a limited number of research steps, the agent returns the final solution for evaluation on the test set. During the agent development process, both `eval_func` and test data are invisible. Figure 2 shows the evaluation pipeline in CO-Bench.

### 3.2 Benchmarking Human Performance

To investigate how existing LLM agents performs compared to humans when within the same research time budget, we follow Wijk et al. (2024) to establish *human expert* performance as a human baseline. Specifically, the authors of this paper, who have numerous publications in related areas and are familiar with the problems in CO-Bench, spent a similar amount of time as the agents to develop the algorithms. To simulate realistic scenarios, human experts were allowed to utilize any available tools (e.g., search engines, any AI, and access to the development set), as well as any software packages or techniques in crafting their solutions. The human expert worked on each problem for up to 30 minutes, and we also record their intermediate solutions for further analysis.

### 3.3 Evaluation Metrics

**Avg Score** The main evaluation metric is similar to the *Primal Gap* (Berthold, 2006), defined as the normalized score of the primal bound  $h(x; p)$  against a pre-computed optimal (or best-known) objective value  $h_p^*$ :

$$s(x, p) = \frac{\min\{|h(x, p)|, |h_p^*|\}}{\max\{|h(x, p)|, |h_p^*|\}}, \quad (5)$$

A higher value indicates better performance and a score of 1 signifies the performance identical to the optimal or best-known solution. Program errors or infeasible solutions lead to a score of 0.0. The score of a solver on a given problem is computed by averaging its scores across all test instances. The overall benchmark score is then obtained by averaging these problem-level scores across all 36 problems.

**Valid Solution** We compute the percentage of problems for which the generated code is correct on all test instances. Any raised error—such as constraint violation or timeout—is treated as an invalid signal. If any test instance for a given problem results in an invalid signal, the entire solution for that problem is considered invalid, even if it produces valid results on other test instances.

**Bradley-Terry Score (BT Score)** While absolute scores provide a straightforward measure of model performance, they may not fully capture the relative competitiveness among different methods. To complement this, we compute the BT Score (Bradley & Terry, 1952), a metric that evaluates models based on head-to-head comparison outcomes and has been widely adopted in recent LLM benchmarking efforts (Chiang et al., 2024). In the Bradley-Terry framework, each model  $i$  is assigned a latent ability parameter  $\theta_i$ . The probability that model  $i$  outperforms model  $j$  is defined as:

$$P(i \succ j) = \frac{\theta_i}{\theta_i + \theta_j}. \quad (6)$$

The  $\theta_i$  parameters are estimated iteratively using the outcomes of all pairwise comparisons across evaluation instances. The resulting values offer an interpretable measure of each model’s relative competitive strength. Additional implementation details can be found in the Appendix A.

**Above Human** Given the human expert performance in Section 3.2, we calculate the portion of problems where the model outperforms the human-expert baseline within the 30-minute time budget.



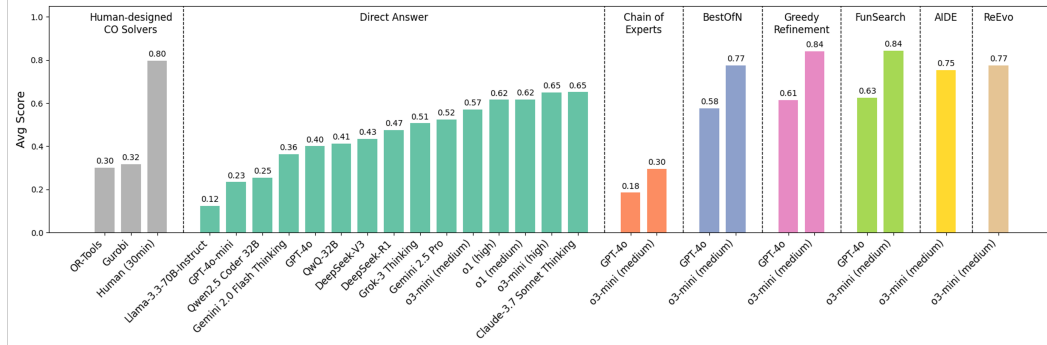


Figure 3: Avg Score of LLMs and Agents on CO-Bench.

**Survival Rate** The survival rate measure that, for each problem, the percentage of test instances where the model’s solution is above 99% of the reference score (reported optimal or best-known solution score from previous literature). This serve as a challenge metric as the model can only get credit when it is very close or better than previous-best algorithm.

## 4 Experimental Setup

### 4.1 Benchmarked Methods

On CO-Bench, we evaluate various LLMs combined with different agent frameworks, and compare them with existing human-designed CO solvers.

**LLMs** We conduct experiments on 5 open-source models and 10 proprietary models. These include instruction-tuned models such as *Llama-3.3-70B-Instruct* (Meta, 2024), *Qwen-2.5-Code-32B-Instruct* (Hui et al., 2024), *DeepSeek-V3* (DeepSeek-AI, 2024), and *GPT-4o* (OpenAI, 2024a), as well as frontier reasoning models, including *o3-mini* (OpenAI), *Claude-3.7-Sonnet-Thinking* (Anthropic, 2025), *DeepSeek-R1* (DeepSeek-AI, 2025a), *Grok-3-Thinking* (xAI, 2025), *QwQ-32B* (Qwen, 2025), *Gemini 2.0 Flash Thinking* and *Gemini 2.5 Pro* (DeepMind, 2025).

**Agent Frameworks** For the aforementioned LLMs, we apply various agent frameworks to evaluate their performance across different strategies. These range from simple approaches, such as direct generation, to more sophisticated frameworks that augment LLM with additional tools, workflows, and test-time compute:

- *Direct Answer*: The simplest approach, where the LLM directly generates a solution to the combinatorial optimization problem without further refinement.
- *BestOfN Sampling* (Chen et al., 2021): Generate  $N$  candidate solutions, evaluate each on a development set, and select the solution with the best performance.
- *Chain of Experts* (Xiao et al., 2024a): A multi-agent prompting framework where agents of different roles cooperate to debug and deliver one solution.
- *Greedy Refinement* (Shinn et al., 2023; Madaan et al., 2023): Iteratively prompt the LLM to refine the current best solution based on the evaluation results of the development set, repeating this refinement process for  $N$  steps.
- *FunSearch* (Romera-Paredes et al., 2023): Prompt the LLM to either draft a new solution or refine an existing one, followed by employing an evolutionary algorithm to iteratively select and improve candidate solutions.
- *AIDE* (Jiang et al., 2025): A representative method for machine learning engineering tasks, which stores existing solutions in a tree structure and selectively prompts the LLM to draft new solutions, debug or improve previously stored solutions.
- *ReEvo* (Ye et al., 2024): A recent evolutionary algorithm that incorporates short-term and long-term reflection modules, as well as a multi-agent framework.

Method		Avg Score	Valid Solution	BT Score	Above Human	Survival Rate
<b>Human-designed CO Solvers</b>						
OR-Tools		0.3005	0.0833	0.2745	0.0833	0.1941
Gurobi		0.3161	0.2500	0.2848	0.0555	0.1902
Human Expert (30min)		0.8025	0.6111	4.2078	-	0.3939
Agent	LLM					
Direct Answer	Llama-3.3-70B-Instruct	0.1230	0.0555	0.1413	0.0277	0.0573
	GPT-4o-mini	0.2338	0.1111	0.1899	0.0555	0.0948
	Qwen2.5 Coder-32B	0.2535	0.1111	0.1869	0.0555	0.1357
	Gemini 2.0 Flash Thinking	0.3358	0.1388	0.3373	0.0555	0.1159
	GPT-4o	0.3999	0.1666	0.3255	0.1111	0.1376
	Qwen QwQ-32B	0.4116	0.2500	0.4546	0.1111	0.1590
	DeepSeek-V3	0.4348	0.1111	0.4291	0.0833	0.1824
	DeepSeek-R1	0.4746	0.3055	0.5984	0.0833	0.1731
	Grok-3 Thinking	0.5066	0.2222	0.5727	0.1388	0.1980
	Gemini 2.5 Pro	0.5249	0.2500	0.7039	0.2222	0.2302
	o3-mini (medium)	0.5709	0.3611	0.6985	0.1666	0.1896
	o1 (high)	0.6151	0.4166	1.0416	0.1666	0.2205
	o1 (medium)	0.6169	0.3611	1.0871	0.2777	0.2430
	o3-mini (high)	0.6495	0.3888	1.4453	0.3055	0.2512
	Claude-3.7 Sonnet Thinking	0.6500	0.3611	0.9224	0.1944	0.2246
Chain of Experts	GPT-4o	0.2761	0.1944	0.1585	0.0555	0.1499
	o3-mini (medium)	0.3821	0.1388	0.2714	0.0833	0.1879
BestOfN	GPT-4o	0.5760	0.2777	0.8845	0.1666	0.2153
	o3-mini (medium)	0.7743	0.4722	2.8245	0.4166	0.2846
Greedy Refinement	GPT-4o	0.6141	0.3055	1.0841	0.2777	0.2087
	o3-mini (medium)	0.8401	<b>0.5555</b>	<b>5.7970</b>	<b>0.6944</b>	<b>0.4482</b>
FunSearch	GPT-4o	0.6257	0.3055	0.9972	0.2777	0.1762
	o3-mini (medium)	<b>0.8423</b>	0.5000	5.5632	0.6388	0.4258
AIDE	o3-mini (medium)	0.7534	0.5277	1.9096	0.2777	0.2266
ReEvo	o3-mini (medium)	0.7741	0.5555	3.4393	0.4444	0.3203

Table 2: **Overall Performance.** *Avg Score* refers to the average normalized objective scores across all problems. *Valid Solution* indicates the percentage of test-set problems for which the solutions are feasible. *BT score* refers to Bradley-Terry Score, which is mentioned in Equation 6. *Above Human* represents the percentage of test instances where the model outperforms the human expert within the 30-minute time budget. *Survival Rate* measures the percentage of test instances where the model’s score exceeds 99% of the reference score. The boldface in each column indicate the best result under the corresponding metric.

**Human-designed CO Solvers** We evaluate two general CO solvers: *Gurobi* (<https://www.gurobi.com/>) and *OR-Tools* (<https://github.com/google/or-tools>). To formulate the problems in the required format, we first use *o3-mini-high* to draft the initial code, followed by manual revisions for correctness and completeness.

## 4.2 Implementation Details

For benchmark evaluation, we limit the solving time of each test instance to 10 seconds on a single CPU, such that the exact solving of the problem (achieving the optimal solution) is impossible on most test instances. Test instances that result in a timeout or error receive a score of 0. For agent implementation, we adapt their approaches to our benchmark setting (i.e., end-to-end algorithm search), and set the maximum steps as 64, where each step corresponds to evaluating one candidate solution on the development set. We then select the best solution on the development set for the final benchmark evaluation.

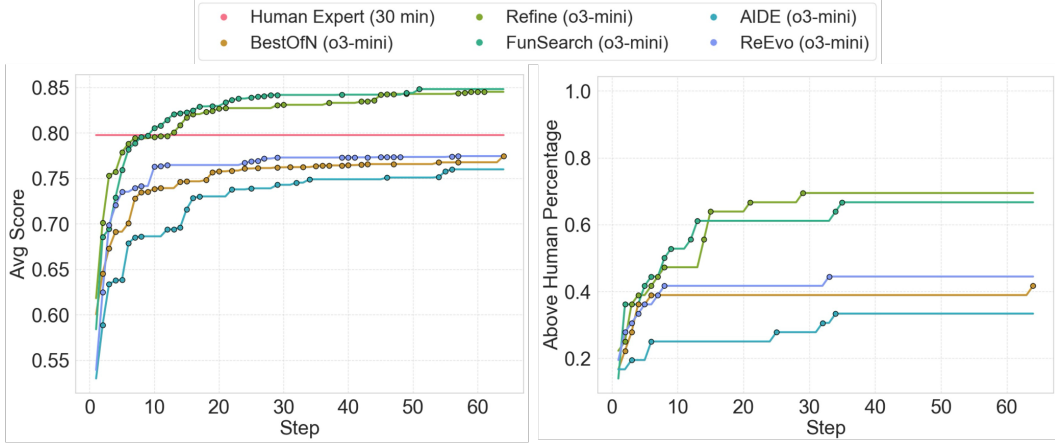


Figure 4: *Avg Score* and *Above Human Percentage* (within a 30-minute budget) of different agents vs. research steps (a maximum of 64 steps).

## 5 Experimental Results

### 5.1 Main Results

Figure 3 illustrates the avg score of LLMs and agents, and Table 2 presents the detailed results on the test set. We have the following findings: (i) For one-shot generation, reasoning models usually achieve better performance than non-reasoning ones. The best LLMs for the one-shot generation are *o3-mini (high)* and *Claude-3.7 Sonnet Thinking*, both achieving a score around 0.65. (ii) Agentic systems significantly enhance the capabilities of LLMs. Among the agent frameworks evaluated, FunSearch demonstrates the best average score, achieving a score of 0.8444 when based on *o3-mini (medium)*. In terms of *BT Score*, *Above Human*, and *Survival Rate*, Greedy Refinement achieves the best results, outperforming human experts in 25 tasks and reaching within 99% of the reference score on 44.82% of test instances. These results demonstrate the effectiveness of LLM-based agents on algorithm development for CO.

However, our evaluation also reveals several limitations of current agents: (i) Some advanced agent frameworks, such as AIDE, underperform compared to BestOfN—a much simpler strategy that increases inference-time computation for LLMs. This suggests that the planning capabilities of these agents are still preliminary and often fail to outperform random sampling. (ii) From the *Valid Solution* metric, existing LLMs achieve correctness (around 30%), and even the best-performing agents fall far short of human expert performance. This indicates that current agents often struggle to ensure the feasibility of solutions.

### 5.2 Performance over Research Steps

Figure 4 illustrates the performance of different agents in various research steps. Here, one research step is defined as one submission to the evaluation system, with an observed evaluation results. We also plot the performance of the human expert under comparable time budget. We can see that all agents show the ability to improve performance with more research steps. FunSearch overall achieves the best performance, achieving a score of 0.8423 and converging after 50 steps. We also observe that, reasoning models like *o3-mini* exhibit superior inference-scaling performance compared to non-reasoning models like GPT-4o.

### 5.3 Comparison to Neural Solvers

Table 3 compares the performance of agents with representative neural solvers on TSP and MIS, two well-studied CO problems. We include DIMES (Qiu et al., 2022), DIFUSCO (Sun & Yang, 2023), and T2T (Li et al., 2023) as neural baselines. For the method with multiple



	TSP-500		TSP-1000		TSP-10000		ER-Small		ER-Large	
	Len ↓	Time ↓	Len ↓	Time ↓	Len ↓	Time ↓	Size ↑	Time ↓	Size ↑	Time ↓
Gurobi	16.55	45.6h	-	-	-	-	41.38	50.0m	-	-
DIMES	18.84	1.1m	26.36	2.4m	85.75	4.8m	42.06	12.0m	332.80	12.5m
DIFUSCO	16.65	11.5m	23.45	48.1m	73.89	6.72h	41.12	26.6m	-	-
T2T	16.61	16.0m	23.30	54.6m	-	-	41.37	29.7m	-	-
LEHD + ReEvo	16.78	-	23.82	-	-	-	-	-	-	-
Greedy Refine (o3-mini)	17.37	19.1m	24.40	19.1m	77.65	2.5m	42.35	20.1m	354.00	2.5m
FunSearch (o3-mini)	17.20	19.1m	25.31	19.1m	80.18	2.5m	41.65	1.9m	356.50	2.1m

Table 3: Objective values and solving time of different solvers on TSP and MIS, with varying data sizes. Results of baselines are taken from the published results in corresponding papers.

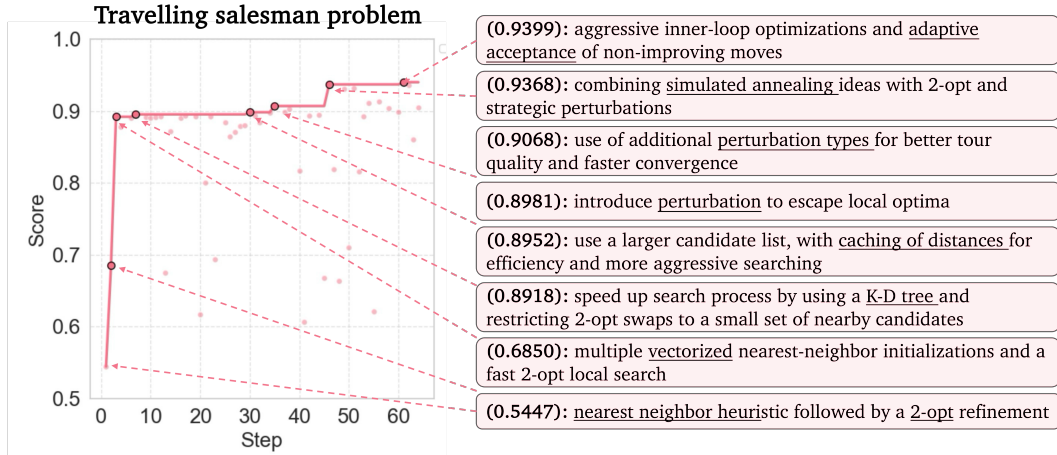


Figure 5: Trajectory of algorithm development for Greedy Refinement (o3-mini) on the traveling salesman problem (TSP) over 64 research steps. Each dot represents the evaluation score at a given step. The curve and highlighted dots indicate the best-ever score and the steps where improvements occurred. The algorithmic ideas behind each improvement step are summarized in the corresponding boxes.

variants, we only include their best results on each dataset. We also consider a hybrid method, LEHD + ReEvo (Ye et al., 2024), which combines the neural solver with LLM-designed heuristics. We report both the objective values (the tour length for TSP and set size for MIS) and the solving time. The results show that the agents such as Greedy Refine and FunSearch achieve competitive performance on both problems, often outperforming existing neural solvers under similar time budget and approaching the best results achieved by previous solvers given extended search time.

#### 5.4 Case Study

Figure 5 illustrates an example trajectory of algorithm development for Greedy Refinement (o3-mini) on TSP across multiple research steps. In the early stage, the agent improves the code efficiency by adopting vectorized data structure and utilizing a K-D tree. It then increases the number of search iterations and introduces perturbations to help escape local optima. Finally, the agent integrates simulated annealing to balance the exploration and exploitation, and applies adaptive heuristics for different instance sizes. This example demonstrates that LLMs excel at applying established techniques to enhance efficiency and optimize implementation, despite the lack of algorithmic novelty.

## 6 Related Work

### 6.1 Automatic Algorithm Search for Combinatorial Optimization

Automating algorithm search for combinatorial optimization (CO) has emerged as a significant research direction in the machine learning community. Traditional machine learning solvers primarily parameterize CO algorithms as trainable neural networks (Bengio et al., 2020; Cappart et al., 2023). Although effective in capturing data distributions, these neural approaches often struggle to generate feasible solutions, necessitating integration with human-designed heuristics such as branch-and-bound (Gasse et al., 2019) and tree search (Böther et al., 2022). To address this limitation, Kuang et al. (2024a;b) propose to decompose CO algorithms into symbolic operators and conduct searches in the symbolic space. However, designing these unit symbolic operators demands substantial human expertise, limiting generalizability and comprehensive coverage of all algorithm types. Recent advances in Large Language Models (LLMs) and LLM-based agents have significantly mitigated this challenge by enabling symbolic searching in programming language formats (Romera-Paredes et al., 2023; Ye et al., 2024; Liu et al., 2024). Building on these developments, CO-Bench aims to extend the success of these methods to more real-world CO problems and facilitate further research in this domain.

### 6.2 CO Benchmark for LLMs

Existing CO benchmarks can be roughly classified into two categories. The first type formulates CO problems as question-answering tasks (Fan et al., 2024; Tang et al., 2025). Although LLMs have the potential to solve CO problems via natural language reasoning, their excessive parameter size makes them inefficient CO solvers in general. Therefore, the second type of benchmarks evaluates the tool-using ability of LLMs, e.g., calling an existing CO solver, to address CO problems (Xiao et al., 2024b; Ahmaditeshnizi et al., 2024; Yang et al., 2025). However, these benchmarks only evaluate the correctness of the generated algorithm on small-scale CO problems, whose problem parameters could be fully expressed in natural language. In contrast, CO-Bench targets scientific and industrial challenges, emphasizing the evaluation of algorithm efficiency on diverse, large-scale CO instances. This results in a more demanding benchmark, well-suited for assessing powerful reasoning models and research agents.

## 7 Conclusion

This work introduces CO-Bench, the first benchmark designed to evaluate the ability of LLMs in the search of combinatorial optimization (CO) algorithms. Our systematic evaluation reveals that reasoning-focused LLMs, especially when paired with agent frameworks, can automatically discover effective algorithms that rival or surpass the ones designed by human experts, with competitive searching time. However, we also identify key limitations of current LLM agents: they struggle to understand the problem constraints, and rely heavily on trial-and-error rather than innovative thinking. These shortcomings highlight the need for future research to enhance agents’ problem comprehension and creative reasoning abilities in CO tasks, enabling more robust and autonomous scientific discovery.

### Broader Impact & Ethics Statement

CO-Bench aims to advance the study of LLM-based agents in combinatorial optimization by providing a diverse and rigorous benchmark. By evaluating LLM agents on real-world CO problems, this work contributes to understanding their potential to automate and accelerate algorithm design in domains such as logistics, manufacturing, and scientific computing.

We acknowledge that the use of LLM agents in decision-making systems—especially those deployed in high-stakes contexts—requires careful validation to ensure safety, fairness, and accountability. Our benchmark does not currently evaluate social or ethical impacts

of discovered algorithms (e.g., fairness or resource allocation bias), which is an important area for future extension. Additionally, as LLMs may replicate biases from training data or generate unverified solutions, we caution against overreliance on these models without human oversight. By releasing CO-Bench and our findings, we aim to foster transparent, reproducible research and encourage the development of responsible, trustworthy LLM agents for optimization tasks.

## References

- Ali Ahmaditeshnizi, Wenzhi Gao, and Madeleine Udell. OptiMUS: Scalable optimization modeling with (MI)LP solvers and large language models. In Ruslan Salakhutdinov, Zico Kolter, Katherine Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp (eds.), *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 577–596. PMLR, 21–27 Jul 2024.
- J. E. Beasley and. Linear programming on cray supercomputers. *Journal of the Operational Research Society*, 41(2):133–139, 1990. doi: 10.1057/jors.1990.21. URL <https://doi.org/10.1057/jors.1990.21>.
- F. Anken and John E. Beasley. Corporate structure optimisation for multinational companies. *Omega-international Journal of Management Science*, 40:230–243, 2012.
- Anthropic. Claude sonnet. <https://www.anthropic.com/claude/sonnet>, 2025. Accessed: 2025-03-24.
- John E. Beasley. A note on solving large p-median problems. *European Journal of Operational Research*, 21:270–273, 1985a.
- John E. Beasley. Algorithms for unconstrained two-dimensional guillotine cutting. *Journal of the Operational Research Society*, 36:297–306, 1985b.
- John E. Beasley. An algorithm for the two-dimensional assortment problem. *European Journal of Operational Research*, 19:253–261, 1985c.
- John E. Beasley. An exact two-dimensional non-guillotine cutting tree search procedure. *Oper. Res.*, 33:49–64, 1985d.
- John E. Beasley. An algorithm for solving large capacitated warehouse location problems. *European Journal of Operational Research*, 33:314–325, 1988.
- John E. Beasley. Or-library: Distributing test problems by electronic mail. *Journal of the Operational Research Society*, 41:1069–1072, 1990.
- John E. Beasley. A heuristic for euclidean and rectilinear steiner problems. *European Journal of Operational Research*, 58:284–292, 1992.
- John E. Beasley. Lagrangean heuristics for location problems. *European Journal of Operational Research*, 65:383–399, 1993.
- John E. Beasley. A population heuristic for constrained two-dimensional non-guillotine cutting. *Eur. J. Oper. Res.*, 156:601–627, 2004.
- John E. Beasley and B. Cao. A tree search algorithm for the crew scheduling problem. *European Journal of Operational Research*, 94:517–526, 1996.
- John E. Beasley and Nicos Christofides. An algorithm for the resource constrained shortest path problem. *Networks*, 19:379–394, 1989.
- John E. Beasley and Kurt Jörnsten. Enhancing an algorithm for set covering problems. *European Journal of Operational Research*, 58:293–300, 1992.
- John E. Beasley, Mohan Krishnamoorthy, Yazid M. Sharaiha, and David Abramson. Scheduling aircraft landings - the static case. *Transp. Sci.*, 34:180–197, 2000.

- John E. Beasley, Mohan Krishnamoorthy, Yazid M. Sharaiha, and David Abramson. Displacement problem and dynamically scheduling aircraft landings. *Journal of the Operational Research Society*, 55:54–64, 2004.
- Yoshua Bengio, Andrea Lodi, and Antoine Prouvost. Machine learning for combinatorial optimization: a methodological tour d’horizon, 2020.
- Timo Berthold. Primal heuristics for mixed integer programs. 2006.
- Eberhard E. Bischoff. Three-dimensional packing of items with limited load bearing strength. *Eur. J. Oper. Res.*, 168:952–966, 2006.
- Eberhard E. Bischoff and M. S. W. Ratcliff. Issues in the development of approaches to container loading. *Omega-international Journal of Management Science*, 23:377–390, 1995.
- Dirk Biskup and Martin Feldmann. Benchmarks for scheduling on a single machine against restrictive and unrestrictive common due dates. *Comput. Oper. Res.*, 28:787–801, 2001.
- Maximilian Böther, Otto Kißig, Martin Taraz, Sarel Cohen, Karen Seidel, and Tobias Friedrich. What’s wrong with deep learning in tree search for combinatorial optimization. In *International Conference on Learning Representations*, 2022.
- Ralph Allan Bradley and Milton E. Terry. Rank analysis of incomplete block designs: I. the method of paired comparisons. *Biometrika*, 39:324, 1952.
- Paola Cappanera and Marco Trubian. A local-search-based heuristic for the demand-constrained multidimensional knapsack problem. *INFORMS J. Comput.*, 17:82–98, 2001.
- Quentin Cappart, Didier ChÃ©telat, Elias B. Khalil, Andrea Lodi, Christopher Morris, and Petar Velićković. Combinatorial optimization and reasoning with graph neural networks. *Journal of Machine Learning Research*, 24(130):1–61, 2023.
- Konstantin Chakhlevitch and Celia A. Glass. Scheduling reentrant jobs on parallel machines with a remote server. *Comput. Oper. Res.*, 36:2580–2589, 2009.
- Jun Shern Chan, Neil Chowdhury, Oliver Jaffe, James Aung, Dane Sherburn, Evan Mays, Giulio Starace, Kevin Liu, Leon Maksin, Tejal A. Patwardhan, Lilian Weng, and Aleksander Mkadry. Mle-bench: Evaluating machine learning agents on machine learning engineering. *ArXiv*, abs/2410.07095, 2024.
- Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Pondé, Jared Kaplan, Harrison Edwards, Yura Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, Nick Ryder, Mikhail Pavlov, Alethea Power, Lukasz Kaiser, Mo Bavarian, Clemens Winter, Philippe Tillet, Felipe Petroski Such, David W. Cummings, Matthias Plappert, Fotios Chantzis, Elizabeth Barnes, Ariel Herbert-Voss, William H. Guss, Alex Nichol, Igor Babuschkin, Suchir Balaji, Shantanu Jain, Andrew Carr, Jan Leike, Joshua Achiam, Vedant Misra, Evan Morikawa, Alec Radford, Matthew M. Knight, Miles Brundage, Mira Murati, Katie Mayer, Peter Welinder, Bob McGrew, Dario Amodei, Sam McCandlish, Ilya Sutskever, and Wojciech Zaremba. Evaluating large language models trained on code. *ArXiv*, abs/2107.03374, 2021.
- Wei-Lin Chiang, Lianmin Zheng, Ying Sheng, Anastasios Nikolas Angelopoulos, Tianle Li, Dacheng Li, Hao Zhang, Banghua Zhu, Michael Jordan, Joseph Gonzalez, and Ion Stoica. Chatbot arena: An open platform for evaluating llms by human preference. *ArXiv*, abs/2403.04132, 2024.
- Nicos Christofides and John E. Beasley. The period routing problem. *Networks*, 14:237–256, 1984.
- Nicos Christofides and Charles Whitlock. An algorithm for two-dimensional cutting problems. *Oper. Res.*, 25:30–44, 1977.

- P. C. Chu and John E. Beasley. Constraint handling in genetic algorithms: The set partitioning problem. *Journal of Heuristics*, 4:323–357, 1998.
- Yves Crama. Combinatorial optimization models for production scheduling in automated manufacturing systems. *European Journal of Operational Research*, 99(1):136–153, 1997. ISSN 0377-2217. doi: [https://doi.org/10.1016/S0377-2217\(96\)00388-8](https://doi.org/10.1016/S0377-2217(96)00388-8).
- Google DeepMind. Flash thinking: Behind the scenes of gemini. <https://deepmind.google/technologies/gemini/flash-thinking/>, 2025. Accessed: 2025-03-24.
- DeepSeek-AI. Deepseek-v3 technical report. *ArXiv*, abs/2412.19437, 2024.
- DeepSeek-AI. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *ArXiv*, abs/2501.12948, 2025a.
- DeepSeek-AI. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning, 2025b.
- Paul L. Erdos and Alfréd Rényi. On the evolution of random graphs. *Transactions of the American Mathematical Society*, 286:257–257, 1984.
- Emanuel Falkenauer. A hybrid grouping genetic algorithm for bin packing. *Journal of Heuristics*, 2:5–30, 1996.
- Lizhou Fan, Wenyue Hua, Lingyao Li, Haoyang Ling, and Yongfeng Zhang. NPHardEval: Dynamic benchmark on reasoning ability of large language models via complexity classes. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 4092–4114, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.225.
- Charles Fleurent and Jacques A. Ferland. Genetic and hybrid algorithms for graph coloring. *Annals of Operations Research*, 63:437–461, 1996.
- Maxime Gasse, Didier Chételat, Nicola Ferroni, Laurent Charlin, and Andrea Lodi. Exact combinatorial optimization with graph convolutional neural networks. In *Advances in Neural Information Processing Systems 32*, 2019.
- Juraj Gottweis, Wei-Hung Weng, Alexander Daryin, Tao Tu, Anil Palepu, Petar Sirkovic, Artiom Myaskovsky, Felix Weissenberger, Keran Rong, Ryutaro Tanno, Khaled Saab, Dan Popovici, Jacob Blum, Fan Zhang, Katherine Chou, Avinatan Hassidim, Burak Gokturk, Amin Vahdat, Pushmeet Kohli, Yossi Matias, Andrew Carroll, Kavita Kulkarni, Nenad Tomaev, Yuan Guan, Vikram Dhillon, Eeshit Dhaval Vaishnav, Byron Lee, Tiago R D Costa, Jos’e R Penad’es, Gary Peltz, Yunhan Xu, Annalisa Pawlosky, Alan Karthikesalingam, and Vivek Natarajan. Towards an ai co-scientist. *ArXiv*, abs/2502.18864, 2025.
- Dan Gusfield. Algorithms on stings, trees, and sequences: Computer science and computational biology. *Acm Sigact News*, 28(4):41–60, 1997.
- Binyuan Hui, Jian Yang, Zeyu Cui, Jiayi Yang, Dayiheng Liu, Lei Zhang, Tianyu Liu, Jiajun Zhang, Bowen Yu, Kai Dang, An Yang, Rui Men, Fei Huang, Shanghaoran Quan, Xingzhang Ren, Xuancheng Ren, Jingren Zhou, and Junyang Lin. Qwen2.5-coder technical report. *ArXiv*, abs/2409.12186, 2024.
- Nancy J. Ivancic. An integer programming based heuristic approach to the three dimensional packing problem. 1988.
- Zhengyao Jiang, Dominik Schmidt, Dhruv Srikanth, Dixing Xu, Ian Kaplan, Deniss Jacenko, and Yuxiang Wu. Aide: Ai-driven exploration in the space of code. *ArXiv*, abs/2502.13138, 2025.
- Carlos E. Jimenez, John Yang, Alexander Wettig, Shunyu Yao, Kexin Pei, Ofir Press, and Karthik Narasimhan. Swe-bench: Can language models resolve real-world github issues? *ArXiv*, abs/2310.06770, 2023.



- Yufei Kuang, Jie Wang, Haoyang Liu, Fangzhou Zhu, Xijun Li, Jia Zeng, Jianye HAO, Bin Li, and Feng Wu. Rethinking branching on exact combinatorial optimization solver: The first deep symbolic discovery framework. In *The Twelfth International Conference on Learning Representations*, 2024a.
- Yufei Kuang, Jie Wang, Yuyan Zhou, Xijun Li, Fangzhou Zhu, Jianye Hao, and Feng Wu. Towards general algorithm discovery for combinatorial optimization: Learning symbolic branching policy from bipartite graph. In Ruslan Salakhutdinov, Zico Kolter, Katherine Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp (eds.), *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 25623–25641. PMLR, 21–27 Jul 2024b.
- Gilbert Laporte. The traveling salesman problem: An overview of exact and approximate algorithms. *European Journal of Operational Research*, 59(2):231–247, 1992. ISSN 0377-2217. doi: [https://doi.org/10.1016/0377-2217\(92\)90138-Y](https://doi.org/10.1016/0377-2217(92)90138-Y).
- Yang Li, Jinpei Guo, Runzhong Wang, and Junchi Yan. From distribution learning in training to gradient search in testing for combinatorial optimization. In *Neural Information Processing Systems*, 2023.
- Fei Liu, Xialiang Tong, Mingxuan Yuan, Xi Lin, Fu Luo, Zhenkun Wang, Zhichao Lu, and Qingfu Zhang. Evolution of heuristics: Towards efficient automatic algorithm design using large language model. In *ICML*, 2024.
- Claudia O. López and John E. Beasley. A formulation space search heuristic for packing unequal circles in a fixed size circular container. *Eur. J. Oper. Res.*, 251:64–73, 2016.
- Claudia O. López and John E. Beasley. Packing unequal rectangles and squares in a fixed size circular container using formulation space search. *Comput. Oper. Res.*, 94:106–117, 2018.
- Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegraffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, Sean Welleck, Bodhisattwa Prasad Majumder, Shashank Gupta, Amir Yazdanbakhsh, and Peter Clark. Self-refine: Iterative refinement with self-feedback. *ArXiv*, abs/2303.17651, 2023.
- Meta. The llama 3 herd of models. *ArXiv*, abs/2407.21783, 2024.
- John C. Mingers and Frances A. O’Brien. Creating student groups with similar characteristics: A heuristic approach. *Omega-international Journal of Management Science*, 23:313–321, 1995.
- Rajeev Motwani and Prabhakar Raghavan. *Randomized Algorithms*. Cambridge University Press, USA, 2013. ISBN 0511814070.
- OpenAI. Openai o3-mini system card.
- OpenAI. Gpt-4o system card. *ArXiv*, abs/2410.21276, 2024a.
- OpenAI. Openai o1 system card, 2024b.
- Ibrahim H. Osman. Heuristics for the generalised assignment problem: simulated annealing and tabu search approaches. *Operations-Research-Spektrum*, 17:211–225, 1995.
- Ibrahim H. Osman and Nicos Christofides. Capacitated clustering problems by hybrid simulated annealing and tabu search. *International Transactions in Operational Research*, 1: 317–336, 1994.
- Christos Papadimitriou and Kenneth Steiglitz. *Combinatorial Optimization: Algorithms and Complexity*, volume 32. 01 1982. ISBN 0-13-152462-3. doi: 10.1109/TASSP.1984.1164450.
- Clifford C. Petersen. Computational experience with variants of the balas algorithm applied to the selection of r&d projects. *Management Science*, 13:736–750, 1967.

- Ruizhong Qiu, Zhiqing Sun, and Yiming Yang. DIMES: A differentiable meta solver for combinatorial optimization problems. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho (eds.), *Advances in Neural Information Processing Systems*, 2022.
- Qwen. Qwq-32b: Embracing the power of reinforcement learning. <https://qwenlm.github.io/blog/qwq-32b/>, 2025. Accessed: 2025-03-24.
- Rindranirina Ramamonjison, Timothy T. Yu, Raymond Li, Haley Li, Giuseppe 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. In *Neural Information Processing Systems*, 2023.
- M. S. W. Ratcliff and E. E. Bischoff. Allowing for weight considerations in container loading. *Operations-Research-Spektrum*, 20:65–71, 1998.
- Bernardino Romera-Paredes, Mohammadamin Barekatin, Alexander Novikov, Matej Balog, M Pawan Kumar, Emilien Dupont, Francisco J. R. Ruiz, Jordan S. Ellenberg, Pengming Wang, Omar Fawzi, Pushmeet Kohli, Alhussein Fawzi, Josh Grochow, Andrea Lodi, Jean-Baptiste Mouret, Talia Ringer, and Tao Yu. Mathematical discoveries from program search with large language models. *Nature*, 625:468 – 475, 2023.
- Noah Shinn, Federico Cassano, Beck Labash, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: language agents with verbal reinforcement learning. In *Neural Information Processing Systems*, 2023.
- Zhiqing Sun and Yiming Yang. Difusco: Graph-based diffusion solvers for combinatorial optimization. *ArXiv*, abs/2302.08224, 2023.
- E. Taillard. Benchmarks for basic scheduling problems. *European Journal of Operational Research*, 64(2):278–285, 1993.
- Jianheng Tang, Qifan Zhang, Yuhan Li, Nuo Chen, and Jia Li. Grapharena: Evaluating and improving large language models on graph computation. In *International Conference on Learning Representations*, 2025.
- Chrysafis Vogiatzis and Panos M. Pardalos. *Combinatorial optimization in transportation and logistics networks*, volume 2-5, pp. 673–722. Springer, Germany, January 2013. ISBN 9781441979964. doi: 10.1007/978-1-4419-7997-1.63. Publisher Copyright: © Springer Science+Business Media New York 2013. All rights are reserved.
- Hjalmar Wijk, Tao Lin, Joel Becker, Sami Jawhar, Neev Parikh, Thomas Broadley, Lawrence Chan, Michael Chen, Josh Clymer, Jai Dhyani, Elena Elicheva, Katharyn Garcia, Brian Goodrich, Nikola Jurkovic, Megan Kinniment, Aron Lajko, Seraphina Nix, Lucas Jun Koba Sato, William Saunders, Maksym Taran, Ben West, and Elizabeth Barnes. Re-bench: Evaluating frontier ai r&d capabilities of language model agents against human experts. *ArXiv*, abs/2411.15114, 2024.
- xAI. Grok-3 and the next phase of xai. <https://x.ai/news/grok-3>, 2025. Accessed: 2025-03-24.
- Ziyang Xiao, Dongxiang Zhang, Yangjun Wu, Lilin Xu, Yuan Jessica Wang, Xiongwei Han, Xiaojin Fu, Tao Zhong, Jia Zeng, Mingli Song, and Gang Chen. Chain-of-experts: When llms meet complex operations research problems. In *International Conference on Learning Representations*, 2024a.
- Ziyang Xiao, Dongxiang Zhang, Yangjun Wu, Lilin Xu, Yuan Jessica Wang, Xiongwei Han, Xiaojin Fu, Tao Zhong, Jia Zeng, Mingli Song, and Gang Chen. Chain-of-experts: When LLMs meet complex operations research problems. In *The Twelfth International Conference on Learning Representations*, 2024b.
- 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.

Zhicheng Yang, Yiwei Wang, Yinya Huang, Zhijiang Guo, Wei Shi, Xiongwei Han, Liang Feng, Linqi Song, Xiaodan Liang, and Jing Tang. Optibench meets resocratic: Measure and improve LLMs for optimization modeling. In *The Thirteenth International Conference on Learning Representations*, 2025.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. *ArXiv*, abs/2210.03629, 2022.

Haoran Ye, Jiarui Wang, Zhiguang Cao, Federico Berto, Chuanbo Hua, Haeyeon Kim, Jinkyoo Park, and Guojie Song. Reevo: Large language models as hyper-heuristics with reflective evolution. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024.

## A Bradley-Terry Score

In the Bradley-Terry framework (Bradley & Terry, 1952), each model  $i$  is associated with a latent ability parameter  $\theta_i$ , and the probability that model  $i$  outperforms model  $j$  in a pairwise comparison is given by

$$P(i \succ j) = \frac{\theta_i}{\theta_i + \theta_j}. \quad (7)$$

This formulation translates head-to-head outcomes into a quantitative measure of relative competitive strength.

For each evaluation instance, every possible pair of models is compared. A win is awarded if one model’s score exceeds the other’s, while a tie contributes a half-win to both models. Denote by  $w_{ij}$  the number of wins of model  $i$  against model  $j$  and by  $n_{ij}$  the total number of comparisons between the two models. The total wins for model  $i$  are then computed as:

$$W_i = \sum_{j \neq i} w_{ij}. \quad (8)$$

The latent ability parameters  $\theta_i$  are estimated by maximizing the likelihood of the observed pairwise outcomes using an iterative update rule. Starting from initial estimates (e.g.,  $\theta_i^{(0)} = 1$  for all models), the update for model  $i$  is given by:

$$\theta_i^{(t+1)} = \frac{W_i}{\sum_{j \neq i} \frac{n_{ij}}{\theta_i^{(t)} + \theta_j^{(t)}}}. \quad (9)$$

Iterations continue until the maximum change across all  $\theta_i$  falls below a predefined tolerance (e.g.,  $10^{-6}$ ), or until a maximum number of iterations is reached. The final  $\theta$  values represent the Bradley-Terry scores of each tested model.