CIRCUMVENTING THE COMPUTATIONAL HARDNESS OF COMPUTATIONAL ARBITRAGE

Anonymous authors
Paper under double-blind review

ABSTRACT

AI model marketplaces host providers with heterogeneous costs and capabilities. These differences create an opportunity for *computational arbitrage* (Olmedo et al., 2026), in which an intermediary purchases inference from a portfolio of models and resells it to users at a profit. Beyond creating profit opportunities for the arbitrageur, computational arbitrage offers financial value to both consumers and providers and could therefore significantly influence AI model markets. While prior work empirically demonstrates the feasibility of computational arbitrage for small model pools, scaling it to large marketplaces requires arbitrage strategies that are both computationally tractable and inexpensive to identify and deploy. In this work, we formalize computational arbitrage as an allocation problem and prove that it is generally NP-hard. However, we show that the optimal allocation can be computed efficiently under a natural diminishing-returns condition on how model performance scales with inference spending. We further develop a sample-efficient method for estimating model performance from limited data, reducing the upfront cost of identifying arbitrage strategies. Our analysis also yields several economic insights: at the optimum, every funded model delivers the same marginal utility per dollar; arbitrage directs revenue away from providers offering little utility relative to their prices; and it rewards complementary specialization.

1 INTRODUCTION

The proliferation of AI models with heterogeneous costs and capabilities makes it increasingly difficult for users to decide which service to choose. Even among models offering comparable performance, inference costs can differ substantially. For example, reaching a 40% solve rate on Terminal-Bench can cost anywhere from less than a dollar to more than $\$60$; reaching 70% can cost from roughly $\$10$ to over $\$180$ (see Figure 1). Such price dispersion creates the potential for *computational arbitrage* (Olmedo et al., 2026), where an intermediary allocates inference budget across model providers, achieves a given level of performance at a lower cost, and resells it to users at a profit. Computational arbitrage brings broad economic benefits, including lower consumer prices, reduced market segmentation, and easier market entry for smaller providers (Olmedo et al., 2026).

Despite these benefits, computational arbitrage is not yet deployed at scale in AI marketplaces, due to several challenges. First, given model prices and capabilities, an arbitrageur must determine how to allocate an inference budget across models to maximize utility. As the number of models grows, exhaustively searching over possible allocations becomes computationally infeasible. Second, even after the budget is allocated, the arbitrageur must determine the order in which to query models to avoid unnecessary spending. Third, computing an effective allocation requires knowing how each model's success probability varies across queries and inference budgets, quantities that are generally unavailable and prohibitively expensive to estimate, as doing so requires many paid model calls.

In this work, we address these challenges and take a step towards making arbitrage practical at scale. We first show that finding the optimal arbitrage strategy is computationally hard in general; however, we identify a natural condition under which the optimal allocation can be computed efficiently. We then show that the arbitrageur can adopt a greedy ordering rule that prioritizes models according to their expected utility per cost on unresolved queries. Finally, we develop a sample-efficient method for recovering utility curves from limited calibration data, substantially reducing the upfront cost of identifying arbitrage strategies.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

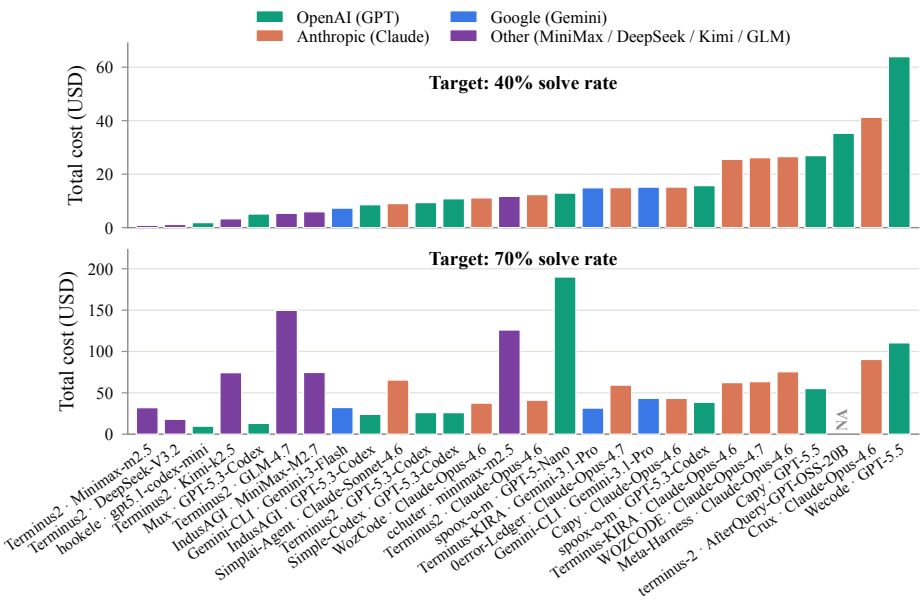


Figure 1: **The cost of a given performance target varies by more than an order of magnitude across the market.** Expected total cost for each model to reach a 40% (top) and a 70% (bottom) solve rate on Terminal-Bench 2.0. Models that are cheapest at 40% target are not the cheapest at 70%, suggesting that cost-effectiveness depends on the target performance. For each model, we give every query the same budget, compute the probability that repeated sampling solves it before the budget is exhausted, and increase the budget until the average solve rate reaches the target. "NA" marks a model that does not reach the target within the budget range considered. Prices are based on the reported cost per trial in the leaderboard when available, and otherwise token counts scaled by the LiteLLM price database as of September 2026. Appendix D.1 provides full reproduction details.

Our results yield several interesting economic implications. At the optimum, every funded model delivers the same marginal utility per dollar. Revenue therefore flows toward providers whose performance justifies their prices, while those offering little utility relative to their prices receive little or no revenue. This result contributes to the growing literature on LLM service pricing: regardless of the billing scheme used, a provider's ability to attract revenue ultimately depends on the utility it delivers per dollar. Providers can therefore compete in two ways: by offering sufficiently low prices or by specializing in queries that other models are likely to fail; computational arbitrage hence creates an incentive for complementary specialization. In more detail, we make the following contributions:

- **Optimal arbitrage strategy.** We formulate arbitrage as a budget-allocation problem and show that it is generally NP-hard. We then show that whenever inference spending has diminishing returns in log-failure reduction, the objective is jointly concave and its global optimum is computable efficiently even in large model markets. Across four model markets spanning agentic and coding tasks, our arbitrage strategy achieves a better cost–performance trade-off than any individual model[1]
- **Deployment order.** An arbitrageur can reduce their realized costs by querying funded models sequentially and stopping once an acceptable solution is found. Although we show that finding the optimal fixed order is NP-hard, we develop a greedy rule that prioritizes models with the highest expected utility per cost on queries missed by earlier models.
- **Sample-efficient utility estimation.** Building on distributional strategies in predicting inference-time scaling, we adapt empirical Bayes estimators that recover query-level utility curves from limited data. Across MATH, CodeContests, AIME 24 and 25, and MMLU benchmarks, an arbitrageur using our estimation method reaches nearly the same solve rate as an arbitrageur estimating the utility curves from orders of magnitude more data.

[1]Code and data are available in an anonymized repository.

2 THE OPTIMAL ARBITRAGE STRATEGY

We consider a market of N model providers where each model can map user queries, $x \in \mathcal{X}$, to proposed solutions $y \in \mathcal{Y}$. The quality of the solution depends on the model, the difficulty of the query, as well as the amount of inference budget, c, devoted to the query. For example, the budget could represent the number of sampling attempts (Chow et al., 2025) or reasoning effort (Wei et al., 2022). Each model i hence induces a conditional distribution $p_i(y \mid x, c)$. Let $A : \mathcal{X} \times \mathcal{Y} \to \{0, 1\}$ indicate whether y is an acceptable solution to query x. For a given query distribution $x \sim D$ and a per-query budget C, a rational user would spend the budget on the model that achieves the highest acceptance rate, i.e.,

$$\arg\max_{i \in [N]} \mathbb{E}_{x \sim D, y \sim p_i(\cdot|x,C)} \left[A(x, y) \right]. \tag{1}$$

This strategy is optimal whenever a single model dominates every other model on every query, but can be strictly suboptimal when models instead have *complementary* strengths. In this case, distributing the budget across several models can strictly outperform the best single model.[2]

An arbitrageur is an intermediary who benefits from this gap through diversification. Rather than allocating the entire budget to the most cost-effective model, the arbitrageur chooses a query-independent allocation $\boldsymbol{c} = (c_1, \ldots, c_N) \in \mathbb{R}^N_{\geq 0}$, such that $\sum_{i \in [N]} c_i \leq C$, i.e., for every query x, an inference budget of c_i is allocated to model i. Let $\boldsymbol{y} = (y_1, \ldots, y_N)$ denote the outputs produced by the N models under allocation $\boldsymbol{c}$. We denote their joint distribution as $p(\boldsymbol{y} \mid x, \boldsymbol{c})$, with marginals $p_i(y_i \mid x, c_i)$. The arbitrageur wants to maximize the acceptance probability, given the budget, i.e.,

$$\underset{\boldsymbol{c} \in \mathbb{R}^N_{\geq 0}}{\text{maximize}} \quad U(\boldsymbol{c}) := \mathbb{E}_{x \sim D, \boldsymbol{y} \sim p(\cdot|x,\boldsymbol{c})} \left[1 - \prod_{i=1}^{N} \big(1 - A(x, y_i)\big) \right] \quad \text{s.t.} \sum_{i \in [N]} c_i \leq C. \tag{2}$$

The combinatorial nature of the problem makes it computationally intractable; formally we have the following result:[3]

Theorem 1. *Finding the optimal arbitrage policy as in Equation* (2) *is NP-hard.*

We therefore identify conditions under which the problem becomes tractable. For each model, we first define the utility function $u_i : \mathcal{X} \times \mathbb{R}_{\geq 0} \to [0, 1]$ as $u_i(x; c_i) := \mathbb{E}_{y \sim p_i(.|x,c_i)}[A(x, y)]$, and assume that model outputs are conditionally independent given the query.[4] We can hence rewrite the objective in Equation (2) as follows:

$$U(\boldsymbol{c}) = \mathbb{E}_{x \sim D} \left[1 - \prod_{i=1}^{N} (1 - u_i(x; c_i)) \right] = \mathbb{E}_{x \sim D} \left[1 - e^{-\sum_{i=1}^{N} \alpha_i(x, c_i)} \right] \quad \text{s.t.} \sum_{i \in [N]} c_i \leq C, \tag{3}$$

where $\alpha_i(x, c) := -\log\big(1 - u_i(x, c)\big)$, i.e., the log-failure reduction on query x purchased with c dollars from model i. Conditional independence of model outcomes given a query does not restore tractability; we therefore impose a natural diminishing property on how model performance scales with inference budget spending, more formally we have the following assumption:

Assumption 1. *For every model $i \in [N]$ and query $x \in \mathcal{X}$, the log-failure reduction $c \mapsto \alpha_i(x, c)$ is non-decreasing and concave on $\mathbb{R}_{\geq 0}$.*

Assumption 1 formalizes diminishing returns to inference compute: it requires the log-failure reduction purchased from each additional dollar to be non-increasing. This property has been empirically observed by common inference-scaling strategies, including repeated sampling (Schaeffer et al., 2025), chain-of-thought reasoning (Ghosal et al., 2025; Snell et al., 2025), and post-training (Chen et al., 2026). This assumption makes the allocation problem tractable:

[2] Example 1 in Appendix A provides a simple two-model example in which the distribution of the budget among the two models strictly beats every single-model allocation of the same total budget.

[3] All proofs can be found in Appendix B.

[4] This assumption says that conditional on a query, the sampling processes of different LLMs are mutually independent; a natural assumption made also in prior work (Ai et al., 2025, Assumption C.1; Farinhas et al., 2024). This assumption does not restrict arbitrary overlap in models' strengths and weaknesses *across* queries.

Theorem 2. *Under Assumption 1, U is non-decreasing and jointly concave. If D has finite support and each $\alpha_i(x;\cdot)$ is differentiable, a global optimum exists and is characterized by the KKT conditions. This optimum can be found efficiently using coordinate-ascent on the Lagrangian.*

Guided by the above theorem, we maximize $U(\mathbf{c}) - \lambda \sum_i c_i$ using coordinate ascent and use bisection over the shadow price $\lambda \geq 0$ to meet the budget constraint. Note that a coordinate-ascent sweep scales linearly with the number of models whereas exhaustive grid search, as in Olmedo et al. (2026), scales exponentially. On the markets of Section 5, with up to 70 models, our method runs in under half a second, whereas exhaustive search would be computationally infeasible at this scale. The resulting optimality conditions yield several interesting insights discussed below.

No further arbitrage opportunities. At the optimum, every funded model delivers the same marginal utility per dollar, i.e.,

$$\frac{\partial U(\boldsymbol{c}^*)}{\partial c_i} = \mathbb{E}_{x\sim D}\left[\underbrace{\frac{\partial \alpha_i(x, c_i^*)}{\partial c_i}}_{\substack{\text{Marginal log-failure reduction} \\ \text{purchased per dollar for } x}} \underbrace{e^{-\sum_{\ell=1}^{N} \alpha_\ell(x, c_\ell^*)}}_{\substack{\text{Probability of failure of} \\ \text{the model pool on } x}}\right] \begin{cases} = \lambda, & c_i^* > 0, \\ \leq \lambda, & c_i^* = 0. \end{cases} \tag{4}$$

As a result, reallocation of the budget between funded models cannot raise utility, i.e., no more arbitrage possibilities are available, which suggests the existence of a single consistent price system (Ross, 1978; Harrison & Kreps, 1979; Dybvig & Ross, 1987; Varian, 1987).

Arbitrage against arbitrary pricing. A growing line of recent work identifies shortcomings in current LLM pricing practices (Artola Velasco et al., 2026; Cai et al., 2025; Saig et al., 2024), and proposes alternatives such as pay-per-character or pay-for-performance. Computational arbitrage offers a complementary perspective: the arbitrageur pays $1/\lambda$ dollars per unit of marginal utility from every funded provider, i.e., the effective price of marginal utility is equalized across providers, and providers whose marginal utility per dollar falls below λ receive no revenue at all. This disciplines pricing regardless of the particular billing scheme each provider adopts. This mechanism parallels the role of arbitrage in goods markets: arbitrage pushes prices to their fundamental values (Ross, 1976; Dybvig & Ross, 1987; Shleifer, 2000), and constrains firms' ability to sustain discriminatory prices (Malueg & Schwartz, 1994; Anderson & Ginsburgh, 1999).

Providers benefit from specialization. Suppose that providers are aware of the arbitrageur's allocation rule and want to increase their marginal utility per dollar. Based on Equation (4), they have two ways: either generally reduce their price, or improve on queries where other models are more likely to fail. Suppose provider i can make a small local improvement to its marginal log-failure reduction at allocation $\boldsymbol{c}$. Let $h_i : \mathcal{X} \to \mathbb{R}_{\geq 0}$ describe where this improvement is targeted, so that $\frac{\partial \tilde{\alpha}_i(x,c_i)}{\partial c_i} = \frac{\partial \alpha_i(x,c_i)}{\partial c_i} + h_i(x)$ and assume a fixed improvement budget, for example $\mathbb{E}_{x\sim D}[h_i(x)] \leq H$. The resulting change in the provider's marginal return is $\frac{\partial \tilde{U}(\boldsymbol{c})}{\partial c_i} - \frac{\partial U(\boldsymbol{c})}{\partial c_i} = \mathbb{E}_{x\sim D}\left[h_i(x) e^{-\sum_{\ell=1}^{N} \alpha_\ell(x,c_\ell)}\right]$. Thus, a provider benefits most by improving its marginal log-failure reduction on queries for which the current model pool has the highest probability of failure. Furthermore, if every provider improves on the same region of D, residual demand there collapses, creating an incentive for complementary specialization.

3 Optimal deployment order

The optimal allocation identifies how to distribute inference spending across providers to maximize utility under a given budget. Deploying this allocation raises a complementary question: in what order should the models be queried? In verifiable domains, the arbitrageur can stop as soon as an acceptable solution is found, so deployment order affects realized spending without changing the allocation's success probability. Taking the allocation as fixed, we characterize cost-minimizing deployment orders and examine their implications for provider revenues and incentives.

Let π be the permutation giving the deployment order. We use the survival function identity to model the spend on each model; after spending t dollars on model i for solving sample x, the probability that it has yet not succeeded is $1 - u_i(x;t) = e^{-\alpha_i(x,t)}$. The expected spend on model i for query x, given that it is reached is hence,

$$R_i(x, c_i) := \int_0^{c_i} e^{-\alpha_i(x,t)} \, \mathrm{d}t. \tag{5}$$

The expected spend for solving query x is hence:

$$R^\pi(x, \boldsymbol{c}) = \sum_{i=1}^{N} R_{\pi(i)}(x, c_{\pi(i)}) e^{-\sum_{j<i} \alpha_{\pi(j)}(x, c_{\pi(j)})}. \tag{6}$$

The query-dependent cost-minimizing order then has a simple characterization:

Proposition 1. *For a fixed allocation* $\mathbf{c}$ *and instance* x*, the order* π *that minimizes* $R^\pi(x; \mathbf{c})$ *queries the models in decreasing order of* $\frac{u_i(x;c_i)}{R_i(x,c_i)}$.

The per-query optimum provides a benchmark for deployment and motivates query-adaptive ordering when reliable performance predictions are available, paralleling results in LLM routing literature. Without such predictions, we seek a fixed order that minimizes expected cost across the query population. In the following, we show that finding the optimal deployment order in this case is NP-hard, using a reduction from the min-sum set cover problem.

Theorem 3. *Finding a fixed deployment order that minimizes* $\mathbb{E}_{x\sim D}[R^\pi(x, \boldsymbol{c})]$ *is NP-hard.*

Given this hardness result, we adopt a greedy ordering rule inspired by the 4-approximate algorithm for min-sum-set-cover (Feige et al., 2004). Starting from an empty set, each time we append the model with the largest reach-weighted coverage per expected dollar. Let S_t be the set of models already queried up to step t and let $\rho_{S_t}(x) = \prod_{i\in S_t}(1 - u_i(x, c_i))$; then our greedy algorithm picks a model at step $t + 1$ such that:

$$\pi(t+1) = \arg\max_{i \notin S_t} \frac{\mathbb{E}_{x\sim D}[\rho_{S_t}(x)\, u_i(x; c_i)]}{\mathbb{E}_{x\sim D}\left[\rho_{S_t}(x)\, R_i(x; c_i)\right]}, \qquad S_t = \{\pi(1), \ldots, \pi(t)\}, \tag{7}$$

i.e., the model with the highest expected utility per expected dollar spent on yet unsolved samples is queried next.

Arbitrage incentivizes models to either be cheap generalists or specialists. Given the above greedy deployment order, model providers can have two strategies for maximizing their revenue. First note that for every query the probability of reaching the ith model in the order, $\exp\left(\sum_{j<i} -\alpha_{\pi(j)}(x, c_{\pi(j)})\right)$, is the probability that models earlier in the order fail to solve the query, and it decreases monotonically for models later in the sequence. As a result, there is a clear advantage for providers to appear early in the deployment order. To do so, they need to offer a competitive utility at a low price, as in the first run of the greedy algorithm the model with highest *average* utility per dollar will be selected. In later runs, the models with the highest utility per dollar on *unsolved* samples get selected. In short, models should either simply be cheap enough to appear early in the deployment order and be allocated budget or provide complementary strength, solving samples on which other models fail.

4 Reducing the cost of arbitrage

Computing the optimal allocation requires query-level utility curves. Estimating these curves with standard unbiased estimators requires many recorded attempts per model–query pair and since each attempt is a paid model call, collecting them can be prohibitively expensive. In this section, we adapt a distributional approach to obtain query-level empirical Bayes estimates from limited observations.

We consider a repeated sampling strategy and assume that $u_i(x, c_i) := \mathrm{pass}_{i,x}@k$ where $k = c_i/q_i(x)$ with $q_i(x)$ representing the average cost per attempt of model i on query x. Following prior distributional approaches (Kazdan et al., 2025; Schaeffer et al., 2025), we model one-shot success probabilities across queries with a model-specific Beta distribution,[5] and fit its parameters

[5]Under this model Schaeffer et al. (2025) show that the negative log of the average success rate follows a power law, consistent with empirical observations (Brown et al., 2024).

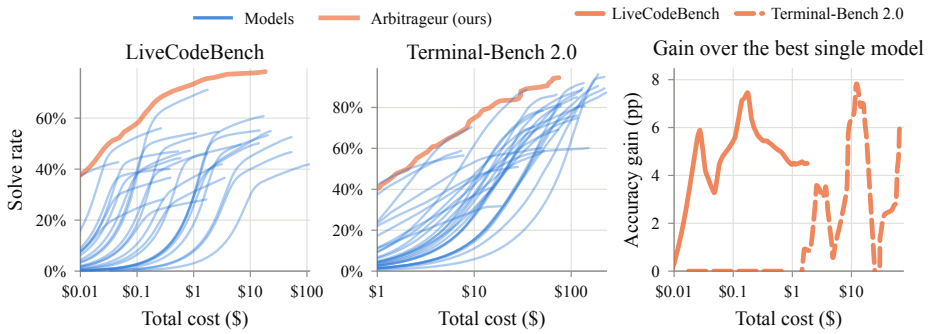


Figure 2: **Arbitrageur is the cost–performance frontier.** Solve rate against total cost on LiveCodeBench and Terminal-Bench 2.0. Each blue curve is one model funded on its own by repeated sampling with early stopping when a solution is found. Arbitrageur's allocation is cross-validated over the problems and deployed in the greedy population order. The rightmost panel shows the resulting gain in solve rate over the best single model, as in Equation (1), at the same total cost, in percentage points, for both benchmarks. The gain is largest in the mid-budget regime where models' strengths are most complementary.

using the Beta–Binomial marginal likelihood, i.e.,

$$(\hat{a}_i, \hat{b}_i) = \arg\max_{a,b} \prod_{j=1}^{|\mathcal{D}|} \binom{n_{i,x}}{s_{i,x}} \frac{\text{Beta}(a + s_{i,x},\, b + n_{i,x} - s_{i,x})}{\text{Beta}(a, b)},$$

where $n_{i,x}$ is the total number of attempts of model i for sample x and $s_{i,x}$ is the number of successful such attempts. Note that this approach can only be used to estimate the dataset-level pass@k for every model. This is insufficient for computational arbitrage as the arbitrageur depends on the entire matrix of model-query utility curves $u_i(x, c_i)$ to find the optimal allocation. We therefore adapt the fitted Beta–Binomial model into a query-level empirical-Bayes estimator. For every model, the fitted distribution $\text{Beta}(\hat{a}_i, \hat{b}_i)$ is used as a prior for the query-specific one-shot success probability pass@1. Given the observed attempts for model i on query x, the posterior distribution of pass@1 is $\text{Beta}(\hat{a}_i + s_{i,x}, \hat{b}_i + n_{i,x} - s_{i,x})$. Hence, the resulting query-level pass@k estimator can be found as follows:

$$\widehat{\text{pass}_{i,x}\,@k} = \mathbb{E}_{\text{pass}_{i,x}\,@1 \sim \text{Beta}(\hat{a}_i + s_{i,x}, \hat{b}_i + n_{i,x} - s_{i,x})}[1 - (1 - \text{pass}_{i,x}\,@1)^k]. \tag{8}$$

In practice, we fit the prior with a point mass at $p = 0$ for queries a model never solves and a weak hyperprior $p(a_i, b_i) \propto (a_i + b_i)^{-5/2}$ (Gelman et al., 2013) on its concentration, since with only two or four attempts per query most cells have no observed success, and without these the marginal likelihood is flat along $a_i/(a_i + b_i)$ and the fit runs to a point mass. In the next section, we show that arbitrage strategies based on our estimates achieve near-oracle solve rates while requiring orders of magnitude fewer model calls for calibration.

5 Experiments

In this part we evaluate our proposed arbitrage strategy (based on Theorem 2) together with the greedy deployment order, as in Equation (7), on four agentic and competitive programming tasks. We further evaluate our estimator proposed in Section 4 on several benchmarks with large public records of repeated sampling, which serves as a high-precision reference for query-level utility curves.

5.1 Arbitrage on real-world model markets

In this section we evaluate our arbitrage strategy on recent benchmarks spanning terminal workflows and competitive-programming problems. They evaluate many models over repeated trials, allowing us to study both cost–performance trade-offs and differences in model strengths.

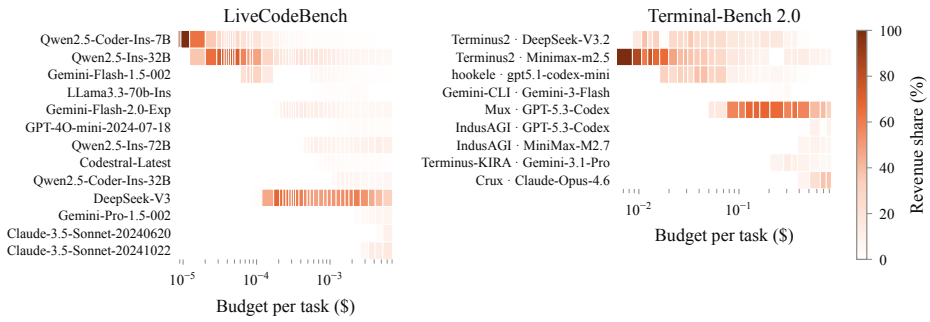


Figure 3: **Revenue spreads across providers as budgets increase.** Revenue share of each funded provider as a function of the budget per task under the arbitrageur's cross-validated allocation of Figure 2, with providers sorted by per-attempt price (cheapest on top). At lower budgets, revenue goes entirely to the cheaper models; as the budget grows, it spreads among more providers, and even at high budgets cheap models retain a significant share of the revenue.

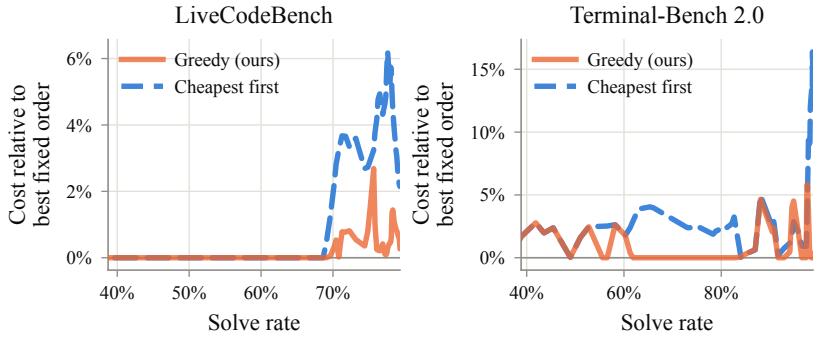


Figure 4: **The greedy deployment order is close to the best fixed order.** Expected spend of the arbitrageur's allocation when the funded models are queried in the greedy order of Equation (7) (solid) or in increasing order of average cost per attempt (dashed), relative to the best fixed order of the funded models found by exact dynamic programming over the funded models (which remains exponential in the number of funded models, the 0% line), as a function of the solve rate of the allocation on LiveCodeBench and Terminal-Bench 2.0. The greedy order matches the best order at the median and stays within 6% of it, whereas cheapest first spends up to 16% more.

We evaluate arbitrage on Terminal-Bench 2.0 (28 models, 89 tasks, at most 5 recorded attempts per pair) and LiveCodeBench (24 models, 713 problems, 10 recorded attempts per pair). We estimate the mean cost per attempt $q_i(x)$ for all model–query pairs using the reported cost per trial when available, and otherwise the recorded token counts scaled by public prices as of September 2026. We present the results on two additional benchmarks, *Terminal-Bench 4.0* (12 models, 66 tasks) and *DeepSWE* (70 models, 111 tasks), as well as more details on the experimental setup, in Appendix D.

For a model–query pair, the expected spend of committing budget c_i to query x under repeated sampling that stops upon success is calculated as in Equation (5); for a portfolio, models are queried in the greedy order of Section 3 and the expected spend is calculated as in Equation (6). The arbitrageur's frontier is traced by sweeping the shadow price λ over a fixed grid of 86 values; at each, the allocation and its deployment order are cross-validated (88 folds for TerminalBench and 10 for LiveCodeBench). Total cost is expected spend per query summed over the problems.

Figure 2 reports the solve rate of the arbitrageur and all the models alone, against their total spend which shows that **the arbitrageur consistently improves the cost–performance frontier**. The largest gains occur at intermediate budgets, where combining models with complementary strengths is most valuable. Figure 3 shows the revenue share of each model provider under the arbitrage strat-

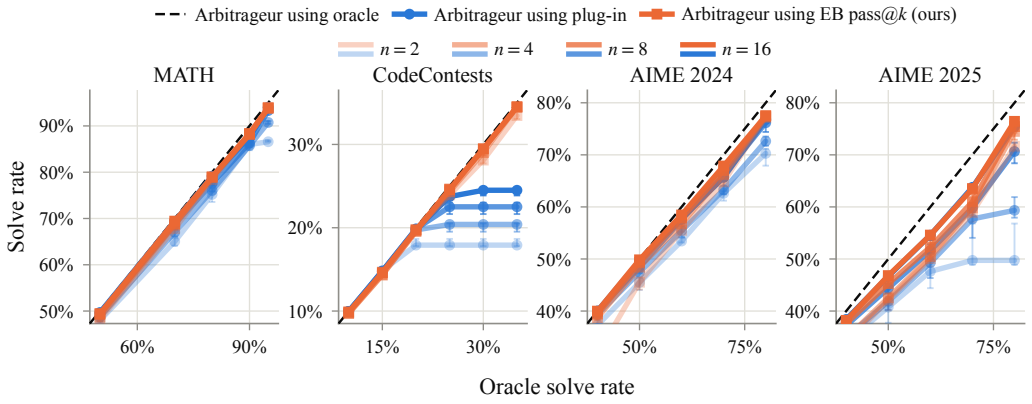


Figure 5: **An arbitrageur using our estimation method approaches the solve rate of the one using the ground truth utility curves which uses several orders of magnitude more data.** On each benchmark of Figure 9 we subsample $n \in \{2, 4, 8, 16\}$ attempts per query–model pair, re-estimate the utility curves, run the arbitrage on the estimates and plot the achieved solve rate; the oracle uses all recorded attempts (10,000 per pair on the Monkey Business benchmarks, $\sim$2,500 on AIME) and is the diagonal. For each Oracle solve rate we find the budget that the Oracle needs to achieve that solve rate and use the same budget for other methods.

egy. At lower budgets, revenue goes entirely to the cheaper models; as the budget grows, it spreads among more providers, and even at high budgets cheap models retain a significant share of the revenue. These results suggest that **arbitrage can reduce market segmentation by directing revenue towards multiple providers, including smaller models that offer utility for a competitive price**.

Figure 4 compares the realized spend of our greedy order proposed in Section 3 with the best fixed order of the funded models, computed by exhaustive search, and with the common heuristic of querying models cheapest first. **Our greedy order matches the best order at most budgets** and is never more than 6% above it, while cheapest first spends up to 6% more on LiveCodeBench and 16% more on Terminal-Bench.

5.2 Arbitrage with estimated utility curves

To evaluate the quality of our estimation method proposed in Section 4, we need to determine how much of the utility does an arbitrageur who uses our method to estimate the utility curves lose, compared to an arbitrageur who finds the true utility curves. We utilize benchmarks with many recorded attempts per query–model pair, allowing us to construct high-precision references for utility curves. We use MATH (10 models, 128 problems) and CodeContest (5 models, 140 problems) from the Monkey Business benchmarks (Brown et al., 2024) which provide $10,000$ recorded binary attempts per query–model pair as well as AIME 2024-2025, Global MMLU Lite (Singh et al., 2025), and MMLU-Pro (Wang et al., 2024b) (12 models, 30 problems per benchmark) where we use the $2,500$ recorded binary attempts per pair from Truong et al. (2026). Results for the MMLU benchmarks are reported in Appendix D.

We consider three estimation methods: (i) the *Oracle* which estimates $\text{pass}_{i,x}@k$ based on all the available attempts using the unbiased estimator of Chen et al. (2021), (ii) the *Plug-in* which estimates based on n attempts per model-query pair, using the plug-in estimator $\hat{p} = s_{i,x}/n_{i,x}$ and extrapolates $1-(1-\hat{p})^k$ to any k, (iii) our *Empirical Bayes (EB)* estimator which estimates based on n attempts per pair using Equation (8).

We first compare our method to the plug-in estimator in terms of the RMSE of the estimated pass@k curves, using the Oracle as the ground truth. Figure 9 in Appendix D shows that **our estimator consistently has lower RMSE than the plug-in estimator**, suggesting that pooling information across queries (through the model-level prior) substantially improves utility estimation in the low-data regime. We next visualize the solve rate of an arbitrageur using different estimators. Figure 5

summarizes the results where for each oracle solve rate, all methods spend the deployment budget the oracle requires for that rate, and we report medians over 10 independent draws. It can be seen that **an arbitrageur using our estimation method tracks the oracle solve rate which uses up to orders of magnitude more data, and hence more cost**. On the CodeContests benchmark, the plug-in estimator often observes no successful attempts for a query and therefore assigns it $\hat{p} = 0$, regardless of k. Its achievable solve rate is consequently limited to queries with at least one observed success. Our estimator avoids this failure mode by borrowing strength across queries through a model-specific prior. Results for the MMLU benchmarks are in Figure 10.

6 Further Related work

Model cascading. In model cascading, models are queried sequentially, typically in increasing order of cost, until a satisfactory solution is found (Varshney & Baral, 2022; Wang et al., 2023b; Chen et al., 2023; Aggarwal et al., 2024; Ramírez et al., 2024). Computational arbitrage addresses the problem of deciding which models should even be funded and how should the inference budget be distributed among them to maximize performance. Consequently, an optimal arbitrage strategy may exclude models that are dominated by the rest of the portfolio, and its cost-minimizing deployment order is not necessarily in increasing model cost.

Model routing. Model routing assigns each query to one of the several available models, typically using predicted difficulty, quality, or cost to improve the cost–performance trade-off (Shnitzer et al., 2024; Ding et al., 2024; Lu et al., 2024; Šakota et al., 2024). Computational arbitrage addresses a complementary problem: rather than selecting a single model, it determines how to distribute inference spending across a portfolio of providers, potentially querying and repeatedly sampling from multiple models for the same query. Routing could nevertheless strengthen an arbitrage strategy by predicting query-level model performance. Conversely, research on model routing can adopt arbitrage strategies as benchmark for algorithm development.

Inference-time scaling. A growing line of work studies the scaling behaviour of LLMs in various aspects including context length Xiong et al., in-context learning Agarwal et al. (2024), inference optimization Snell et al. (2025); Liu et al. (2023) and repeated sampling (Brown et al., 2024; Schaeffer et al., 2025; Kazdan et al., 2025). Our work is most closely related to studies of repeated sampling where we estimate query-level scaling curves under repeated sampling to identify arbitrage strategies using limited calibration data. Examining arbitrage strategies under other inference-time scaling methods such as chain-of-thought prompting (Ghosal et al., 2025; Snell et al., 2025; Wei et al., 2022; Wang et al., 2023a), majority voting Wan et al. (2025), and best-of-N Kalayci et al. (2025); Komiyama et al. (2026) is an interesting direction for future work.

Economics of LLMs. A rapidly growing literature addresses the economic roles that LLMs can play in society (Fish et al., 2024a; Boehmer et al., 2025; Filippas et al., 2024; Kovařík et al., 2023; Raman et al., 2024; Fish et al., 2024b; Dütting et al., 2024), examines cost-aware fine-tuning Laufer et al. (2024), test-time training Chen et al. (2026), evaluation Erol et al. (2026); Wang et al. (2024a) and pricing and incentive problems in LLM markets, including opacity and potential overcharging (Artola Velasco et al., 2026; Sun et al., 2025; Cai et al., 2025; Saig et al., 2024). We study how computational arbitrage can itself reshape the economics of LLM markets by reallocating demand across them, affecting their revenue shares, and creating incentives for competitive pricing and complementary specialization.

7 Conclusion

In this work we take a step towards making computational arbitrage practical in large AI model markets. We identify a natural condition under which the optimal arbitrage strategy can be computed efficiently. At this optimal allocation, every funded model provides the same marginal utility per dollar, directing revenue toward providers that are either broadly cost-effective or specialized in queries that the rest of the portfolio is likely to fail on. The deployment order reinforces these incentives: inexpensive models with strong average performance tend to be queried early, while complementary specialists remain valuable later in the sequence. Together, these findings suggest that arbitrage can discipline pricing while rewarding specialization in AI model markets.

AI USE STATEMENT

In this work, we used generative AI tools to aid or polish writing, assist with or provide feedback on the theoretical results, format references, create or modify scientific figures, and assist with the implementations. All mathematical arguments, references, experimental results, and AI-suggested text were independently reviewed and verified by the authors. We have not used generative AI tools for research ideation, generating datasets, drafting sections of the paper, or interpreting results. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

ETHICS STATEMENT

All authors have read and agree to adhere to the ICLR Code of Ethics. This work complies with all ethical guidelines outlined therein.

REPRODUCIBILITY STATEMENT

Detailed proofs of the theoretical statements are provided in Appendix B. Details on the benchmarks, experimental procedures, filters applied, cost construction and pricing are in Section 5. Additional details are reported in Appendices D.1 to D.4. The code, data, and instructions to reproduce the results are in an anonymized repository: `https://anonymous.4open.science/r/arb_code-B760/README.md`.

REFERENCES

Rishabh Agarwal, Avi Singh, Lei Zhang, Bernd Bohnet, Luis Rosias, Stephanie Chan, Biao Zhang, Ankesh Anand, Zaheer Abbas, Azade Nova, et al. Many-shot in-context learning. *Advances in Neural Information Processing Systems*, 37:76930–76966, 2024.

Pranjal Aggarwal, Aman Madaan, Ankit Anand, Srividya Pranavi Potharaju, Swaroop Mishra, Pei Zhou, Aditya Gupta, Dheeraj Rajagopal, Karthik Kappaganthu, Yiming Yang, et al. Automix: Automatically mixing language models. *Advances in Neural Information Processing Systems*, 37:131000–131034, 2024.

Rui Ai, Yuqi Pan, David Simchi-Levi, Milind Tambe, and Haifeng Xu. Beyond majority voting: Llm aggregation by leveraging higher-order information. *arXiv preprint arXiv:2510.01499*, 2025.

Simon P. Anderson and Victor A. Ginsburgh. International pricing with costly consumer arbitrage. *Review of International Economics*, 7(1):126–139, 1999. doi: 10.1111/1467-9396.00151.

Ander Artola Velasco, Stratis Tsirtsis, Nastaran Okati, and Manuel Gomez-Rodriguez. Is your LLM overcharging you? tokenization, transparency, and incentives. In *Proceedings of the 43rd International Conference on Machine Learning*, volume 306 of *Proceedings of Machine Learning Research*, 2026. Oral presentation.

Niclas Boehmer, Sara Fish, and Ariel D. Procaccia. Generative social choice: The next generation. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 4627–4659. PMLR, 2025. URL `https://proceedings.mlr.press/v267/boehmer25a.html`.

Stephen Boyd and Lieven Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004.

Bradley Brown, Jordan Juravsky, Ryan Ehrlich, Ronald Clark, Quoc V. Le, Christopher Ré, and Azalia Mirhoseini. Large language monkeys: Scaling inference compute with repeated sampling. *arXiv preprint arXiv:2407.21787*, 2024.

Will Cai, Tianneng Shi, Xuandong Zhao, and Dawn Song. Are you getting what you pay for? auditing model substitution in LLM APIs. In *Workshop on Regulatable Machine Learning at the 39th Conference on Neural Information Processing Systems*, 2025. URL `https://openreview.net/forum?id=thhrtv9P0s`.

Lingjiao Chen, Matei Zaharia, and James Zou. FrugalGPT: How to use large language models while reducing cost and improving performance. *Transactions on Machine Learning Research*, 2023.

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.

Yatong Chen, Guanhua Zhang, and Moritz Hardt. Leaderboard incentives: Model rankings under strategic post-training. In *Proceedings of the 43rd International Conference on Machine Learning*, volume 306 of *Proceedings of Machine Learning Research*, 2026. URL https://arxiv.org/abs/2603.08371.

Yinlam Chow, Guy Tennenholtz, Izzeddin Gur, Vincent Zhuang, Bo Dai, Aviral Kumar, Rishabh Agarwal, Sridhar Thiagarajan, Craig Boutilier, and Aleksandra Faust. Inference-aware fine-tuning for best-of-N sampling in large language models. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=77gQUdQhE7.

Dujian Ding, Ankur Mallick, Chi Wang, Robert Sim, Subhabrata Mukherjee, Victor Rühle, Laks V. S. Lakshmanan, and Ahmed Hassan Awadallah. Hybrid LLM: Cost-efficient and quality-aware query routing. In *The Twelfth International Conference on Learning Representations*, 2024.

Paul Dütting, Vahab Mirrokni, Renato Paes Leme, Haifeng Xu, and Song Zuo. Mechanism design for large language models. In *Proceedings of the ACM Web Conference 2024*, pp. 144–155. Association for Computing Machinery, 2024. doi: 10.1145/3589334.3645511. URL https://doi.org/10.1145/3589334.3645511.

Philip H. Dybvig and Stephen A. Ross. Arbitrage. In John Eatwell, Murray Milgate, and Peter Newman (eds.), *The New Palgrave: A Dictionary of Economics*, volume 1, pp. 100–106. Macmillan, London, 1987.

Mehmet Hamza Erol, Batu El, Mirac Suzgun, Mert Yuksekgonul, and James Y Zou. Cost-of-pass: An economic framework for evaluating language models. In *International Conference on Learning Representations*, volume 2026, pp. 34103–34127, 2026.

António Farinhas, Haau-Sing Li, and André F Martins. Reranking laws for language generation: A communication-theoretic perspective. *Advances in Neural Information Processing Systems*, 37: 111074–111105, 2024.

Uriel Feige, László Lovász, and Prasad Tetali. Approximating min sum set cover. *Algorithmica*, 40 (4):219–234, 2004.

Apostolos Filippas, John J. Horton, and Benjamin S. Manning. Large language models as simulated economic agents: What can we learn from homo silicus? In *Proceedings of the 25th ACM Conference on Economics and Computation*, pp. 614–615. Association for Computing Machinery, 2024. doi: 10.1145/3670865.3673513. URL https://doi.org/10.1145/3670865.3673513.

Sara Fish, Paul Gölz, David C. Parkes, Ariel D. Procaccia, Gili Rusak, Itai Shapira, and Manuel Wüthrich. Generative social choice. In *Proceedings of the 25th ACM Conference on Economics and Computation*, pp. 985. Association for Computing Machinery, 2024a. doi: 10.1145/3670865.3673547. URL https://doi.org/10.1145/3670865.3673547.

Sara Fish, Yannai A. Gonczarowski, and Ran I. Shorrer. Algorithmic collusion by large language models. *arXiv preprint arXiv:2404.00806*, 2024b. doi: 10.48550/arXiv.2404.00806. URL https://arxiv.org/abs/2404.00806. Accepted at the 27th ACM Conference on Economics and Computation (EC 2026).

Andrew Gelman, John B. Carlin, Hal S. Stern, David B. Dunson, Aki Vehtari, and Donald B. Rubin. *Bayesian Data Analysis*. Chapman and Hall/CRC, 3 edition, 2013.

Soumya Suvra Ghosal, Souradip Chakraborty, Avinash Reddy, Yifu Lu, Mengdi Wang, Dinesh Manocha, Furong Huang, Mohammad Ghavamzadeh, and Amrit Singh Bedi. Does thinking more always help? mirage of test-time scaling in reasoning models. In *Advances in Neural Information Processing Systems*, volume 38, 2025. doi: 10.52202/085713-5747.

J. Michael Harrison and David M. Kreps. Martingales and arbitrage in multiperiod securities markets. *Journal of Economic Theory*, 20(3):381–408, 1979. doi: 10.1016/0022-0531(79)90043-7.

Naman Jain, King Han, Alex Gu, Wen-Ding Li, Fanjia Yan, Tianjun Zhang, Sida Wang, Armando Solar-Lezama, Koushik Sen, and Ion Stoica. LiveCodeBench: Holistic and contamination free evaluation of large language models for code. In *The Thirteenth International Conference on Learning Representations*, 2025.

Yusuf Kalayci, Vinod Raman, and Shaddin Dughmi. Optimal stopping vs best-of-N for inference time optimization. *arXiv preprint arXiv:2510.01394*, 2025. doi: 10.48550/arXiv.2510.01394.

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B. Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*, 2020.

Richard M. Karp. Reducibility among combinatorial problems. In Raymond E. Miller, James W. Thatcher, and Jean D. Bohlinger (eds.), *Complexity of Computer Computations*, pp. 85–103. Plenum Press, New York, 1972.

Joshua Kazdan, Rylan Schaeffer, Youssef Allouah, Colin Sullivan, Kyssen Yu, Noam Levi, and Sanmi Koyejo. Efficient prediction of pass@k scaling in large language models. *arXiv preprint arXiv:2510.05197*, 2025. doi: 10.48550/arXiv.2510.05197.

Junpei Komiyama, Daisuke Oba, and Masafumi Oyamada. Best-of-infinity: Asymptotic performance of test-time LLM ensembling. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://arxiv.org/abs/2509.21091.

Vojtěch Kovařík, Caspar Oesterheld, and Vincent Conitzer. Game theory with simulation of other players. In *Proceedings of the Thirty-Second International Joint Conference on Artificial Intelligence*, pp. 2800–2807. International Joint Conferences on Artificial Intelligence Organization, 2023. doi: 10.24963/ijcai.2023/312. URL https://www.ijcai.org/proceedings/2023/312.

Benjamin Laufer, Jon Kleinberg, and Hoda Heidari. Fine-tuning games: Bargaining and adaptation for general-purpose models. In *Proceedings of the ACM Web Conference 2024*, pp. 66–76, 2024.

Xin Liu, Daniel McDuff, Geza Kovacs, Isaac Galatzer-Levy, Jacob Sunshine, Jiening Zhan, Ming-Zher Poh, Shun Liao, Paolo Di Achille, and Shwetak Patel. Large language models are few-shot health learners. *arXiv preprint arXiv:2305.15525*, 2023.

Keming Lu, Hongyi Yuan, Runji Lin, Junyang Lin, Zheng Yuan, Chang Zhou, and Jingren Zhou. Routing to the expert: Efficient reward-guided ensemble of large language models. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pp. 1964–1974, 2024.

David A. Malueg and Marius Schwartz. Parallel imports, demand dispersion, and international price discrimination. *Journal of International Economics*, 37(3–4):167–195, 1994. doi: 10.1016/0022-1996(94)90044-2.

Ricardo Olmedo, Bernhard Schölkopf, and Moritz Hardt. Computational arbitrage in AI model markets. In *Proceedings of the 43rd International Conference on Machine Learning*, volume 306 of *Proceedings of Machine Learning Research*, 2026. URL https://arxiv.org/abs/2603.22404.

Narun Krishnamurthi Raman, Taylor Lundy, Samuel Joseph Amouyal, Yoav Levine, Kevin Leyton-Brown, and Moshe Tennenholtz. STEER: Assessing the economic rationality of large language models. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 42026–42047. PMLR, 2024. URL https://proceedings.mlr.press/v235/raman24b.html.

Guillem Ramírez, Alexandra Birch, and Ivan Titov. Optimising calls to large language models with uncertainty-based two-tier selection. *arXiv preprint arXiv:2405.02134*, 2024.

Stephen A. Ross. The arbitrage theory of capital asset pricing. *Journal of Economic Theory*, 13(3): 341–360, 1976. doi: 10.1016/0022-0531(76)90046-6.

Stephen A. Ross. A simple approach to the valuation of risky streams. *The Journal of Business*, 51 (3):453–475, 1978. doi: 10.1086/296008.

Eden Saig, Ohad Einav, and Inbal Talgam-Cohen. Incentivizing quality text generation via statistical contracts. In *Advances in Neural Information Processing Systems*, volume 37, 2024.

Marija Šakota, Maxime Peyrard, and Robert West. Fly-swat or cannon? cost-effective language model choice via meta-modeling. In *Proceedings of the 17th ACM International Conference on Web Search and Data Mining*, pp. 606–615, 2024.

Rylan Schaeffer, Joshua Kazdan, John Hughes, Jordan Juravsky, Sara Price, Aengus Lynch, Erik Jones, Robert Kirk, Azalia Mirhoseini, and Sanmi Koyejo. How do large language monkeys get their power (laws)? In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 53132–53176. PMLR, 2025. URL https://proceedings.mlr.press/v267/schaeffer25a.html.

Andrei Shleifer. *Inefficient Markets: An Introduction to Behavioral Finance*. Oxford University Press, Oxford, 2000.

Tal Shnitzer, Anthony Ou, Mírian Silva, Kate Soule, Yuekai Sun, Justin Solomon, Neil Thompson, and Mikhail Yurochkin. Large language model routing with benchmark datasets. In *First Conference on Language Modeling*, 2024.

Shivalika Singh, Angelika Romanou, Clémentine Fourrier, David Ifeoluwa Adelani, Jian Gang Ngui, Daniel Vila-Suero, Peerat Limkonchotiwat, Kelly Marchisio, Wei Qi Leong, Yosephine Susanto, et al. Global mmlu: Understanding and addressing cultural and linguistic biases in multilingual evaluation. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 18761–18799, 2025.

Charlie Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. Scaling LLM test-time compute optimally can be more effective than scaling parameters for reasoning. In *The Thirteenth International Conference on Learning Representations*, 2025.

Guoheng Sun, Ziyao Wang, Bowei Tian, Meng Liu, Zheyu Shen, Shwai He, Yexiao He, Wanghao Ye, Yiting Wang, and Ang Li. CoIn: Counting the invisible reasoning tokens in commercial opaque LLM APIs. *arXiv preprint arXiv:2505.13778*, 2025. doi: 10.48550/arXiv.2505.13778.

Sang Truong, Yuheng Tu, Rylan Schaeffer, and Sanmi Koyejo. Item response scaling laws: A measurement theory approach for efficient and generalizable neural scaling estimation. In *Proceedings of the 43rd International Conference on Machine Learning*, volume 306 of *Proceedings of Machine Learning Research*, 2026. URL https://arxiv.org/abs/2606.07616.

Hal R. Varian. The arbitrage principle in financial economics. *Journal of Economic Perspectives*, 1 (2):55–72, 1987. doi: 10.1257/jep.1.2.55.

Neeraj Varshney and Chitta Baral. Model cascading: Towards jointly improving efficiency and accuracy of NLP systems. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pp. 11007–11021, 2022.

Guangya Wan, Zixin Stephen Xu, Sasa Zorc, Manel Baucells, Mengxuan Hu, Hao Wang, and Sheng Li. BEACON: Bayesian optimal stopping for efficient LLM sampling. *arXiv preprint arXiv:2510.15945*, 2025. doi: 10.48550/arXiv.2510.15945.

Junlin Wang, Siddhartha Jain, Dejiao Zhang, Baishakhi Ray, Varun Kumar, and Ben Athiwaratkun. Reasoning in token economies: Budget-aware evaluation of llm reasoning strategies. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 19916–19939, 2024a.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models. In *The Eleventh International Conference on Learning Representations*, 2023a. doi: 10.48550/arXiv.2203.11171.

Yiding Wang, Kai Chen, Haisheng Tan, and Kun Guo. Tabi: An efficient multi-level inference system for large language models. In *Proceedings of the Eighteenth European Conference on Computer Systems*, pp. 233–248, 2023b.

Yubo Wang, Xueguang Ma, Ge Zhang, Yuansheng Ni, Abhranil Chandra, Shiguang Guo, Weiming Ren, Aaran Arulraj, Xuan He, Ziyan Jiang, Tianle Li, Max Ku, Kai Wang, Alex Zhuang, Rongqi Fan, Xiang Yue, and Wenhu Chen. MMLU-Pro: A more robust and challenging multi-task language understanding benchmark. In *Advances in Neural Information Processing Systems 37, Datasets and Benchmarks Track*, 2024b.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed H. Chi, Quoc V. Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems*, volume 35, pp. 24824–24837, 2022.

Wenhan Xiong, Jingyu Liu, Igor Molybog, Hejia Zhang, Prajjwal Bhargava, Rui Hou, Louis Martin, Rashi Rungta, Karthik Abinav Sankararaman, Barlas Oguz, et al. Effective long-context scaling of foundation models, 2023. *URL https://arxiv. org/abs/2309.16039*.

A Additional examples

Example 1. *Suppose we have two models and the query space is partitioned into two equally likely disjoint regions $\mathcal{X} = \mathcal{X}_1 \sqcup \mathcal{X}_2$, with $D(\mathcal{X}_i) = 1/2$. Model i performs better on $\mathcal{X}_i$, and its utility is*

$$\Pr[A(x, y) = 1] = \begin{cases} 1 - e^{-ac}, & x \in \mathcal{X}_i, \\ 1 - e^{-bc}, & x \notin \mathcal{X}_i, \end{cases} \qquad a > b > 0.$$

Thus, every model's performance increases with its allocated budget on every query, but it improves faster on queries in its specialty. Assume further that the models' outputs are independent conditional on the query. Allocating the entire budget C to either model gives the average acceptance probability of:

$$\Pr_{x \sim D, y \sim p_i(.|x,C)}[A(x, y) = 1] = 1 - \frac{e^{-aC} + e^{-bC}}{2}.$$

In contrast, if we allocate $c_i = C/2$ to every model, for every query we have that

$$\Pr[A(x, y_1) = 1 \text{ or } A(x, y_2) = 1] = 1 - e^{-\frac{C}{2}(a+b)} > 1 - \frac{e^{-aC} + e^{-bC}}{2},$$

where the strict inequality follows from the arithmetic–geometric mean inequality. Hence, at the same spend C on every query, a diversified portfolio can strictly outperform every single model.

B Proofs

Theorem 1. *Finding the optimal arbitrage policy as in Equation* (2) *is NP-hard.*

Proof. We use a standard reduction from Maximum Coverage (MC), which is NP-hard (Karp, 1972). In MC, given a collection of sets $S := (S_1, \ldots, S_N)$, the objective is to select at most B sets such that the number of covered elements of $\mathcal{X}$ is maximized.

Given an instance of MC, let D be a uniform distribution over $\mathcal{X}$. For each set S_i, we construct model i such that

$$\Pr_{y_i \sim p_i(\cdot|x,c_i)}[A(x, y_i) = 1] = \mathbf{1}\{x \in S_i\}\mathbf{1}\{c_i \geq 1\},$$

and we set $C = B$.

Let $\boldsymbol{c} := (c_1, \ldots, c_N)$ be an optimal solution to our problem. Let $I(\boldsymbol{c}) := \{i : c_i \geq 1\}$. Model i succeeds deterministically on $x \in S_i$ whenever a budget of at least 1 is assigned to it, and never succeeds outside S_i. Therefore, the portfolio succeeds on x exactly when $x \in \bigcup_{i \in I(\boldsymbol{c})} S_i$, hence we have that

$$U(\boldsymbol{c}) = \frac{|\bigcup_{i \in I(\boldsymbol{c})} S_i|}{|\mathcal{X}|}.$$

Now for any such optimal solution to our problem we construct a solution to MC where a set S_i is selected whenever $i \in I(\boldsymbol{c})$. Note that since each $i \in I(\boldsymbol{c})$ receives at least one unit of budget, $|I(\boldsymbol{c})| \leq \sum_{i \in I(\boldsymbol{c})} c_i \leq \sum_{i \in [N]} c_i \leq B$.

Conversely, given a solution to MC, we construct a solution to our problem by setting $c_i = 1$ whenever S_i is selected and $c_i = 0$ otherwise. Our utility equals the fraction of elements covered by the selected sets, thus maximizing $U(\boldsymbol{c})$ is equivalent to MC up to the constant factor $\frac{1}{|\mathcal{X}|}$. Given both directions of the proof, our problem is NP-hard. □

Theorem 2. *Under Assumption 1, U is non-decreasing and jointly concave. If D has finite support and each $\alpha_i(x; \cdot)$ is differentiable, a global optimum exists and is characterized by the KKT conditions. This optimum can be found efficiently using coordinate-ascent on the Lagrangian.*

Proof. We first show that under Assumption 1 the objective is monotone and jointly concave over the budget simplex $\Delta := \{\boldsymbol{c} \in \mathbb{R}^N_{\geq 0} : \sum_{i \in [N]} c_i \leq C\}$. As a result, every KKT point is a global optimum.

We start by proving monotonicity and joint concavity. Recall that $\alpha_i(x; c) := -\log(1 - u_i(x; c))$, and that we can rewrite the objective function in Equation (2) as:

$$U(\boldsymbol{c}) = \mathbb{E}_{x \sim D}[1 - \exp(-\sum_{i=1}^{N} \alpha_i(x; c_i))],$$

where by assumption, each $\alpha_i(x; c_i)$ is non-decreasing and concave, hence, $\sum_{i=1}^{N} \alpha_i(x; c_i)$ is jointly concave in $\boldsymbol{c}$. Furthermore, $g \mapsto 1 - \exp(-g)$ is strictly increasing and concave in g, and hence the composition $1 - \exp(-\sum_{i=1}^{N} \alpha_i(x; c_i))$ is also jointly concave in $\boldsymbol{c}$. Taking expectation preserves concavity, hence, $U(\boldsymbol{c})$ is jointly concave in $\boldsymbol{c}$. Monotonicity is immediate from each $\alpha_i(x; c_i)$ being non-decreasing. Since each $\alpha_i(x, \cdot)$ is differentiable, the portfolio utility for every fixed x,

$$1 - \exp\left(-\sum_{i=1}^{N} \alpha_i(x, c_i)\right)$$

is continuous in $\boldsymbol{c}$. Since D has finite support, $U(\boldsymbol{c})$ is a finite weighted sum of these continuous functions and is therefore continuous. Since Δ is compact, the extreme value theorem guarantees that U attains a global maximum on Δ.

The feasible set Δ is convex and described by affine constraints. For $C > 0$, the allocation $c_i = C/(2N)$ is strictly feasible, hence Slater's condition holds. Therefore, the KKT conditions are both necessary and sufficient (Boyd & Vandenberghe, 2004). As a result, a global maximizer is characterized by the existence of $\lambda \geq 0$ such that $\lambda(C - \sum_{i=1}^{N} c_i^*) = 0$, and for all $i \in [N]$,

$$\frac{\partial U(\boldsymbol{c}^*)}{\partial c_i} = \mathbb{E}_{x \sim D}\left[\underbrace{\frac{\partial \alpha_i(x, c_i^*)}{\partial c_i}}_{\substack{\text{Marginal log-failure reduction} \\ \text{purchased per dollar for } x}} \underbrace{e^{-\sum_{\ell=1}^{N} \alpha_\ell(x, c_\ell^*)}}_{\substack{\text{Probability of failure of} \\ \text{the model pool on } x}}\right] \begin{cases} = \lambda, & c_i^* > 0, \\ \leq \lambda, & c_i^* = 0. \end{cases} \tag{9}$$

In practice, we maximize $U(\mathbf{c}) - \lambda \sum_i c_i$ over $\mathbf{c} \geq 0$ using coordinate ascent, adjusting the shadow price λ to meet the budget constraint.

□

Proposition 1. *For a fixed allocation* $\mathbf{c}$ *and instance* x*, the order* π *that minimizes* $R^\pi(x;\mathbf{c})$ *queries the models in decreasing order of* $\frac{u_i(x;c_i)}{R_i(x,c_i)}$.

Proof. Consider two models a, b occupying adjacent positions in π, reached with common probability ρ (all earlier models failed), and abbreviate $u_a := u_a(x;c_a)$ and $R_a := R_a(x;c_a)$. Their contribution to $R^\pi(x;\boldsymbol{c})$ is $\rho\big[R_a + (1-u_a)R_b\big]$ if a precedes b and $\rho\big[R_b + (1-u_b)R_a\big]$ otherwise. Every position before the pair is unchanged, and every position after the pair sees the same combined survival probability $(1-u_a)(1-u_b)$ regardless of the pair's internal order, so a swap changes only this pair's contribution. Placing a first is weakly better iff $R_a + (1-u_a)R_b \le R_b + (1-u_b)R_a$, i.e. iff $R_a u_b \le R_b u_a$, or equivalently $\frac{u_a}{R_a} \ge \frac{u_b}{R_b}$. So by an exchange argument the total cost is minimized by sorting the models in decreasing order of their expected utility per cost on every query. □

Theorem 3. *Finding a fixed deployment order that minimizes* $\mathbb{E}_{x\sim D}[R^\pi(x,\mathbf{c})]$ *is NP-hard.*

Proof. We reduce from minimum-sum set cover. Let $\mathcal{X} = \{x_1,\dots,x_m\}$ be a universe and let $\mathcal{S} = \{S_1,\dots,S_N\}$ be a collection of sets whose union is $\mathcal{X}$. For a permutation π of the sets, define the cover time of element x by

$$T_\pi(x) := \min\{t : x \in S_{\pi(t)}\}.$$

The minimum-sum set-cover problem asks for a permutation minimizing

$$\sum_{x\in\mathcal{X}} T_\pi(x),$$

and is NP-hard (Feige et al., 2004).

Construct a model-ordering instance with one query for every $x \in \mathcal{X}$, drawn uniformly, and one model p_i for every set S_i. Set $q_i(x) = c_i = 1$ and let model p_i succeed deterministically exactly on the queries contained in S_i:

$$p_i(x) = \begin{cases} 1, & x \in S_i, \\ 0, & x \notin S_i. \end{cases}$$

We interpret the endpoint cases in the survival-function formulation directly. If $x \notin S_i$, then

$$R_i(x;1) = \int_0^1 (1-p_i(x))^t \,\mathrm{d}t = 1,$$

whereas if $x \in S_i$, then

$$R_i(x;1) = \int_0^1 (1-p_i(x))^t \,\mathrm{d}t = 0.$$

For a fixed query x, every model preceding the first set containing x fails and contributes one unit of cost. The first model whose set contains x succeeds and terminates the cascade. Consequently,

$$R^\pi(x;\mathbf{c}) = T_\pi(x) - 1.$$

Taking the expectation under the uniform query distribution gives

$$\mathbb{E}_{x\sim D}[R^\pi(x;\mathbf{c})] = \frac{1}{m}\sum_{x\in\mathcal{X}} T_\pi(x) - 1.$$

Therefore, an algorithm that computed the optimal model order would also solve minimum-sum set cover. Hence, choosing the cost-minimizing fixed model order is NP-hard. □

C Additional details on utility estimation methods

Sampling strategy to estimate pass@k. An unbiased and stable estimator of $\mathrm{pass}_{i,x}\,@k$ was proposed by Chen et al. (2021): $\widehat{\mathrm{pass}_{i,x}\,@k} := 1 - \frac{\binom{n_{i,x}-s_{i,x}}{k}}{\binom{n_{i,x}}{k}}$, where $n_{i,x}$ is the total number of attempts of model i for query x and $s_{i,x}$ is the number of successful such attempts; note that the estimator above requires $n_{i,x} \ge k$ attempts, for large values of k, it requires many generations per problem

which is costly. Therefore, in the next part we adopt a distributional strategy (Schaeffer et al., 2025; Kazdan et al., 2025) to estimate $\mathrm{pass}_{i,x}\,@k$.

Distributional strategy to estimate pass@k. Since attempts are i.i.d., $\mathrm{pass}_{i,x}\,@k$ can be written as $\mathrm{pass}_{i,x}\,@k = 1-(1-\mathrm{pass}_{i,x}\,@1)^k$. Hence, to estimate $\mathrm{pass}_{i,x}\,@k$, it suffices to estimate $\mathrm{pass}_{i,x}\,@1$. Schaeffer et al. (2025) propose a distributional strategy estimator where they first select a prior distribution on $\mathrm{pass}_{i,x}\,@1$, they discretize this distribution according to the sampling resolution, and fit the distribution's parameters using MLE under the discretized distribution's PMF. Kazdan et al. (2025) further show that this discretization introduces bias and propose a strategy which does not require discretization.

D Additional Experimental Details

D.1 Additional details for reproducing Figure 1

We use Terminal-Bench 2.0,[6] a benchmark of 88 shared agentic software engineering tasks. Of its submissions, 75 agent–model systems publish per-trial artifacts.[7] We keep the 28 that record either a per-trial dollar cost or usable token counts. Each model–query pair has at most 5 recorded attempts with binary outcomes. We estimate the mean cost per attempt for all model–query pairs using the reported cost per trial when available, and otherwise the recorded input, output, and cached token counts scaled by the LiteLLM price database as of September 2026. [8]

We estimate the success probabilities $u_i(x, c_i)$ under repeated sampling using the unbiased estimator of Chen et al. (2021) within the recorded attempts and from the empirical Bayes estimator of Section 4 beyond that. For every model–query pair, we find the expected spend under repeated sampling that stops upon success using the survival function $\int_0^{c_i}(1 - u_i(x; t))\,\mathrm{d}t$. We use the same budget for each query, find the probability that the query is solved using that budget, find the average solve rate over the dataset, and increase the budget until the solve rate reaches the target. Budgets are swept over 600 log-spaced points from $\$10^{-4}$ to $\$10$ per task.

D.2 Additional details on benchmarks in Section 5.1

For details on *Terminal-Bench 2.0* refer to Appendix D.1.

LiveCodeBench[9] (Jain et al., 2025) consists of 73 models on 713 shared problems, however only 36 models record more than one attempt out of which 10 models are reasoning models which do not record reasoning traces so their token cost is not recoverable. We are hence left with 24 models. Each attempt is treated as a separate request and its input is billed without caching. We use the model-specific tokenizer when publicly available and `cl100k_base`[10] as a proxy otherwise. In this regard note that Terminal-Bench trials are multi-turn agent runs whose billed context is not recoverable from the stored logs, so systems that report neither cost nor token counts are excluded rather than priced based on proxy tokenizers. Prices are taken from the same dated LiteLLM and OpenRouter snapshots as TerminalBench; historical list prices are used for models absent from both snapshots.

D.3 Results on additional benchmarks

We test whether our findings in Section 5.1 holds on two additional agentic markets with recorded per-attempt dollar costs.

Terminal-Bench 4.0[11] provides 12 agent–model systems on 66 shared tasks with 5 recorded attempts per (system, task). Here the leaderboard records the dollar cost of every attempt directly, so no token

[6] Terminal-Bench 2.0
[7] Terminal-Bench 2.0 leaderboard submission
[8] LiteLLM price database snapshot, September 17, 2026
[9] LiveCodeBench leaderboard submissions
[10] `cl100k_base` BPE tokenizer
[11] Terminal-Bench 4.0 leaderboard

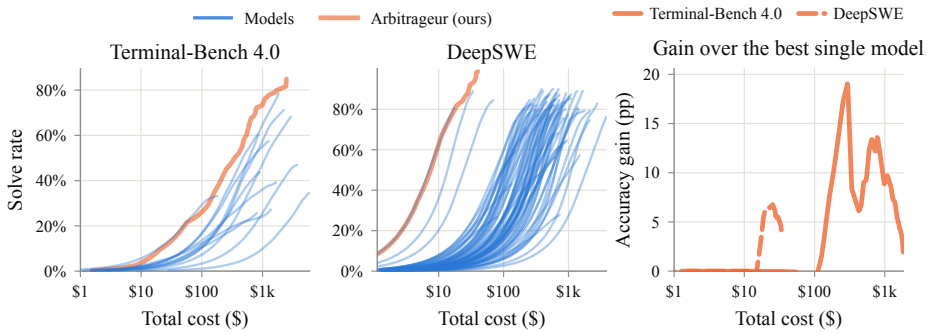


Figure 6: **Arbitrage frontier on Terminal-Bench 4.0 and DeepSWE.** Solve rate against total cost on Terminal-Bench 4.0 (12 systems, 66 tasks) and DeepSWE (70 configurations, 111 tasks). Each blue curve is one system funded on its own by repeated sampling with early stopping; the arbitrageur's allocation is cross-validated leave-one-out over the tasks and deployed in the greedy population order. The rightmost panel is the gain in solve rate over the best single system at the same total cost, in percentage points. Interestingly on DeepSWE, one single model, the cheapest one, is the most cost effective.

pricing is needed; one submission whose attempts all report a cost of zero is excluded, as a free arm would be infinitely cheap to the arbitrageur.

DeepSWE[12] provides 70 (model, reasoning-effort) configurations of 28 base models run under a common agent harness on 111 shared software-engineering tasks, with 4 attempts per pair (a handful of pairs have fewer) and a recorded dollar cost per attempt. As in the main text, every configuration is a distinct arm.

Figure 6 shows the cost–solve-rate frontiers and Figure 7 shows the revenue shares. The results of Section 5 carry over: on both markets the arbitrageur reaches every solve rate at least as cheaply as any single system and solves more at any budget. On Terminal-Bench 4.0 the gain over the best single system peaks at roughly 20 percentage points, and revenue spreads from the cheapest system at low budgets to several providers as the budget grows. DeepSWE is a more concentrated market: a single cheap model is the most cost-effective over most of the budget range, so the arbitrageur tracks it there, funds a more expensive model of the same provider only at the highest budgets, and the gain over the best single system is correspondingly smaller (up to about 8 percentage points).

Figure 8 compares the realized spend of our greedy order proposed in Section 3 with the best fixed order of the funded models, computed by exact dynamic programming over the funded models (which remains exponential in the number of funded models), and with the common heuristic of querying models cheapest first. The greedy order stays within 10% of the best order everywhere on TerminalBench, whereas cheapest-first order spends up to 16–17% more. On DeepSWE the arbitrageur funds at most two models and both rules recover the best order at every budget.

D.4 Experimental details of utility curve estimation

For the Monkey Business benchmarks the dollar cost of an attempt is not available so we use the number of FLOPs of one generation as the compute proxy which is calculated as the model's parameter count times the mean number of tokens it generates for that question Kaplan et al. (2020). For the remaining benchmarks (AIME 2024-2025, Global MMLU Lite, and MMLU-Pro), the released data contains only the binary outcome of each attempt, not the generated text, so the length of a generation, and with it the cost of an attempt, cannot be recovered per question. We therefore use a compute proxy that is constant across questions for each model, its number of active parameters, which is proportional to the operations per generated token.

We report the estimation error of our method on all six benchmarks (Figure 9) and repeat the evaluation of Figure 5 on Global-MMLU-Lite (Singh et al., 2025) and MMLU-Pro (Wang et al., 2024b).

[12] DeepSWE v1.1 public trials

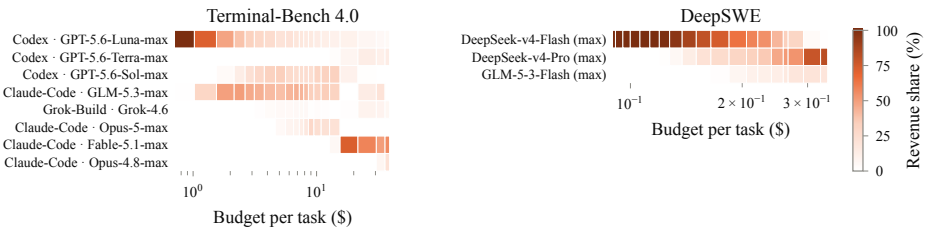


Figure 7: **Revenue spreads across providers as budgets increase.** Revenue share of each funded provider as a function of the budget per task under the arbitrageur's cross-validated allocation of Figure 6, with providers sorted by per-attempt price (cheapest on top). At lower budgets, revenue goes entirely to the cheaper models; as the budget grows, it spreads among more providers, and even at high budgets cheap models retain a significant share of the revenue. In DeepSWE, the market remains concentrated on several cheap models that offer the highest utility per cost.

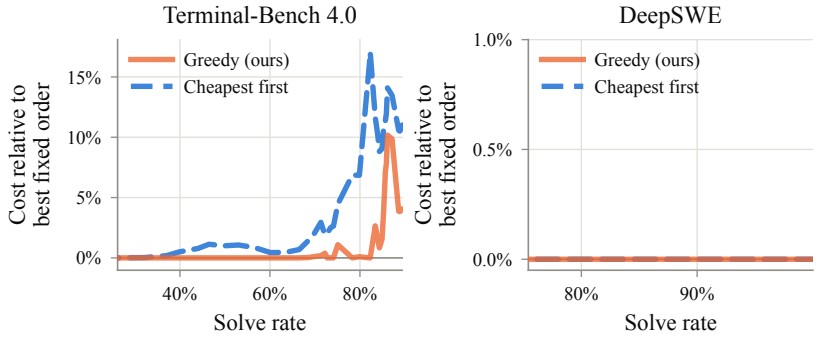


Figure 8: **The greedy deployment order is close to the best fixed order on every market.** Expected spend of the arbitrageur's allocation when the funded models are queried in the greedy order of Equation (7) (solid) or in increasing order of average cost per attempt (dashed), relative to the best fixed order of the funded models found by exact dynamic programming over the funded models (which remains exponential in the number of funded models, the 0% line), as a function of the solve rate of the allocation. All quantities are computed in sample on all problems, so the best order is a floor and the figure measures the optimality gap of the two rules. Both orders coincide with the best order while a single model is funded; the axis starts at the first allocation that funds two or more models. The greedy order matches stays within 10% of the best order everywhere on TerminalBench, whereas cheapest-first order spends up to 16–17$\%$ more. On DeepSWE the arbitrageur funds at most two models and both rules recover the best order at every budget.

Results are in Figure 10. For both, we use the 30 questions per benchmark and the $\sim 2{,}500$ recorded binary attempts per model–question pair released by Truong et al. (2026) for 12 open-weight reasoning models. As these models are self-hosted, the per-attempt cost is a compute proxy equal to the model's active parameter count; as only the cost ratios matter to the arbitrageur as scaling every cost by the same constant rescales the budget without changing the arbitrageur's allocation.

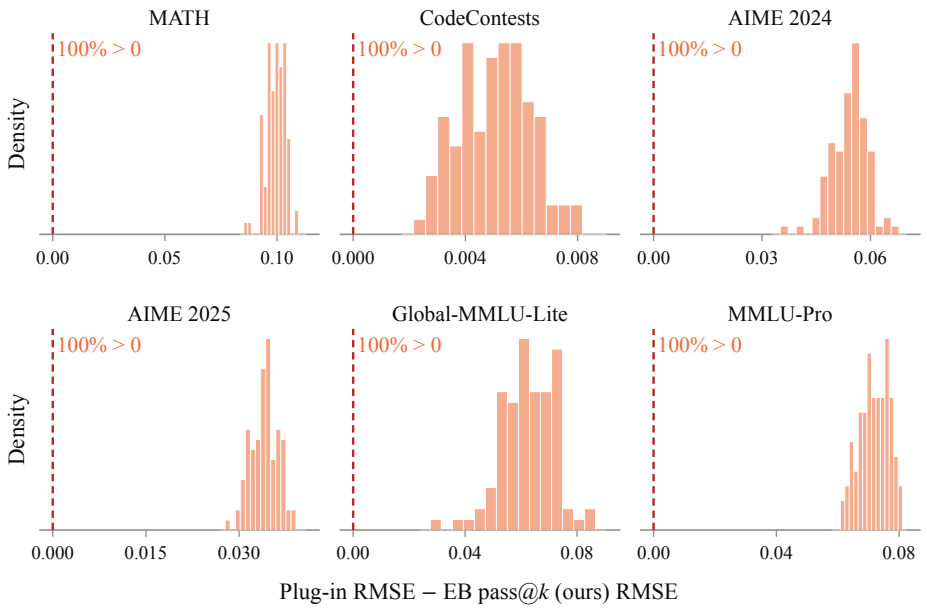


Figure 9: **Our estimator consistently outperforms the plug-in estimator.** Distribution over 100 random calibration draws of the paired gap in pass@k RMSE against the Oracle reference, averaged over models and $k \leq 500$, with $n = 8$ attempts per query–model pair, on MATH, CodeContests, AIME 2024, AIME 2025, Global-MMLU-Lite and MMLU-Pro; the two estimators are fitted on the same subsample of each draw. The dashed line marks no difference; mass to its right means our estimator is more accurate on that draw, and the annotation gives the share of such draws. The gap is positive on every draw on all benchmarks showing that our estimator consistently has lower RMSE than the plug-in estimator. This suggests that pooling information across queries (through the model-level prior) substantially improves utility estimation in the low-data regime.

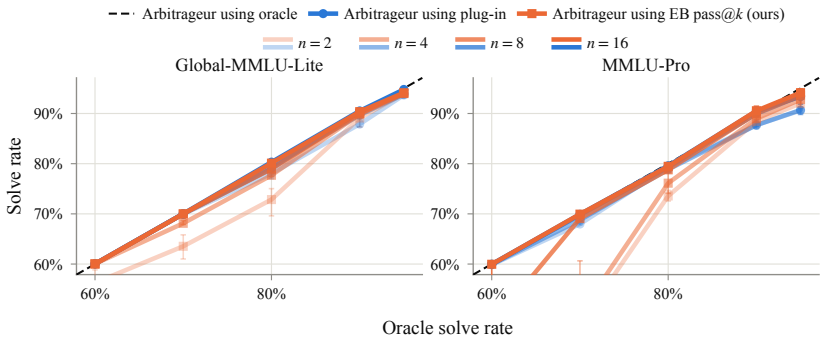


Figure 10: **An arbitrageur using our estimation method tracks the solve rate of the one using the ground truth utility curves which uses several orders of magnitude more calibration data.** On each of the MMLU benchmarks, we subsample $n = \{2, 4, 8, 16\}$ attempts per query–model pair, re-estimate the utility curves, run the arbitrage on the estimates and plot the achieved solve rate; the oracle uses all the 2,500 recorded attempts and is the diagonal. For each Oracle solve rate we find the budget that the Oracle needs to achieve that solve rate and use the same budget for other method.