Contracting for LLM Delegation: Moral Hazard in Technology and Effort Choice

Nanda Kishore Sreenivas, Kate Larson
University of Waterloo
{nksreenivas, kate.larson}@uwaterloo.ca

Abstract

We extend the standard Principal-Agent framework to scenarios where the Agent selects from a suite of technologies, each characterized by a distinct cost-capability profile. This framework is increasingly critical in the era of Large Language Models (LLMs), where Agents choose both a model and an associated effort level (e.g., token budget). We model the relationship between output quality and effort as a concave, saturating function, which depends on the Agent's hidden two-dimensional action choice balancing technology selection and effort allocation. We derive the optimal linear contract for the Principal, demonstrating that the Agent's best response is characterized by a threshold reward share that triggers technology switching. Finally, we calibrate our model using open-weight LLM pairings across the MATH and MMLUPro benchmarks. We show that both Principal and Agent, when employing bandit algorithms to navigate this environment, converge to strategies that closely align with our theoretical equilibrium. These results suggest that simple linear contracts can effectively incentivize complex, technology-aware delegation in agentic workflows.

1 Introduction

The rapid capability improvements in large language models (LLMs) in recent years have fundamentally changed how tasks are executed: from reactive, single-prompt chat interfaces toward delegation to autonomous agents. Principals, whether individual users or firms, now routinely hand off complex, open-ended tasks to specialized AI Agents. This is evident in the rise of both generalized and domain-specific tools, e.g., legal firms delegate due diligence to systems like Harvey [13], software teams assign bug fixes to autonomous coding agents like Devin [8], etc. In these interactions, the Agent operates as a black box; the Principal provides the objective and receives the final output, completely blind to the internal reasoning process and choices.

As this ecosystem matures, delegation will increasingly occur not just from humans to AI, but from AI to AI across open 'agentic markets'. Orchestrator algorithms will dynamically delegate specialized sub-tasks to agents to minimize computation costs or leverage domain expertise [12, 25]. In these agent-oriented markets, the two transacting parties possess distinct economic identities and potentially misaligned utilities [22].

This economic reality exposes an issue with the current standard of "pay-per-token" or pay-for-compute API pricing. When an autonomous Agent is billed based on its computational effort, it creates a severe moral hazard. As Bu and Ma note [7], token-based pricing under-incentivizes hidden effort and misaligns the Agent's objectives with those of the Principal. An Agent paid per token is incentivized to maximize verbosity, unnecessarily "overthink" simple problems, or covertly utilize cheaper, lower-capability underlying models to save on its own costs [26, 24].

Consequently, delegating a task to an autonomous Agent increasingly means delegating a choice of *which* tool to use, not just how hard to work. This is a moral hazard problem, but not one the classical Principal-Agent literature typically addresses. We formalize this with a linear contract (parametrized by $\alpha \in [0, 1]$) between a Principal and an Agent who selects a model m and a token budget x, where output quality follows a saturating, diminishing-returns production function in x which is a natural fit for LLM inference. Under this model, we characterize the Agent's best response as a threshold in the linear contract at which the Agent's optimal model choice switches between models, and we also derive the Principal's optimal linear contract. We calibrate the production function on six pairings of open-weight models spanning three model families, across two task domains (MATH and MMLUPro), and show that a Principal and Agent using simple online learning algorithms converge to contracts and best responses close to our theoretical predictions.

2 Related Work

Our work sits at the intersection of several areas: classical and algorithmic contract theory, anytime algorithms, inference-time compute allocation, and LLM mechanism design.

The classical **principal-agent model** [16] studies how a Principal can incentivize an Agent whose action or effort is hidden. Algorithmic contract theory [10] extends this to discrete, combinatorial action spaces and multiple outcomes, but treats the technology matrix (the action-outcome probabilities) as exogenous and fixed. A related thread studies online learning of contracts focusing on questions of learnability and regret, typically over discrete action and outcome spaces [32, 2, 3]. Existing work, including the multitask model of Holmstrom and Milgrom [15], decomposes a single technology into multiple effort or task dimensions; none lets the Agent choose *which* technology governs the effort-to-outcome mapping.

The **anytime algorithm** literature studies systems that can be interrupted at any point, returning an output whose quality improves with computation time [33, 18]. This relationship is formalized via *performance profiles*, concave and monotonically increasing functions mapping allocated compute to expected output quality. We borrow this object to model the production function of an LLM. This literature typically treats resource allocation as a single-agent deliberation [17, 6]. Since our model uses common performance profiles from this literature, our results apply here too.

Recent LLM research investigates how LLMs can dynamically scale their **inference-time compute** based on task difficulty. AnytimeReasoner [21] trains a single model via RL to produce a usable answer at any token budget; related approaches similarly train models to regulate their own reasoning length [29, 1], and document diminishing and eventually saturating returns to additional tokens. We rely on these findings to motivate the shape of

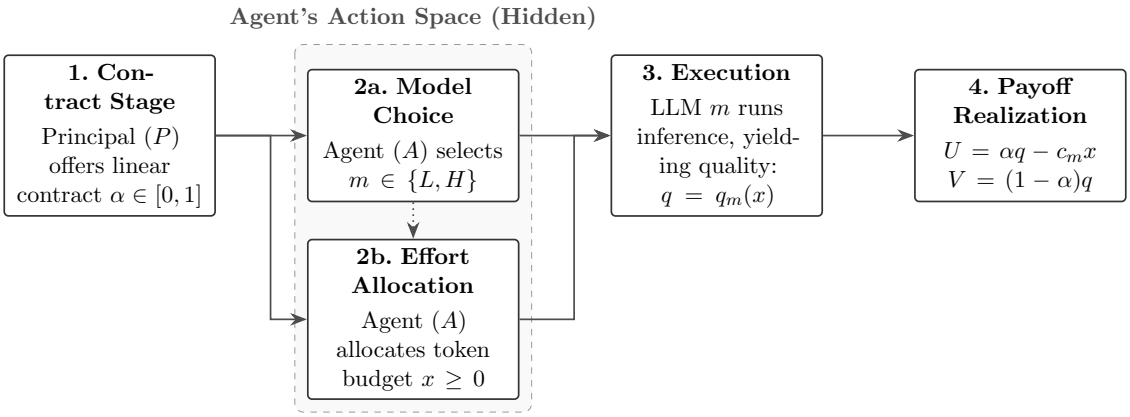


Figure 1: Principal-Agent Model with hidden model choice *and* effort.

our production curves. There are also recent frameworks for model routing with the objective of minimizing inference costs from a single-agent perspective [9, 19, 30].

Our work also relates to the emerging application of **mechanism design to LLMs**. Much of this literature focuses on the seller's side, analyzing how an LLM provider screens buyers who have different task requirements [4, 5]. Dutting *et al.* [11] explore token-level auctions to influence the output of LLMs for applications like ad generation. Contemporary work explores contracts over LLM generation, focusing on how and when the Principal should verify the Agent's work when verification is costly [24]. Closest in spirit to our application is the work of Saig *et al.* [23], who motivate 'pay-for-performance' contracts and design threshold and monotone contracts for an Agent choosing among a discrete menu of LLMs, robust to unknown Agent costs, over a discrete space of quality levels. We build on this premise of contracting over AI generation but alter the fundamental mechanics. The Agent is a task/domain specialist using publicly available models and, therefore, costs are known in our theoretical model. We also introduce an explicit continuous, unobservable token budget nested within the discrete model choice.

3 Model

A **Principal** P delegates a task to an **Agent** A. The Agent is an *anytime reasoner*: it selects a language model m from a finite set of models and allocates token budget $x \geq 0$ to inference. Output quality q is a function of m and x, and only the quality is observable by P; not the Agent's model choice or token budget. We assume that the Agent specializes in a specific task domain, and therefore, all contracted tasks are reasonably similar. The overall setup is shown in Figure 1.

The Principal offers a *linear contract*[1] with reward share $\alpha \in [0, 1]$. The Agent receives $\alpha \cdot q$ and the Principal retains $(1 - \alpha) \cdot q$. The contract is over output q only; model choice and token count are not directly observable, and therefore, not contractible.

Let $\mathcal{M}$ be the set of available models, and models are arranged in increasing order of

[1]We explore fixed payments in Appendix C and show that, under limited liability, it is equivalent to this reward sharing model.

capability in the task domain. We generalize to multiple models in Appendix H. For clarity, we present the two-model scenario with $\mathcal{M} = \{L, H\}$, where L represents the cheaper model with lower capability and H denotes a more capable and more expensive model. For example, L could be a smaller model that is optimized for edge applications, while H could be an expensive, slower frontier model. Alternatively, L could be a simple, instruction-tuned chat model while H could have other capabilities such as reasoning, tool calls, etc. Each model $m \in \{L, H\}$ is characterized by a set of performance and cost parameters.

We assume linear cost per token, and the output q is measured in terms of accuracy (as a percent, $q \in [0, 100]$)[2].

The Agent's two decisions are:

- **Model choice:** $m \in \{L, H\}$.
- **Effort:** token count $x \geq 0$.

The utilities of the Agent and Principal are given by:

$$\bar{U} = \alpha \cdot v \cdot q_m(x) - \bar{c}_m x, \quad \bar{V} = (1 - \alpha) \cdot v \cdot q_m(x).$$

where m is the model choice and $\bar{c}_m$ is the true cost per token. We assume that the monetary return to the Principal scales linearly with accuracy by a factor v. We normalize utilities by dividing by the Principal's valuation parameter v throughout[3] to get:

$$\begin{aligned} U(\alpha, m, x) &= \alpha \cdot q_m(x) - c_m x, \\ V(\alpha, m, x) &= (1 - \alpha) \cdot q_m(x). \end{aligned} \tag{1}$$

where c_m denotes the normalized cost per token for model m, *i.e.*, $c_m = \bar{c}_m / v$. We will use these normalized utilities for the Agent and Principal in the rest of this paper.

3.1 Optimal Effort

To further analyse the Agent's optimal model choice and effort level, we model the output function $q_m(x)$ as a saturation function, where M_m is the capability ceiling (in terms of accuracy on the task) and k_m is the saturation rate of model m.

$$q_m(x) = M_m\big(1 - e^{-k_m x}\big) \tag{2}$$

Note that the function is increasing, concave, and has an asymptotic upper bound M_m, and this shape is consistent with empirical studies of LLM inference. Several papers report that accuracy stagnates and the marginal returns diminish at higher token budgets [28, 31, 21]. Beyond LLMs, the anytime algorithm literature has also used similar curves to model performance profiles [33, 6].

[2]Measuring quality of models is outside the scope of this paper.

[3]This assumes that the Principal's task valuation v is public, which may not always be true. The information asymmetry over the Principal's 'type' forces the Agent to 'screen' the Principal. We leave this for future work.

We assume that the model H is more capable (in the limit) and also costs more per token.

$$M_H > M_L \quad c_H > c_L$$

For a given model m and reward share α, the Agent solves:

$$\max_{x \geq 0} U(\alpha, m, x) \implies \max_{x \geq 0} \alpha M_m \big(1 - e^{-k_m x}\big) - c_m x.$$

The FOC gives:

$$\alpha M_m k_m e^{-k_m x^*} = c_m.$$

Solving for the optimal effort for model m, x_m^*:

$$x_m^* = \frac{1}{k_m} \ln\left(\frac{\alpha M_m k_m}{c_m}\right). \tag{3}$$

This is valid (i.e. $x_m^* \geq 0$) only when $\frac{\alpha M_m k_m}{c_m} \geq 1$. **Activation threshold** τ_m is defined as the minimum contract share α at which it is viable for the Agent to use model m to start producing tokens. In other words, for any reward share less than τ_m, the marginal cost exceeds marginal benefit for model m. It is given by:

$$\tau_m = \frac{c_m}{M_m k_m}.$$

- If $\alpha \leq \tau_m$: the Agent will not spend any tokens on reasoning, *i.e.*, $x_m^* = 0$ and $U_A = 0$.
- If $\alpha > \tau_m$: $x_m^* = \frac{1}{k_m} \ln\left(\frac{\alpha}{\tau_m}\right)$.

Substituting x_m^* into Eq. 1, and for simplicity, we denote $U^*(\alpha, m, x_m^*)$ as $U_m(\alpha)$ which is given by:

$$U_m(\alpha) = M_m \left[\alpha - \tau_m - \tau_m \ln\left(\frac{\alpha}{\tau_m}\right)\right], \quad \alpha > \tau_m. \tag{4}$$

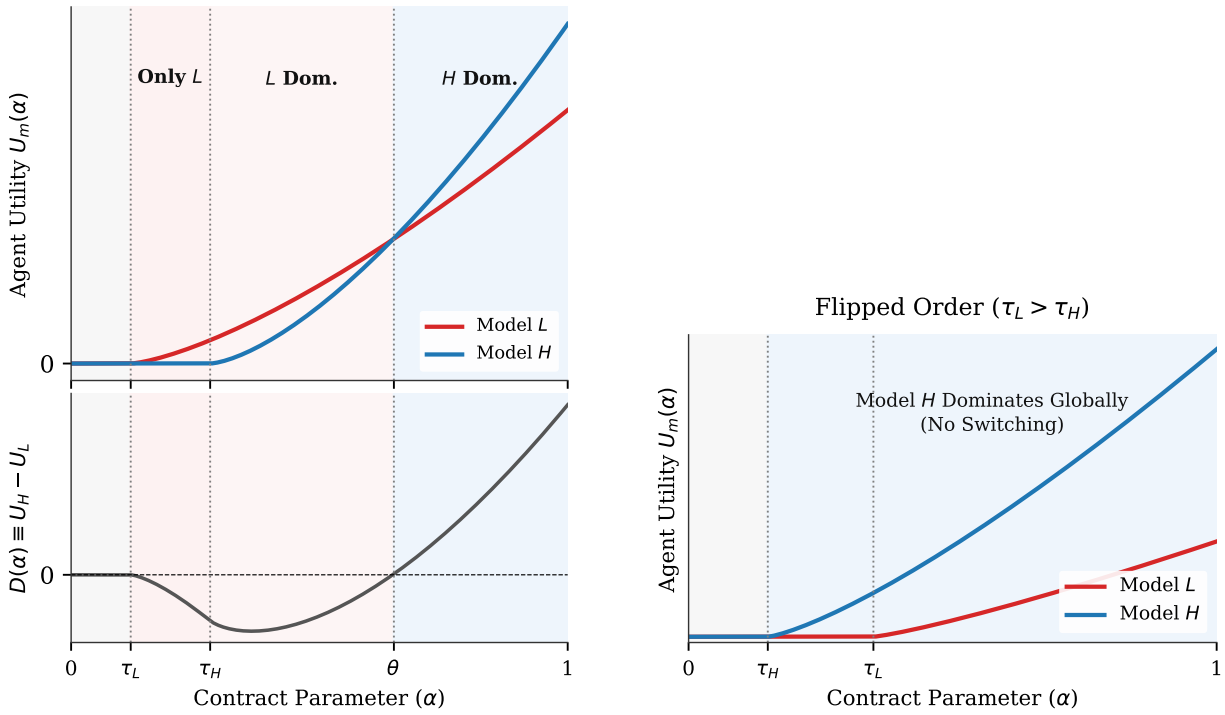
Note that $U_m(\tau_m) = M_m[\tau_m - \tau_m - \tau_m \cdot 0] = 0$. So utility is continuous at the threshold. The first derivative is:

$$U_m'(\alpha) = M_m \left[1 - \frac{\tau_m}{\alpha}\right]$$

It is positive when $\alpha > \tau_m$, and the second derivative is also positive, which implies $U_m(\alpha)$ is convex and increasing. We have now derived the optimal token budget (x_m^*) and resulting utility $U_m(\alpha)$ for the Agent using any model m given a linear contract α proposed by the Principal.

3.2 Switching Threshold θ

Next, we characterize the optimal model choice of the Agent given the linear contract α. Specifically, at what value of $\alpha = \theta$ the Agent's optimal model choice switches from one model to another. We consider two scenarios depending on which model activates first, *i.e.*, which model requires the least reward share α to start producing tokens.



(a) Standard Order $\tau_L < \tau_H$

(b) Flipped Order $\tau_L > \tau_H$

Figure 2: Illustration of model utilities for Agent $U_m(\alpha)$ under both scenarios.

Standard Order $\tau_L < \tau_H$

Model L activates earlier than H, and therefore, at τ_H, $U_L(\tau_H) > 0$, whereas model H is just activated, *i.e.*, $U_H(\tau_H) = 0$.

We define the utility differential as:

$$D(\alpha) \equiv U_H(\alpha) - U_L(\alpha) \tag{5}$$

Differentiating $D(\alpha)$ yields:

$$D'(\alpha) = M_H - M_L - \frac{M_H\tau_H - M_L\tau_L}{\alpha}$$

$$D''(\alpha) = \frac{M_H\tau_H - M_L\tau_L}{\alpha^2}$$

Under standard order ($\tau_H > \tau_L$) and the fact that $M_H > M_L$, the second derivative D'' is positive throughout. Therefore, the utility differential $D(\alpha)$ is convex. At the threshold, $D(\tau_H) = -U_L(\tau_H) < 0$. D' is also negative at this point. Due to convexity and the negative slope, as we increase α beyond τ_H, D decreases further until $D' = 0$ and then begins to increase. Therefore, there is exactly one solution θ in $[\tau_H, \infty)$ where $D(\theta) = 0$ (see Fig. 2a). The switching threshold θ exists within the permissible range $[0, 1]$ only if $D(1)$ is positive. Otherwise, model L will dominate throughout the domain.

If $D(1) > 0$, the domain of α is partitioned into 4 regions (shown in Fig. 2a in Appendix A). Neither model is viable in the first interval (until τ_L) and only L is viable in the second. Both models are viable in the third region where $\alpha > \tau_H$, but L is still more rewarding for the Agent. Beyond θ, the quality 'premium' of H is worth its higher cost, and the Agent's model choice jumps from L to H. Note that the optimal token budget is discontinuous at this point (Eq. (3)).

Flipped Order $\tau_L > \tau_H$

In this case, model L activates at a higher contract share than the higher model H. Because $\tau_H < \tau_L$, it follows that $\frac{\tau_H}{\alpha} < \frac{\tau_L}{\alpha}$, which implies:

$$\left(1 - \frac{\tau_H}{\alpha}\right) > \left(1 - \frac{\tau_L}{\alpha}\right)$$

Coupled with the fact that $M_H > M_L$, this implies $U'_H(\alpha) > U'_L(\alpha)\ \forall \alpha > \tau_L$. Since Model H activates earlier and climbs strictly faster at every point, model H dominates L throughout the domain, and there is no switching in this case (see Fig. 2b for illustration).

We have thus derived the optimal token budget (Eq. (3)) for each model m given a contract from the Principal, parametrized by α. Using that, we then derived the optimal model choice for the Agent, characterized by the switching threshold θ (its existence and meaning determined by the two scenarios outlined above).

3.3 Principal's Optimization

Using the two-dimensional best response of the Agent for any given contract α, we now derive the Principal's optimal linear contract. The Principal maximises $V(\alpha, m, x) = (1-\alpha) \cdot q_m(x^*_m(\alpha))$ over α, but m here is not a free choice for the Principal; rather it is determined by the Agent's best response. For a given model m, Agent's optimal token budget x^*_m is given by (3). The corresponding quality is $q_m(x^*_m) = M_m(1 - \tau_m/\alpha)$ and therefore, the Principal's utility for a given model is:

$$V_m(\alpha) = (1-\alpha)\, q_m(x^*_m) = M_m\Big[1 + \tau_m - \alpha - \frac{\tau_m}{\alpha}\Big]$$

This is the payoff the Principal would get *if* the Agent's optimal model choice would be m at share α. Differentiating,

$$\frac{dV}{d\alpha} = M_m\Big(\frac{\tau_m}{\alpha^2} - 1\Big) = 0 \implies \alpha^*_m = \sqrt{\tau_m},$$

$$\frac{d^2V}{d\alpha^2} = -\frac{2M_m\tau_m}{\alpha^3} < 0$$

So α^*_m is a maximum of V_m over $\alpha > 0$, and V_m is strictly concave, single-peaked at $\sqrt{\tau_m}$. We assume $\tau_m < 1$ for both models, so that $\sqrt{\tau_m} \in (\tau_m, 1)$ is a feasible share.

Standard order $\tau_L < \tau_H$.

If the switching threshold θ exists within $[0, 1]$, then the Agent picks L on $[\tau_L, \theta)$ and H on $[\theta, 1]$. $V_m(\alpha)$ is achievable only when restricted to the interval where m is actually the Agent's choice, and because each $V_m(\alpha)$ is single-peaked, the constrained optimum on each interval is either the peak (if it falls inside the interval) or the boundary (if the peak falls outside), i.e., clip α_m^* such that m is chosen by the Agent.

$$\alpha_L^\dagger = \min\big(\sqrt{\tau_L},\, \theta\big), \qquad \alpha_H^\dagger = \max\big(\sqrt{\tau_H},\, \theta\big). \tag{6}$$

If $\sqrt{\tau_L} \geq \theta$, the function $V_L(\alpha)$ is increasing in $[\tau_L, \theta)$ with the peak not yet attained. So, the maximum value is at the boundary, but note that the Agent switches to model H at θ. Therefore, the maximum value of V_L is the left-hand limit of $V_L(\theta)$. Symmetrically, if $\sqrt{\tau_H} \leq \theta$, $V_H(\alpha)$ is decreasing on $[\theta, 1]$ and its maximum is attained at the left endpoint θ.

The Principal's optimal contract is then

$$\alpha^* = \arg \max_{\alpha \in \{\alpha_L^\dagger,\ \alpha_H^\dagger\}} V(\alpha, m(\alpha)) \tag{7}$$

Note that if $D(1) \leq 0$, *i.e.*, the switching θ does not exist in the domain, L dominates throughout $[0, 1]$ and the Principal's problem reduces to $\alpha^* = \sqrt{\tau_L}$.

Flipped order $\tau_L > \tau_H$. Here H weakly dominates L everywhere it is active, so the Agent never chooses L and the Principal's problem reduces to the single unconstrained optimization $\alpha^* = \sqrt{\tau_H}$.

Thus, we have derived the optimal linear contract for the Principal and characterized the Agent's best response. Further derivation of the first-best benchmark, total surplus, and agency costs are in Appendix A.

3.4 Burn-in Tokens for Reasoning Models

The modeling choice of saturation function for the LLM quality is appropriate for simpler tasks with instruction-tuned models. For complex tasks, especially with reasoning models, there is a burn-in period, where accuracy is zero despite spending tokens. We denote $\tilde{x}$ to be the raw tokens spent, and b_m denotes the number of burn-in tokens for model m. Then,

$$q_m(\tilde{x}) = \begin{cases} 0, & \tilde{x} < b_m \\ M_m\big(1 - e^{-k_m(\tilde{x} - b_m)}\big), & \tilde{x} \geq b_m \end{cases} \tag{8}$$

Rewriting $x = \tilde{x} - b_m$ as the effective tokens spent by the Agent, we get the basic saturation function from earlier (2). However, the associated cost $c_m b_m$ was not considered. The true utility of the Agent is:

$$\mathcal{U}(\alpha, m, x) = \alpha \cdot q_m(x) - c_m x - c_m b_m$$

However, this extra fixed cost is a constant in terms of x, and therefore the optimal effective token x_m^* does not change from (3). Note that the optimal true token $\tilde{x}_m^*$ is shifted

by b_m. The Agent utility under a given model m now becomes:

$$\mathcal{U}_m(\alpha) = M_m\left[\alpha - \tau_m - \tau_m \ln\left(\frac{\alpha}{\tau_m}\right)\right] - c_m b_m$$

Note that the burn-in utility is simply the same $U_m(\alpha)$ as before with an additional negative term corresponding to the fixed cost due to the burn-in tokens. While the activation threshold τ_m ensured the Agent's participation in the base case, now the Agent's utility is in fact negative at τ_m due to the (fixed) burn-in cost. The *participation threshold* τ_m^P defines the lowest value of α such that the Agent breaks even using model m, *i.e.*, $\mathcal{U}_m(\tau_m^P) = 0$.

$$M_m\left[\tau_m^P - \tau_m - \tau_m \ln\left(\frac{\tau_m^P}{\tau_m}\right)\right] = c_m b_m$$

Simplifying, we get

$$\left(\frac{\tau_m^P}{\tau_m}\right) - 1 - \ln\left(\frac{\tau_m^P}{\tau_m}\right) = k_m b_m$$

Rewriting the improper fraction as z, consider $f(z) = z - 1 - \ln z$ evaluated over the active domain $z > 1$. Computing its derivatives:

$$f'(z) = 1 - \frac{1}{z} > 0, \quad f''(z) = \frac{1}{z^2} > 0$$

Thus, $f(z)$ is strictly increasing and strictly convex. Its inverse function $f^{-1}(x)$ is strictly increasing and strictly concave. This allows an exact closed-form expression of the participation threshold:

$$\tau_m^P = \tau_m \cdot f^{-1}(k_m b_m) \tag{9}$$

Switching Threshold θ_0

We rewrite the burn-in utility as $\mathcal{U}_m(\alpha) = U_m(\alpha) - c_m b_m$, and reuse the baseline differential $D(\alpha) \equiv U_H(\alpha) - U_L(\alpha)$ from (5). The burn-in switching threshold θ_0 satisfies $\mathcal{U}_H(\theta_0) = \mathcal{U}_L(\theta_0)$, i.e. $U_H(\theta_0) - c_H b_H = U_L(\theta_0) - c_L b_L$. Rearranging:

$$D(\theta_0) = c_H b_H - c_L b_L \tag{10}$$

This is not closed-form in general: D is a transcendental function of α. In practice, θ_0 is found numerically as the root of (10), via bracketed root-finding like Brent's method.

To obtain sharper qualitative results, we restrict to a special case with common k and b for both models. Under this assumption, θ_0 is a strict rightward shift of the base-model threshold θ (i.e., burn-in delays switching) in the Standard Order case ($\tau_L < \tau_H$). In the Flipped Order case ($\tau_H < \tau_L$), H continues to dominate everywhere, exactly as in the base model. See Appendix B for further details.

4 Experimental Setup

We evaluate our proposed framework empirically across two distinct domains: advanced mathematical reasoning and multi-discipline question answering. Specifically, we evaluate model pairings on the `MATH` dataset (difficulty levels 3 and 4) and the `MMLU-Pro` dataset [14, 27]. Our model suite consists of instruction-tuned models (e.g., Llama-3.2), distilled reasoning models (e.g. DeepSeek-R1-Distill-Qwen), and recent edge reasoning models (e.g. Gemma4). We consider models of different sizes and consider pairings of same and different types. In this section, we describe how we calibrate the production function and then describe the learning processes.

4.1 Calibration of Production Functions

To bridge our analytical model with real-world LLM performance, we first empirically calibrate the production parameters (M_m, k_m, b_m) for each model on both evaluation datasets. For each dataset and model, we let the model answer every question under a maximum token budget of $x_{\max}$ tokens at temperature 0. Let x_i denote the number of tokens used to answer question i. For a dense grid of budgets from 0 to $x_{\max}$, accuracy at budget x is the percentage of questions answered correctly using at most x tokens, i.e. with $x_i \leq x$ and a correct answer. This produces an empirical production curve for each model-dataset pair, to which we fit the saturating function with burn-in tokens (Eq. (8)) via nonlinear least squares (using `curve_fit` method from `scipy`). Table 1 reports the fitted parameters, along with R^2, for both tasks. Figure 3 shows the accuracy at each budget level along with the fitted curves for the Llama models in MATH (refer Fig. 7 in Appendix E for all other models and tasks).

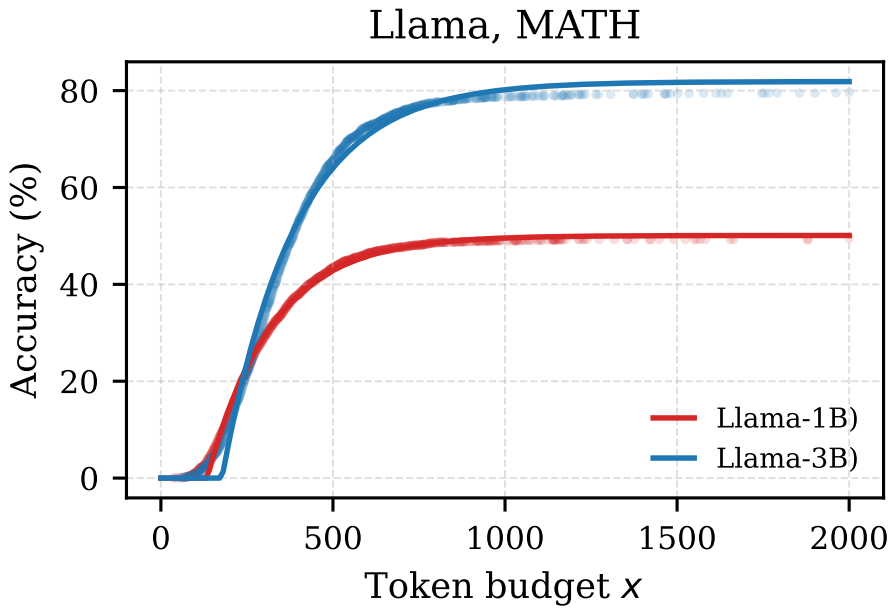


Figure 3: Accuracy *vs.* budget for Llama models on `MATH`.

Model	M_m	k_m	b_m	R^2	*RMSE*
MATH task domain					
Llama-3.2-1B-Instruct	50	0.00529	134	0.998	0.772
Llama-3.2-3B-Instruct	82	0.00473	177	0.994	2.153
DS-R1-Distill-1.5B	75	0.00081	982	0.998	0.979
DS-R1-Distill-7B	92	0.00073	1009	0.997	1.373
Gemma-4-E2B-it	84	0.00159	430	0.999	0.952
Gemma-4-E4B-it	85	0.00190	382	0.997	1.411
MMLUPro task domain					
Llama-3.2-1B-Instruct	22	0.00366	162	0.983	0.928
Llama-3.2-3B-Instruct	40	0.00474	164	0.986	1.589
DS-R1-Distill-1.5B	24	0.00212	479	0.996	0.536
DS-R1-Distill-7B	41	0.00141	440	0.999	0.414
Gemma-4-E2B-it	59	0.00202	422	0.997	1.029
Gemma-4-E4B-it	68	0.00235	413	0.998	1.084

Table 1: Model calibration and fitted params for both tasks.

We anchor the token cost of the lower-tier model to $c_L = 0.005$ monetary units per token in every pairing. This fixes a common scale and the variation between various pairings is through the capability parameters (M_m, k_m, b_m) and the cost ratio c_H/c_L. The H model's cost c_H is chosen per pairing: for most pairings we select c_H to fall within the range that yields Standard Order with a genuine switching threshold $\theta_0 \in (0, 1]$, giving representative cases of model switching; for a small number of pairings we deliberately choose c_H outside this range to illustrate the Flipped Order and always dominant L scenarios discussed in the Model Section.

4.2 Learning Processes of Agent and Principal

To evaluate how efficiently the Agent can learn the optimal mechanism without prior knowledge of the calibration parameters, we frame the model choice and budget selection problem as a Contextual Multi-Armed Bandit. At each sequential round t, the Agent receives a contract stake α_t drawn uniformly at random from the domain $\alpha_t \in [0, 1]$, which acts as the context. The task at each round t is a randomly sampled set of 16 questions from the dataset. See Appendix E for further experimental details and parameters.

The action space $\mathcal{A}$ is structured as a joint choice space $\mathcal{A} = \mathcal{M} \times \mathcal{X}$, where $\mathcal{M} = \{L, H\}$ represents model choices and $\mathcal{X} = \{x^{(1)}, x^{(2)}, \ldots, x^{(N)}\}$ is a discretized set of token budgets spanning from x_{min} up to x_{max} tokens. For a given context α_t, the net reward observed by choosing arm $a = (m, x)$ is given by $R_t(a) = \alpha_t \cdot y_{t,m}(x) - c_m \cdot x$ where $y_{t,m}(x)$ is the empirical accuracy achieved by model m under budget x (questions answered correctly by the LLM).

We deploy the LinUCB algorithm [20] to model the expected reward of each arm. Crucially, the optimal utility of the Agent $U_m(\alpha)$ maps exactly to a linear combination of α and $\ln(\alpha)$

due to the structure of the Agent's FOC; see Eq. (4). To ensure faster convergence[4], we construct a handcrafted context feature vector: $\begin{bmatrix}1 & \alpha_t & \ln(\alpha_t)\end{bmatrix}$.

The Principal's learning is modeled as classic (non-contextual) UCB, with arms representing a discretized grid of contract shares $\alpha \in [0, 1]$. Unlike the Agent, the Principal has no natural context to condition on as it chooses α rather than responding to it. Each round, the Principal selects an arm (contract share), the frozen, previously-trained Agent best-responds with its model and token-budget choice, and the Principal observes its realized payoff; $(1 - \alpha_t) \cdot y_{t,m}(x)$. This reward is used to update the estimate for the pulled arm.

5 Results

In this section, we compare the learned policies against our theoretical model with fitted parameters across different model pairings and two task domains.

5.1 `MATH` domain

First, we evaluate our model on the MATH domain [14]. The LLM prompts for question-answering are included in App. G. The theoretical value of θ_0 is calculated using the calibrated parameters in Table 1 and solving for the root of (10) using `brentq` method in `scipy` package. We train the LinUCB controller for $2,000$ episodes, the learned policy is used to greedily choose model and token budget for 50 equally interspersed values of $\alpha \in [0.001, 1]$ to obtain the learned value of θ_0. These numbers are reported in Table 2. We observe that the learned values are generally close to the estimated values (typically within about $10 - 15\%$); the difference is an artifact of mapping the continuous budget x onto a discrete token budget space ($N = 21$ bins) compounded by noisy LLM inference. Fig. 4 shows the learned token budget along with the learned switching threshold for the Llama 1B *vs.* 3B pairing. Similar figures for all other pairings are in App. E.

Model Pairing	c_H	θ_0 (est.)	θ_0 (learn)
Intra-Family Pairs			
Llama 3.2: 1B *vs.* 3B	0.0200	0.367	0.409
DeepSeek R1: 1.5B *vs.* 7B	0.0075	0.731	0.817
Gemma 4: E2B *vs.* E4B	0.0060	0.255	0.286
Inter-Family Pairs			
Llama 1B *vs.* Gemma 4B	0.0100	0.419	0.388
Llama 1B *vs.* DS 1.5B	0.0075	1.331	L dom.
DS 1.5B *vs.* Gemma 4B	0.0075	N/A	H dom.

Table 2: **MATH Domain**: Learned model choices (θ_0) for the LinUCB controller. We fix $c_L = 0.005$ for all configurations.

[4]We also explored using standard UCB with discretized α; similar trends (Fig. 11 in App. E) but takes longer to converge.

While values of c_H were mostly chosen to induce standard order model switching, we also included some other scenarios. Llama 1B *vs.* DS 1.5B is one such case where the switching threshold θ_0 lies outside the domain $[0, 1]$. Intuitively, H, which is the Deepseek model here, is significantly more capable than Llama 1B (about 50%; see Table 1). However, Deepseek has higher burn-in tokens and that combined with the higher cost makes it unattractive for any α within the domain. The LinUCB Agent learns to always pick L in this scenario.

The cost c_H for the pairing DS 1.5B *vs.* Gemma 4B was chosen to exhibit yet another interesting scenario. Here, $\tau_L = 0.0833$ and $\tau_H = 0.0464$. That is, this specific configuration belongs to the *flipped order* scenario, where H activates earlier and dominates L everywhere. There is no switching threshold θ_0 in this scenario and the LinUCB Agent matches this exactly, where H, the Gemma 4B model dominates everywhere. This is explained by the fact that Gemma models have higher capability ceiling than Deepseek R1 1.5B on this task and they have lower burn-in tokens (see calibration Table 1). So, despite the slightly higher cost, it is economically rational to always use the expensive model.

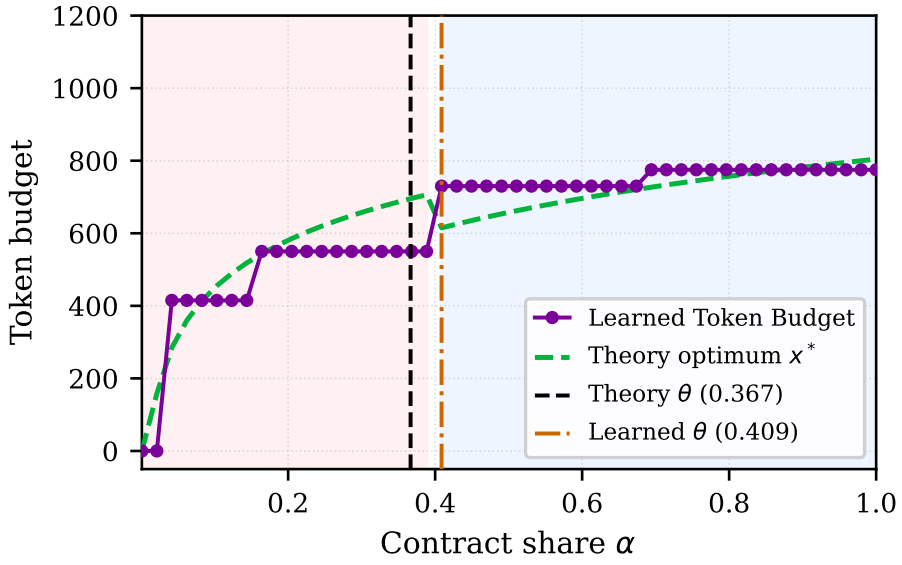


Figure 4: MATH: Llama 1B vs. Llama 3B

Instead of the LinUCB bandit controller, we also explored prompting a much stronger LLM to choose the model and token budget in Appendix F; results are broadly consistent with optimal model choice but not the optimal budget.

5.2 `MMLUPro` Domain

We also evaluate our results on the MMLUPro domain [27], and the results are shown in Table 3. Similar to the results from the MATH domain, the bandit algorithms learned the switching thresholds that are generally close to the estimated theoretical value of θ_0. Figure 5 shows the learned token budget and model choice for the DeepSeek 1.5B vs. Gemma 4B pairing (other pairings are in Fig. 9 in App. E).

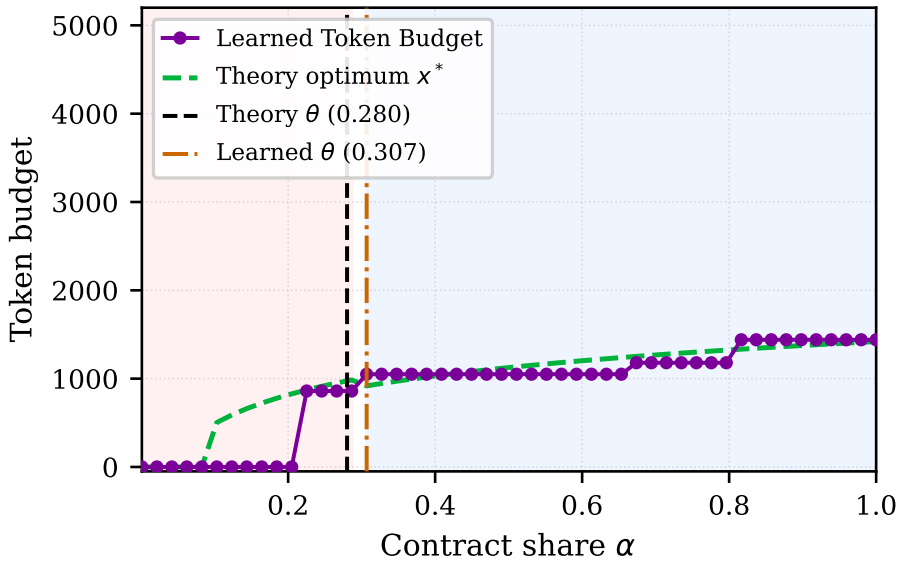


Figure 5: MMLUPro: DeepSeek 1.5B vs. Gemma 4B

Model Pairing	c_H	θ_0 (est.)	θ_0 (learn)
Intra-Family Pairs			
Llama 3.2: 1B *vs.* 3B	0.0200	0.589	0.633
DeepSeek R1: 1.5B *vs.* 7B	0.0075	0.309	0.286
Gemma 4: E2B *vs.* E4B	0.0060	0.339	0.286
Inter-Family Pairs			
Llama 1B *vs.* Gemma 4B	0.0200	0.520	0.592
Llama 1B *vs.* DS 1.5B	0.0075	6.931	L dom.
DS 1.5B *vs.* Gemma 4B	0.015	0.410	0.409
DS 1.5B *vs.* Gemma 2B	0.015	0.280	0.307
DS 1.5B *vs.* Llama 3B	0.006	N/A	H dom.

Table 3: **MMLUPro Domain**: Learned model choices (determined by θ_0). We fix $c_L = 0.005$ for all pairings.

5.3 Principal's Learning

In this section, we show that the Principal can learn the appropriate linear contract when interacting with a trained best-responding Agent from previous sections. The Principal's learning process is outlined in the previous section.

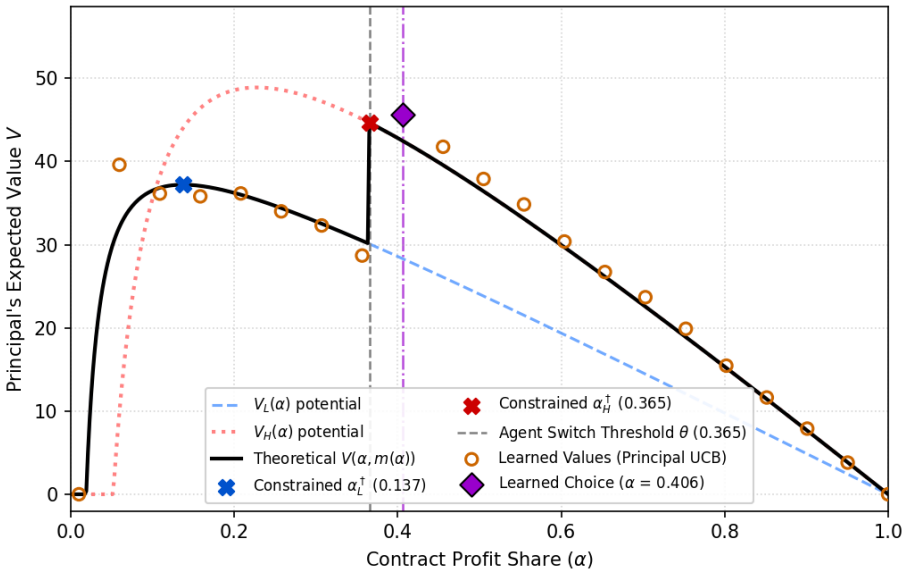


Figure 6: Principal's Learning for Llama 3.2: 1B *vs.* 3B

We use 21 arms for the Principal and train it for $3,000$ episodes against a frozen, trained Agent from the previous section. We test this in the `MATH` domain with different model pairings. For the Llama 3.2: 1B *vs.* 3B pairing, Fig. 6 shows the theoretically expected utility values of the Principal over the domain (in black) and the learned value at each arm (orange circles). The constrained $\alpha_m^\dagger$ (see Eq. (6)) for each model are calculated and shown based on the calibrated parameters. Finally, the arm (contract share) chosen by a greedy Principal post training is shown as the purple diamond, which is close to theoretical expectations. Similar results are seen for the other pairing; refer to Fig. 10 in App. E.

6 Conclusion

In this paper, we extended the classical Principal-Agent framework to address the emerging moral hazard in autonomous LLM delegation, a setting where an Agent dynamically selects both a reasoning technology (model choice) and a continuous effort level (token budget). By modelling the LLM inference process as a saturating production function, we derived the Agent's optimal token allocation and characterized the switching threshold at which it becomes economically rational to adopt a more capable, yet more expensive, model. Furthermore, we determined the Principal's optimal linear contract that maximizes expected utility despite the Agent's hidden actions. Our empirical calibrations on the MATH and MMLUPro benchmarks, paired with simulations using contextual bandit algorithms, demonstrated that both the Principal and the Agent converge toward strategies consistent with our theoretical equilibrium.

As LLM inference-time scaling and budget-aware reasoning continue to advance, the economic implications of optimizing joint model and budget choices will only grow in relevance. While our current framework assumes verification is costless, we have established the foundations for constant-cost verification in Appendix D, and it would be worthwhile to explore other, more complex verification cost families in the future. Additionally, extending this framework to non-ground truth settings presents a compelling direction for future research. In such environments, both the evaluation of correctness and the valuation of the task depend entirely on the Principal's specific type, introducing severe information asymmetry. Without knowing the exact type of the Principal, the Agent's best response would require reasoning over the type distribution, bridging mechanism design with Bayesian delegation in complex agentic markets.

References

[1] Mohammad Ali Alomrani, Yingxue Zhang, Derek Li, Qianyi Sun, Soumyasundar Pal, Zhanguang Zhang, Yaochen Hu, Rohan Deepak Ajwani, Antonios Valkanas, Raika Karimi, Peng Cheng, Yunzhou Wang, Pengyi Liao, Hanrui Huang, Bin Wang, Jianye Hao, and Mark Coates. Reasoning on a Budget: A Survey of Adaptive and Controllable Test-Time Compute in LLMs, July 2025. http://arxiv.org/abs/2507.02076.

[2] Francesco Bacchiocchi, Matteo Castiglioni, Alberto Marchesi, and Nicola Gatti. Learning Optimal Contracts: How to Exploit Small Action Spaces. *International Conference on Learning Representations*, 2024:11944–11970, May 2024.

[3] Francesco Bacchiocchi, Matteo Castiglioni, Alberto Marchesi, and Nicola Gatti. Regret Minimization for Piecewise Linear Rewards: Contracts, Auctions, and Beyond. In *Proceedings of the 26th ACM Conference on Economics and Computation*, EC '25, page 1020, New York, NY, USA, July 2025. Association for Computing Machinery.

[4] Dirk Bergemann, Alessandro Bonatti, and Alex Smolin. The Economics of Large Language Models: Token Allocation, Fine-Tuning, and Optimal Pricing. In *Proceedings of the 26th ACM Conference on Economics and Computation*, EC '25, page 786, New York, NY, USA, July 2025. Association for Computing Machinery.

[5] Dirk Bergemann, Alessandro Bonatti, and Alex Smolin. Menu Pricing of Large Language Models, March 2026. http://arxiv.org/abs/2502.07736.

[6] Mark Boddy and Thomas Dean. Solving time-dependent planning problems. In *Proceedings of the 11th International Joint Conference on Artificial Intelligence - Volume 2*, IJCAI'89, pages 979–984, San Francisco, CA, USA, August 1989. Morgan Kaufmann Publishers Inc.

[7] Yuheng Bu and Yueyuan Ma. Position: Ai-agent pricing should become more outcome-dependent: An economic perspective. Preprint available at https://buyuheng.github.io/publications.html, 2026.

[8] Devin. Devin — the ai software engineer. https://devin.ai/, 2026. Accessed: 2026-07-20.

[9] Dujian Ding, Ankur Mallick, Shaokun Zhang, Chi Wang, Daniel Madrigal, Mirian Del Carmen Hipolito Garcia, Menglin Xia, Laks V. S. Lakshmanan, Qingyun Wu, and Victor Rühle. BEST-Route: Adaptive LLM Routing with Test-Time Optimal Compute. In *ICML 2025*, June 2025.

[10] Paul Dütting, Michal Feldman, and Inbal Talgam-Cohen. Algorithmic contract theory: A survey. *Found. Trends Theor. Comput. Sci.*, 16(3–4):211–412, December 2025.

[11] 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*, WWW '24, pages 144–155, New York, NY, USA, May 2024. Association for Computing Machinery.

[12] Gillian K. Hadfield and Andrew Koh. An Economy of AI Agents, September 2025. arXiv:2509.01063 [econ.GN].

[13] Harvey AI. Harvey — ai software for legal and professional services. https://www.harvey.ai/, 2026. Accessed: 2026-07-20.

[14] Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the math dataset. *NeurIPS*, 2021.

[15] Bengt Holmstrom and Paul Milgrom. Multitask Principal–Agent Analyses: Incentive Contracts, Asset Ownership, and Job Design. *The Journal of Law, Economics, and Organization*, 7(special_issue):24–52, January 1991.

[16] Bengt Holmström. Moral Hazard and Observability. *The Bell Journal of Economics*, 10(1):74–91, 1979.

[17] Eric Horvitz and John Breese. Ideal partition of resources for metareasoning. Technical Report KSL-90-26, Stanford University, February 1990.

[18] Eric J. Horvitz. Reasoning about beliefs and actions under computational resource constraints. In *Proceedings of the Third Conference on Uncertainty in Artificial Intelligence*, UAI'87, page 429–447, Arlington, Virginia, USA, 1987. AUAI Press.

[19] Wittawat Jitkrittum, Harikrishna Narasimhan, Ankit Singh Rawat, Jeevesh Juneja, Congchao Wang, Zifeng Wang, Alec Go, Chen-Yu Lee, Pradeep Shenoy, Rina Panigrahy, Aditya Krishna Menon, and Sanjiv Kumar. Universal Model Routing for Efficient LLM Inference. In *ICML 2026*, October 2025.

[20] Lihong Li, Wei Chu, John Langford, and Robert E. Schapire. A contextual-bandit approach to personalized news article recommendation. In *Proceedings of the 19th International Conference on World Wide Web*, WWW '10, pages 661–670, New York, NY, USA, April 2010. Association for Computing Machinery.

[21] Penghui Qi, Zichen Liu, Tianyu Pang, Chao Du, Wee Sun Lee, and Min Lin. Optimizing Anytime Reasoning via Budget Relative Policy Optimization. In *Advances in Neural Information Processing Systems*, volume 38, pages 23429–23451. Curran Associates, Inc., 2025.

[22] Paulius Rauba, Simonas Cepenas, and Mihaela van der Schaar. Multi-agent systems should be treated as principal-agent problems, 2026. https://arxiv.org/abs/2601.23211.

[23] Eden Saig, Ohad Einav, and Inbal Talgam-Cohen. Incentivizing quality text generation via statistical contracts. In *Proceedings of the 38th International Conference on Neural Information Processing Systems*, volume 37 of *NIPS '24*, pages 51196–51222, Red Hook, NY, USA, December 2024. Curran Associates Inc.

[24] Eden Saig, Tamar Garbuz, Ariel D. Procaccia, Inbal Talgam-Cohen, and Jamie Tucker-Foltz. Adaptive Contracts for Cost-Effective AI Delegation. In *ICML 2026*. ICML, July 2026. http://arxiv.org/abs/2603.17212.

[25] Nenad Tomašev, Matija Franklin, and Simon Osindero. Intelligent AI Delegation, February 2026. arXiv:2602.11865 [cs.AI].

[26] Ander Artola Velasco, Stratis Tsirtsis, and Manuel Gomez Rodriguez. Auditing pay-per-token in large language models. In *The 29th International Conference on Artificial Intelligence and Statistics*, 2026.

[27] Yubo Wang, Xueguang Ma, Ge Zhang, Yuansheng Ni, Abhranil Chandra, Shiguang Guo, Weiming Ren, Aaran Arulraj, Xuan He, Ziyan Jiang, et al. Mmlu-pro: A more robust and challenging multi-task language understanding benchmark. *arXiv preprint arXiv:2406.01574*, 2024.

[28] Hao Wen, Yifan Su, Feifei Zhang, Yunxin Liu, Yunhao Liu, Ya-Qin Zhang, and Yuanchun Li. ParaThinker: Native Parallel Thinking as a New Paradigm to Scale LLM Test-time Compute, August 2025. http://arxiv.org/abs/2509.04475.

[29] Hao Wen, Xinrui Wu, Yi Sun, Feifei Zhang, Liye Chen, Jie Wang, Yunxin Liu, Yunhao Liu, Ya-Qin Zhang, and Yuanchun Li. BudgetThinker: Empowering Budget-aware LLM Reasoning with Control Tokens, August 2025. http://arxiv.org/abs/2508.17196.

[30] Jingbo Yang, Bairu Hou, Wei Wei, Yujia Bao, and Shiyu Chang. Ares: Adaptive Reasoning Effort Selection for Efficient LLM Agents, March 2026. http://arxiv.org/abs/2603.07915.

[31] Shu Zhou, Rui Ling, Junan Chen, Xin Wang, Tao Fan, and Hao Wang. When More Thinking Hurts: Overthinking in LLM Test-Time Compute Scaling, April 2026. http://arxiv.org/abs/2604.10739.

[32] Banghua Zhu, Stephen Bates, Zhuoran Yang, Yixin Wang, Jiantao Jiao, and Michael I. Jordan. The Sample Complexity of Online Contract Design. In *Proceedings of the 24th ACM Conference on Economics and Computation*, EC '23, page 1188, New York, NY, USA, July 2023. Association for Computing Machinery.

[33] Shlomo Zilberstein. Using Anytime Algorithms in Intelligent Systems. *AI Magazine*, 17(3):73–83, 1996.

Appendix

A Surplus and Agency Costs

We now consider the first-best benchmark for the base model. In the first-best scenario, a single party holds the model, reaps the benefit and spends compute cost, *i.e.*, single party receives the total social surplus, with no moral hazard. This benchmark is the limiting case $\alpha = 1$ of the contract space above. The optimal token budget is the same as optimal effort for model m at $\alpha = 1$ from Eq. (3):

$$x_m^{*,FB} = \frac{1}{k} \ln\left(\frac{1}{\tau_m}\right) = x_m^*(1)$$

At $\alpha = 1$ the Agent's payoff $U_m(1) = q_m(x_m^*(1)) - c_m x_m^*(1)$ equals total surplus under model m at efficient effort, so $D(1) = U_H(1) - U_L(1)$ is exactly the comparison of first-best surplus across the two models. Recall that, in the standard order, $D(1) > 0$ is the condition under which a switching threshold θ exists in $[0, 1]$. The interpretation: the Agent eventually adopts H under some contract if and only if H is the first-best efficient model. We write $m^{FB} = \arg\max_m U_m(1)$ for the first-best model, and

$$S_m^{FB} \;\equiv\; U_m(1) \;=\; M_m\, h(\tau_m), \qquad h(z) \equiv 1 - z + z \ln z,$$

for the first-best surplus achievable under model m.

Agency Cost At the Principal's unconstrained optimum $\alpha_m^* = \sqrt{\tau_m}$, total surplus is $S_m(\sqrt{\tau_m}) = U_m(\sqrt{\tau_m}) + V_m(\sqrt{\tau_m})$, strictly below S_m^{FB}. The loss or cost of inducing Agent to select each model m is given by:

$$S_m^{FB} - S_m(\sqrt{\tau_m}) \;=\; M_m \sqrt{\tau_m}\, h(\sqrt{\tau_m}),$$

and normalizing by first-best surplus gives a loss ratio that depends only on the activation threshold τ_m,

$$\ell_m \;\equiv\; \frac{S_m^{FB} - S_m(\sqrt{\tau_m})}{S_m^{FB}} \;=\; \frac{\sqrt{\tau_m}\, h(\sqrt{\tau_m})}{h(\tau_m)}.$$

Further, re-arranging terms gives us:

$$U_m(\sqrt{\tau_m}) \;=\; S_m^{FB} - S_m(\sqrt{\tau_m}).$$

The Agent's utility at the Principal's optimal contract is exactly equal to the Agency cost, a consequence of the specific saturating-exponential functional form we assume; not a general property of moral hazard models.

B Burn-in Switching Threshold

This appendix derives the qualitative behavior of the burn-in switching threshold θ_0 (Eq. 10) for an illustrative special case. Motivated by calibration tables where k and b are roughly

similar (not exactly equal) for models of the same family for a given task, we consider a special case with identical token saturation rates ($k_L = k_H = k$) and identical burn-in token requirements ($b_L = b_H = b$). Under this assumption, the participation thresholds (Eq. 9) become constant scalar multiples of the corresponding activation thresholds, with a common scale factor $\rho = f^{-1}(kb) \geq 1$:

$$\tau_L^P = \rho\,\tau_L, \qquad \tau_H^P = \rho\,\tau_H.$$

B.1 Flipped Order ($\tau_H < \tau_L$)

When the high-capability model activates first, the baseline derivative condition remains intact, since the fixed burn-in costs vanish under differentiation:

$$\mathcal{U}_H'(\alpha) = M_H\left(1 - \frac{\tau_H}{\alpha}\right) > M_L\left(1 - \frac{\tau_L}{\alpha}\right) = \mathcal{U}_L'(\alpha).$$

Coupled with $\tau_H^P < \tau_L^P$ (which follows immediately from $\tau_H < \tau_L$ and the common scale factor ρ), model H both breaks even earlier and climbs strictly faster than model L at every point in the active domain. Consequently, model H globally dominates, and no switching point exists exactly as in the base model without burn-in.

Remark. *Without the same-family assumption, it is possible to have $\tau_H < \tau_L$ but $\tau_H^P > \tau_L^P$ (i.e., different k_m, b_m break the common scale factor ρ). This would open a brief window in which L is the only economically viable model, before H eventually catches up and dominates.*

B.2 Standard Order ($\tau_L < \tau_H$)

When model L activates first, the uniform scaling yields $\tau_L^P < \tau_H^P$. We reuse the baseline utility differential $D(\alpha)$ from Eq. (5), which is strictly convex under this ordering (since $M_H\tau_H > M_L\tau_L$ term-wise) with a unique zero θ on its increasing branch.

Rewriting Eq. (10) with common b:

$$D(\theta_0) = (c_H - c_L)\,b. \tag{11}$$

Under the standing assumption $c_H > c_L$ (the more capable model carries a higher per token cost) and $b > 0$, the right-hand side is strictly positive, so

$$D(\theta_0) > D(\theta) = 0.$$

Since D is increasing on (θ, ∞) (the increasing branch of the convex function), $D(\theta_0) > D(\theta)$ directly implies

$$\theta_0 > \theta.$$

The burn-in cost, being more expensive in absolute terms for the costlier model ($c_H b > c_L b$), strictly delays the switch. As in the base model, θ_0 exists in $[0, 1]$ only if $D(1) \geq (c_H - c_L)b$; otherwise L dominates throughout the domain.

C Linear Contracts with Fixed Payments

We now consider a contract $\langle \alpha, \beta \rangle$, where $\alpha \in [0, 1]$ remains the performance-based revenue share and $\beta \in \mathbb{R}$ is a flat transfer payment independent of output q.

The updated normalized utilities for the Agent and Principal are:

$$U(\alpha, \beta, m, x) = \alpha \cdot q_m(x) - c_m x + \beta,$$
$$V(\alpha, \beta, m, x) = (1 - \alpha) \cdot q_m(x) - \beta.$$

Optimal Effort and Model Choice

The introduction of a fixed payment β does not alter the marginal incentives for effort. Taking the first-order derivative of $U(\alpha, \beta, m, x)$ with respect to x eliminates β, leaving the first-order condition identical to the pure revenue-sharing case. Consequently, the optimal effort x_m^* and the activation threshold τ_m remain exactly as derived in Equations (3) and (4).

Substituting optimal effort back into the Agent's utility gives:

$$\begin{aligned} U_m(\alpha, \beta) &= U_m(\alpha) + \beta \\ &= M_m \left[\alpha - \tau_m - \tau_m \ln\left(\frac{\alpha}{\tau_m} \right) \right] + \beta, \quad \alpha > \tau_m. \end{aligned} \tag{12}$$

Similarly, because β shifts the utility curves of all models $m \in \mathcal{M}$ by the same constant amount, it cancels out during the Agent's model selection phase. The condition $U_H(\alpha, \beta) > U_L(\alpha, \beta)$ is mathematically equivalent to $U_H(\alpha) > U_L(\alpha)$. Therefore, the switching threshold θ and the two scenarios (Standard vs. Flipped) remain entirely unchanged from the previous section.

Principal's Optimization

The presence of the fixed transfer fundamentally alters the Principal's optimization strategy. The Principal seeks to maximize $V_m(\alpha, \beta)$ subject to the Agent's Individual Rationality (IR) constraint, assuming an outside option utility of zero:

$$U_m(\alpha, \beta) \geq 0 \implies U_m(\alpha) + \beta \geq 0$$

To maximize its own utility, the Principal will extract all surplus from the Agent by setting the fixed payment such that the IR constraint binds exactly:

$$\beta^* = -U_m(\alpha)$$

If $U_m(\alpha) > 0$, this requires $\beta^* < 0$, functioning as a fee paid by the Agent to the Principal for the right to perform the task.

Substituting β^* into the Principal's objective function aligns the Principal's utility with the total social surplus of the system $S_m(\alpha)$:

$$\begin{aligned} V_m(\alpha, \beta^*) &= (1-\alpha) \cdot q_m(x_m^*) - (-U_m(\alpha)) \\ &= (1-\alpha) \cdot q_m(x_m^*) + \alpha \cdot q_m(x_m^*) - c_m x_m^* \\ &= q_m(x_m^*) - c_m x_m^* \\ &= S_m(\alpha) \end{aligned}$$

The Principal effectively designs the contract to maximize total surplus, which they then fully extract via β^*. Using $q_m(x_m^*) = M_m(1 - \tau_m/\alpha)$ and $x_m^* = \frac{1}{k_m} \ln\left(\frac{\alpha}{\tau_m}\right)$, the total surplus function is:

$$S_m(\alpha) = M_m\left(1 - \frac{\tau_m}{\alpha}\right) - \frac{c_m}{k_m} \ln\left(\frac{\alpha}{\tau_m}\right)$$

Differentiating with respect to α and setting to zero:

$$\frac{dS_m}{d\alpha} = M_m \frac{\tau_m}{\alpha^2} - \frac{c_m}{k_m} \frac{1}{\alpha} = 0$$

Recall from the activation threshold definition that $\frac{c_m}{k_m} = M_m \tau_m$. Substituting this identity yields:

$$M_m \tau_m \left(\frac{1}{\alpha^2} - \frac{1}{\alpha}\right) = 0.$$

Since $M_m > 0$ and $\tau_m > 0$, the only strictly positive solution is $\alpha = 1$; also $dS_m/d\alpha > 0$ throughout $(0, 1)$. This is the classic *selling the firm* result: by setting $\alpha = 1$, the Principal eliminates the misalignment between the Agent's private return and total surplus, since the Agent now keeps the full marginal return and bears the full marginal cost of every token spent.

The Agent's model choice at $\alpha = 1$ is governed by the switching threshold θ from previous section , independent of β. Since S_m is increasing throughout $(0, 1)$ for both models, $\alpha = 1$ is surplus-maximizing and the sign of the utility differential $D(1)$ (see Eq. (5)) identifies which model the Agent selects:

$$\alpha^* = 1, \qquad m^* = \begin{cases} H & \text{if } D(1) > 0 \\ L & \text{if } D(1) \leq 0. \end{cases}$$

The Principal sets the fixed transfer to bind the Agent's IR constraint exactly, extracting the realized surplus in full:

$$\beta^* = -U_{m^*}(1) = -M_{m^*}\left[1 - \tau_{m^*} - \tau_{m^*} \ln\left(\frac{1}{\tau_{m^*}}\right)\right].$$

Remark: This implements the first-best outcome. At $\alpha = 1$, the Agent's optimal effort (see Eq. (3)) coincides with the surplus-maximizing FOC a Principal with direct control over tokens would solve, so $x_m^*(1)$ is the efficient token budget. The fixed-payment contract therefore induces the efficient model and efficient effort simultaneously, with zero loss due to moral hazard (albeit trivially by selling the firm, which is not always realistic).

Limited Liability Constraint ($\beta \geq 0$)

If we assume limited liability, which is common in most real-world settings, it imposes the added constraint $\beta \geq 0$. The Principal solves:

$$\max_{\alpha,\beta} (1-\alpha) \cdot q_m(x_m^*(\alpha)) - \beta \quad \text{s.t.} \quad U_m(\alpha) + \beta \geq 0, \quad \beta \geq 0$$

Because any $\beta > 0$ directly reduces the Principal's payoff without modifying the Agent's marginal incentives for effort, the limited liability constraint gives:

$$\beta_{LL}^* = 0$$

Consequently, the problem collapses entirely back to the pure revenue-sharing model from previous section.

D Costly Verification

We extend the base model to allow the Principal to incur a fixed cost $c_v \geq 0$ per round to observe (verify) the realized output quality q. The constant cost assumption is justified in cases such as math verification or unit tests for code outputs. In the base model this cost is implicitly zero; here we make it explicit. Note that we consider the case where the Principal verifies in every round. An interesting extension could be probabilistic verification, but we leave that for future work.

The verification cost c_v is borne entirely by the Principal and, therefore, does not appear in the Agent's utility. Consequently, the Agent's optimal budget for model m and the Agent's model choice through switching threshold θ remain unchanged.

The Principal's utility from inducing model m at share α becomes

$$V_m^v(\alpha) \;=\; (1-\alpha)\, q_m(x_m^*(\alpha)) - c_v \;=\; V_m(\alpha) - c_v,$$

i.e. the verified payoff is the unverified payoff $V_m(\alpha)$ shifted down by the constant fixed cost c_v.

For any $c_v \geq 0$ constant in α, the FOC remains the same because the constant term gets differentiated out to zero. This implies the unconstrained optima $\sqrt{\tau_m}$ remains unchanged. We have shown that the model choice from Agent's side, determined by θ also does not change. Consequently the constrained optima $\alpha_L^\dagger$ and $\alpha_H^\dagger$, which are $\sqrt{\tau_m}$ clipped against the switching threshold θ, are also unchanged.

So a constant verification cost changes *none* of the base model's structural results: the Principal's optimal contract share is exactly as derived in the base model. Verification cost only ever shows up as a constant shift in the Principal's realized payoff. However, this matters for whether the Principal wants to contract at all. In the base model, the Principal's utility was always non-negative, but the fixed cost introduces the need to check participation constraint.

D.1 Participation

The Principal's payoff at the (unconstrained) optimum is

$$V_m(\sqrt{\tau_m}) \;=\; M_m\big(1 + \tau_m - \sqrt{\tau_m} - \tau_m/\sqrt{\tau_m}\big) \;=\; M_m\big(1 - \sqrt{\tau_m}\big)^2,$$

The Principal participates *i.e.*, offers a contract to induce model m only if $V_m^v(\alpha_m^\dagger) \geq 0$,

$$c_v \;\leq\; V_m(\alpha_m^\dagger).$$

At the unconstrained optimum this becomes the clean threshold $c_v \leq M_m(1-\sqrt{\tau_m})^2$: a small verification cost relative to this never changes the Principal's optimal share, but a sufficiently large c_v can make model m non-viable for the Principal.

D.2 Alternative verification costs

Treating c_v as a fixed constant, while realistic in some scenarios, makes the verification problem fairly trivial as it gets differentiated out in all first-order conditions. Other interesting choices for verification cost are as follows.

- c_v **depending on** q**.** If verification cost scales with the quality being verified (e.g., low quality or buggy code will not even compile or error out quickly with the unit tests), $c_v = c_v(q)$ enters $V_m^v(\alpha)$ as a function of α through $q_m(x_m^*(\alpha))$, and the first-order condition picks up a $c_v'(q) \cdot q_m'(x_m^*) \cdot \frac{dx_m^*}{d\alpha}$ term and α_m^* would shift in general.
- c_v **depending on** m**.** Even holding c_v constant in α, allowing $c_v = c_{v,m}$ to differ by model (e.g. verifying a more elaborate H-model's output costs more than a short L-model output) does not change any of the structural results, but it does mean the participation thresholds in the previous subsection differ across models for a second reason (beyond M_m, τ_m already varying), which could be a source of an additional, verification-driven bias toward the cheaper-to-verify model.
- **Endogenous/probabilistic verification.** The Principal could choose to verify only with some probability $p < 1$, trading off expected verification cost $p \cdot c_v$ against reduced ability to enforce the contract (an Agent who anticipates low verification probability may deviate). This introduces strategic aspects to the interaction between verification and the Agent's incentives.

E Experiment Details and Additional Results

Datasets and Filtering. For the MATH dataset [14], which contains questions across difficulty levels 1 through 5, we filter exclusively for questions in levels 3 and 4 to maintain consistent task difficulty for calibrating production curves. We filter out questions containing graphic components by matching raw markup tags such as `[asy]`, resulting in $2,962$ questions for evaluation. The final JSON file is included in the code package. For MMLU-Pro [27], we evaluate across all $12,032$ available questions in the test partition.

Models, Serving, and Infrastructure. All base models are loaded directly using their official Hugging Face repository identifiers, as listed in Table 4. All bandit runs for different pairings are executed on a single NVIDIA H100 GPU (80GB). Open-weights LLMs are served locally via vLLM using an OpenAI-compatible HTTP API server without additional model quantization. To ensure deterministic generation, all inference requests are made with zero sampling temperature (temperature = 0.0).

Answer Parsing and Evaluation.

- **MATH Dataset:** Evaluated using the `math_verify` library to verify symbolic and numerical equivalence against the ground truth answer inside `\boxed{}` outputs.
- **MMLU-Pro Dataset:** Using regex pattern matching to locate response strings matching `"answer is (X)"` or `"answer: (X)"`.

In both benchmarks, we first strip reasoning scratchpads (e.g., extracting text following `</think>` tags for DeepSeek models). Also, if a model generation is truncated due to budget constraints or fails answer extraction, it is assigned a accuracy score of 0 (0 reward).

Action Space and Discretization. Across both tasks, the action space for the bandit is discretized into $N = 21$ linearly spaced token bins spanning dataset- and model-specific minimum ($x_{\min}$) and maximum ($x_{\max}$) token limits.

		$(x_{\min}, x_{\max})$	
Model Family	**Hugging Face Repository ID**	**MATH**	**MMLU-Pro**
Llama 3.2	`meta-llama/Llama-3.2-1B-Instruct`	$[100, 1000]$	$[100, 1000]$
	`meta-llama/Llama-3.2-3B-Instruct`	$[100, 1000]$	$[100, 1000]$
DeepSeek R1	`deepseek-ai/DeepSeek-R1-Distill-Qwen-1.5B`	$[1000, 5000]$	$[400, 5000]$
	`deepseek-ai/DeepSeek-R1-Distill-Qwen-7B`	$[1000, 5000]$	$[400, 5000]$
Gemma 4	`google/gemma-4-E2B-it`	$[400, 3000]$	$[400, 3000]$
	`google/gemma-4-E4B-it`	$[400, 3000]$	$[400, 3000]$

Table 4: Model identifiers and corresponding discretized token budget bounds $(x_{\min}, x_{\max})$

Hyperparameters and Reproducibility. All experiments use a single fixed random seed of 42 (`seed` $= 42$). The decision agent runs for $2,000$ episodes on MATH and MMLU-Pro, using a batch size parameter of $k = 16$, i.e., in each round, the reward is the accuracy over a set of 16 questions. All code is attached as a zip. A more organized version will be made publicly available upon acceptance.

LinUCB is a contextual bandit algorithm [20]: it assumes each arm's expected reward is a linear function of the current context, and maintains a running estimate of the weights (using ridge-regression) from observed rewards. Similar to UCB, at each round, it adds an exploration bonus which shrinks as more data accumulates for that arm. We use a smaller exploration weight at the model-choice level ($\gamma = 1$) and a larger one at the token-budget level ($\gamma = 2$): since the model-choice level's reward estimate depends on the token-budget policy already being close to its optimum, we use a larger exploration weight at the token-budget level, which has a larger action space (21 arms vs. 2), to ensure its reward estimates are reliable before the model-choice level's decision settles.

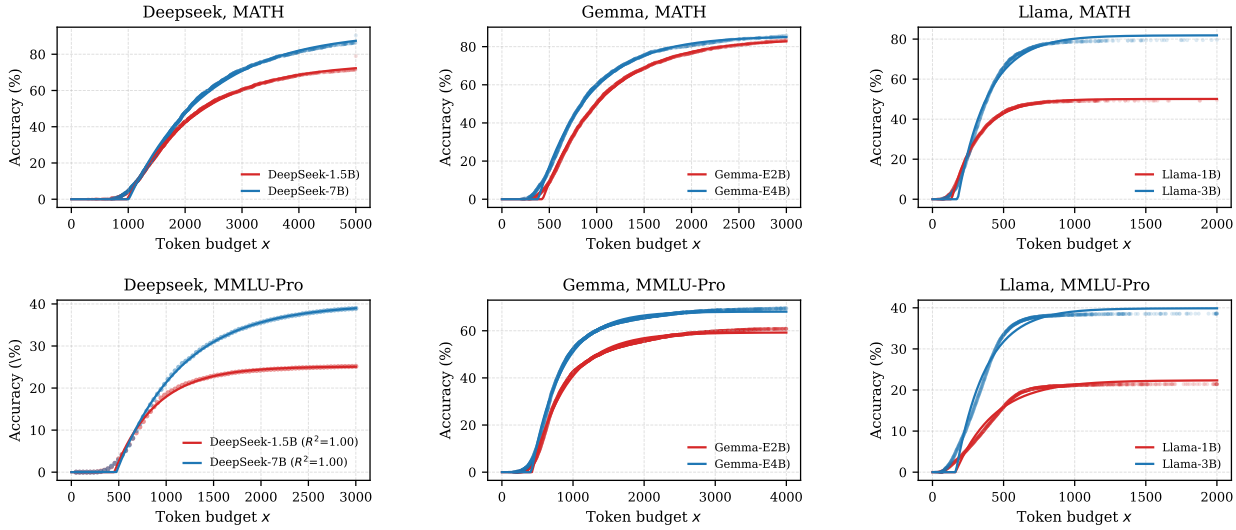
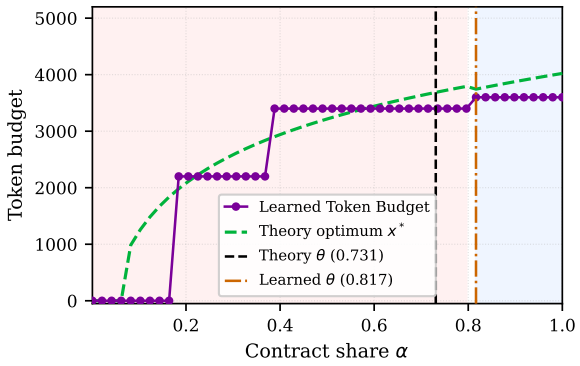
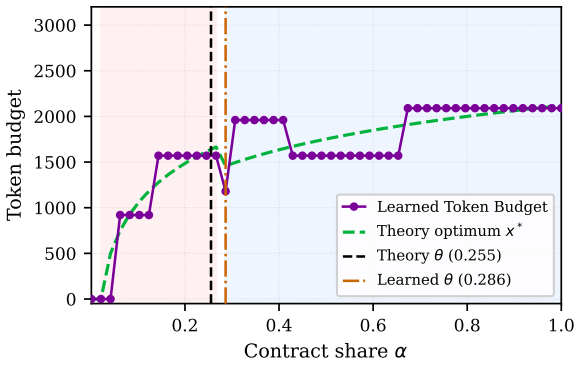


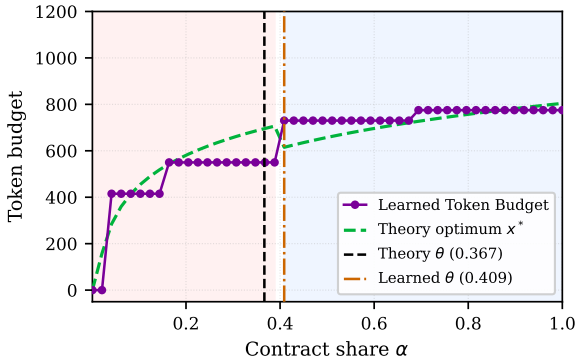
Figure 7: Accuracy vs budget for all 3 model families and 2 task domains, along with the fitted curves.



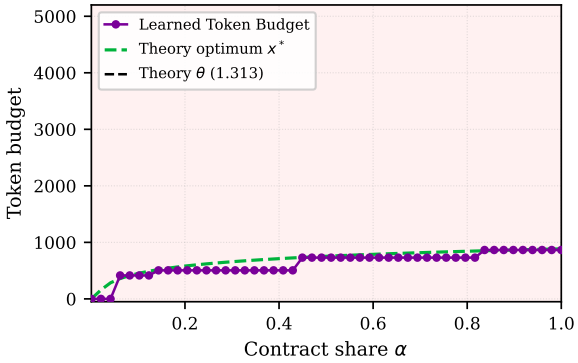
(a) DeepSeek R1 1.5B *vs.* 7B



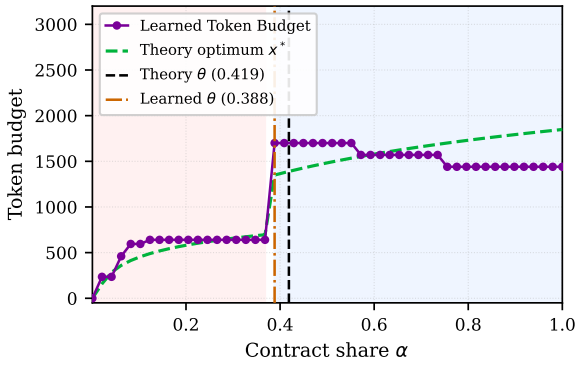
(b) Gemma E2B *vs.* E4B



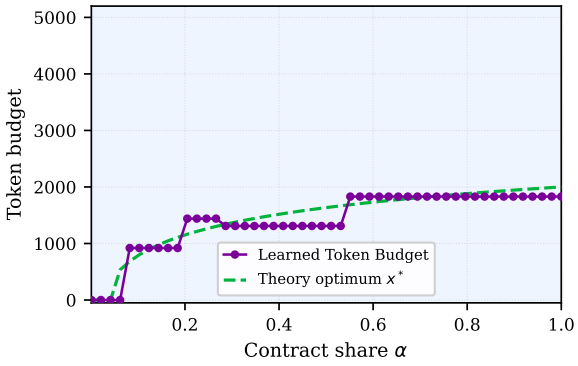
(c) Llama 1B *vs.* 3B



(d) Llama 1B *vs.* DeepSeek 1.5B

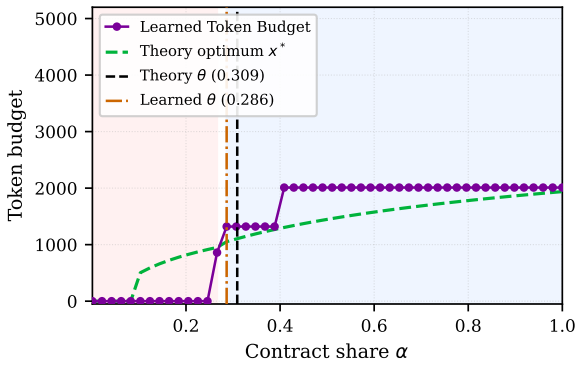


(e) Llama 1B *vs.* Gemma 4B

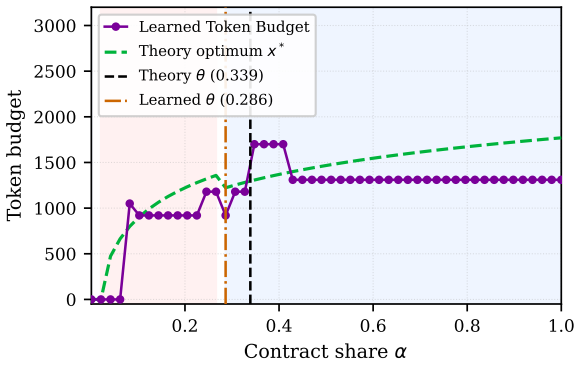


(f) DeepSeek 1.5B *vs.* Gemma 4B

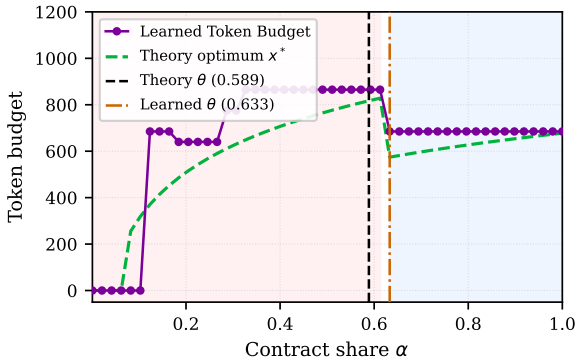
Figure 8: Learned policies of various model pairings in the `MATH` domain.



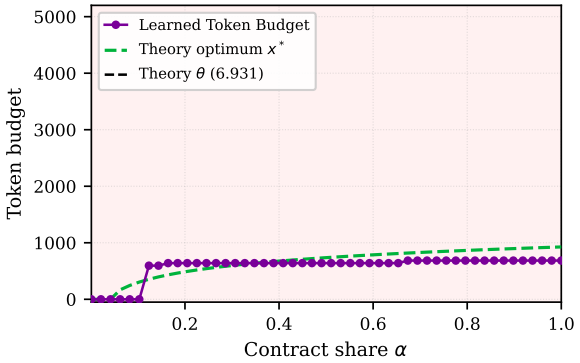
(a) DeepSeek R1 1.5B *vs.* 7B



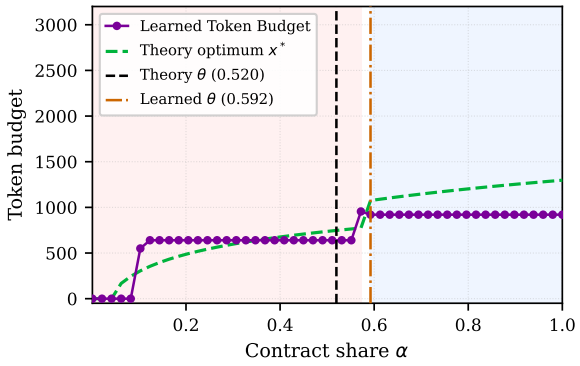
(b) Gemma E2B *vs.* E4B



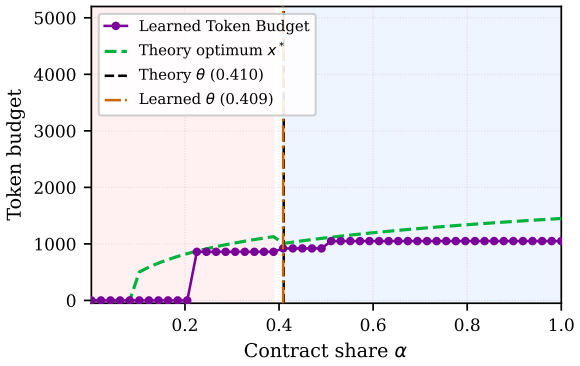
(c) Llama 1B *vs.* 3B



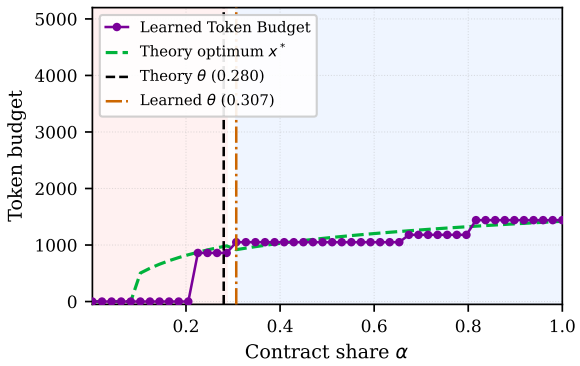
(d) Llama 1B *vs.* DeepSeek 1.5B



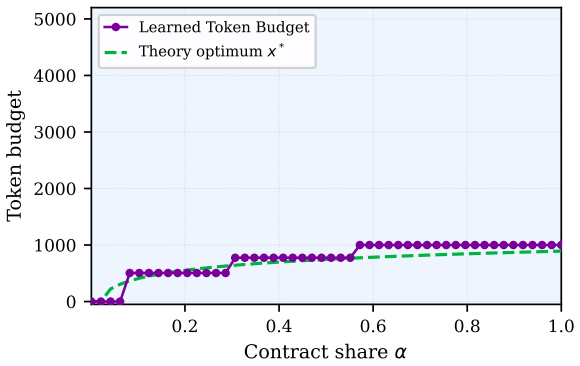
(e) Llama 1B *vs.* Gemma 4B



(f) DeepSeek 1.5B *vs.* Gemma 2B

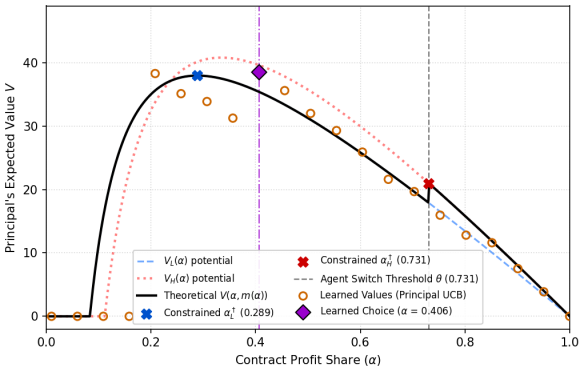


(g) DeepSeek 1.5B *vs.* Gemma 4B

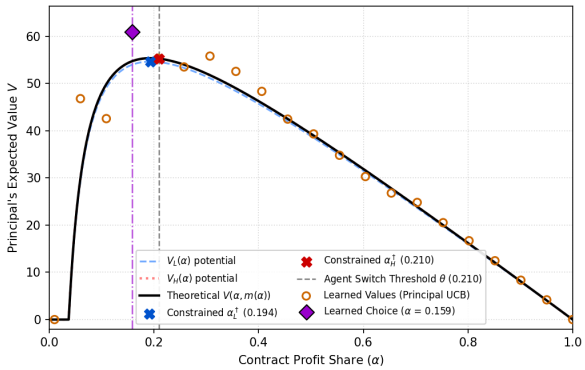


(h) DeepSeek 1.5B *vs.* Llama 3B

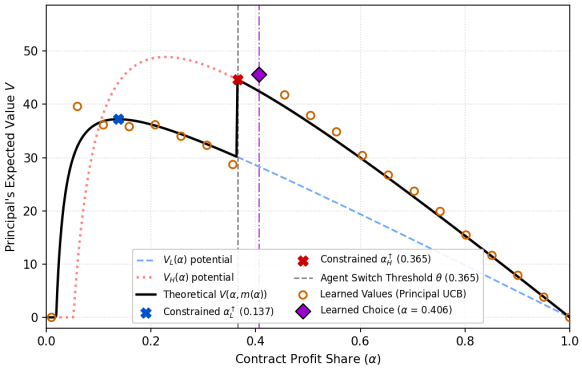
Figure 9: Learned policies for the `MMLU Pro` domain.



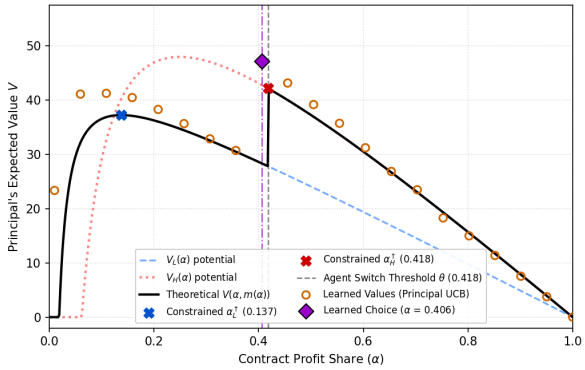
(a) DeepSeek R1 1.5B *vs.* 7B



(b) Gemma E2B *vs.* E4B

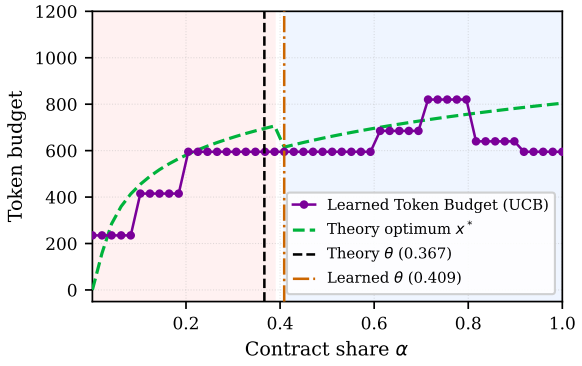


(c) Llama 1B *vs.* 3B

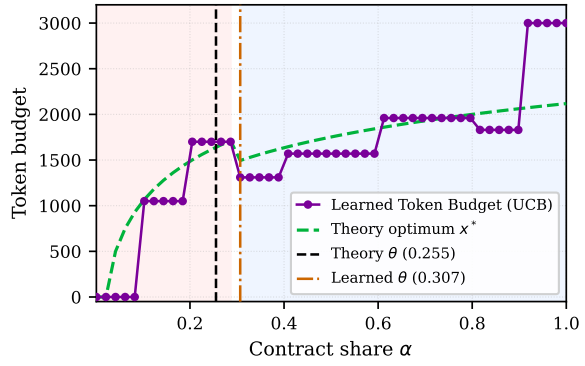


(d) Llama 1B *vs.* Gemma 4B

Figure 10: Principal's Learning for different model pairings on the MATH task



(a) Llama 1B *vs.* 3B



(b) Gemma E2B *vs.* E4B

Figure 11: Agent using simple UCB in MATH task for two different pairings. The context α is discretized into 10 bins, and we use a standard UCB instead of LinUCB with handcrafted features. The Agent considers each bin to be independent, and therefore takes a longer time to converge, and these were trained for $5,000$ episodes.

F Appendix F: LLM as Controller

Instead of the LinUCB learning Agent, we replace it with a LLM Controller, where we prompt a LLM with details about the general task domain, the 2 LLM models it has access to, their respective costs, and the profit objective it is supposed to maximize. Note that the prompt does not include any of our theory or calibrated parameters. We instead include a sliding window of the last 20 decisions including details about the contract offered, the model chosen, token budget allocated, the resulting accuracy and reward. Finally, we append the contract offered in the current round, and prompt it to return a JSON object with model choice, token budget, and a justification. The exact prompt used is in Appendix G.

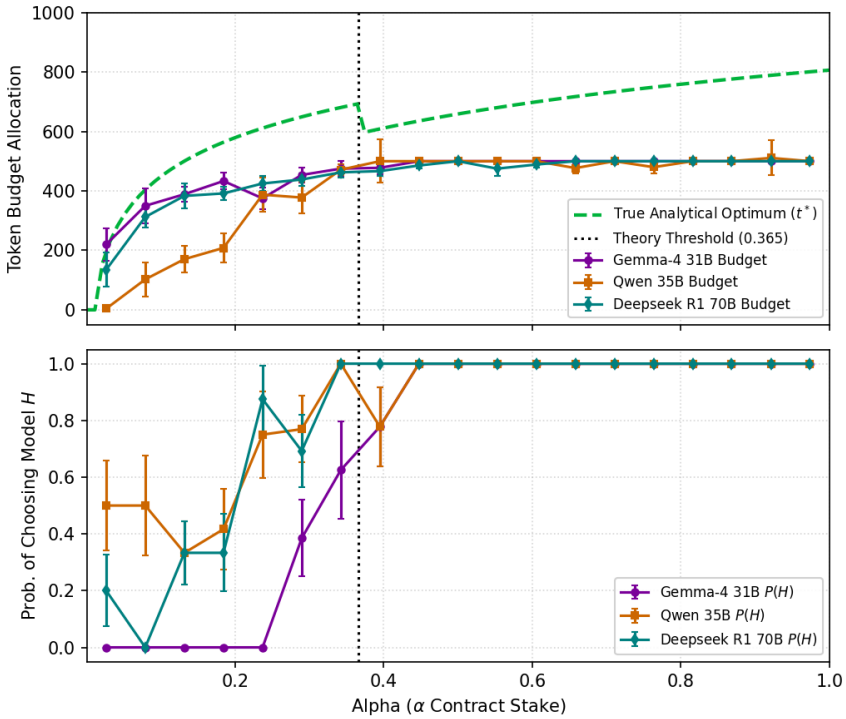


Figure 12: LLM as Controller

We experiment with a few different controller models; significantly more advanced and larger models compared to the model choices. We test it out for 220 rounds, with the first 20 rounds not being used in analysis as the sliding window of history is not yet full. We consider Gemma 4 31B, Qwen 3.6 35B, Deepseek R1 70B with zero temperature. We test this on the MATH domain and the Llama 1B *vs.* 3B pairing, and the observed results are shown in Figure 12. Recall that the controller receives a contract stake α_t drawn uniformly at random from the domain $\alpha_t \in [0, 1]$. So, we discretize the contracts into 20 bins and average the model choices within each bin; the standard errors are also reported.

All 3 controllers pick up the qualitative direction of model choice, with $P(H)$ rising with α and saturating near the theoretical threshold. The sharpness of this switch is not the same as the bandit controller and varies significantly within these models; the Qwen and Deepseek controllers pick model H at higher rates than optimal below the threshold. On token budgets,

all controllers spend less than optimal, with the token budget plateauing past the threshold. We take it as preliminary evidence that finding the Agent's two-dimensional best response is not trivial, and leave a systematic study of LLM controllers for future work. More importantly, this also does not account for the *token cost of prompting the controller* itself each round, on top of the Agent's own token spend for the actual task; a cost the bandit-based approach does not incur.

G Appendix G: Prompts

MATH Prompt Template

```
[System Instructions]
Please reason step by step, and put your final answer within \boxed{}.
[User Payload]
Question: [problem_text]
```

MMLU-Pro Prompt Template

```
[System Instructions]
The following is a multiple-choice question (with answer) about [category].
Think step by step and then finish your answer with "The answer is (X)"
where X is the correct letter choice.
[User Payload]
Question: [question_text]
Options: (A) [Option 1] (B) [Option 2] ... (N) [Option N]
Let's think step by step.
```

LLM Controller Prompt Template

```
[System Instructions]
You are an adaptive, economically optimal routing agent.
Context: In each round, you manage a batch of 16 mathematical problems
sampled uniformly from a fixed distribution (MATH dataset, difficulty levels
3-4).
Objective: Maximize net profit per question = (alpha * accuracy_pct) -
(cost_per_token * token_budget)
Configuration Space:
- Option A: '[Model Low Name]' (cost: [Cost Low])
- Option B: '[Model High Name]' (cost: [Cost High])
- Allowed Budget Range per question: [Token Min] to [Token Max]
--- START RECENT PERFORMANCE LOG ---
[History Window Log String, e.g., ``Round i: Alpha=... | Model=... |
Budget=...``]
--- END RECENT PERFORMANCE LOG ---
Constraint: Respond ONLY with a valid JSON object matching this schema.
Choose any integer value between [Token Min] and [Token Max] for
token_budget:
{
  "justification": "reasoning based on the objective, history and current
alpha",
  "chosen_model": "model_name",
  "token_budget": <integer value here>
}
[User Payload]
Current Contract: alpha = <sampled alpha>
```

H Appendix H: Extension to N Models

We consider $\mathcal{M} = \{1, \ldots, N\}$ in place of $\{L, H\}$. The per-model quantities τ_m, $x_m^*(\alpha)$, and $U_m(\alpha)$ are unchanged, since they follow from a single-model optimization and do not depend on the size of $\mathcal{M}$. What changes is the Agent's model choice, which is now

$$m^*(\alpha) = \arg\max_{m \in \mathcal{M}} U_m(\alpha),$$

i.e., the Agent's optimal model choice is based on the upper envelope of N curves of the form (4), rather than a single crossing between two. We characterize this envelope in two steps: first, which models can be removed from consideration entirely; second, whether the remaining models are visited in capability order as α increases.

Capability Order

Model i is *dominated* by model j if $M_j \geq M_i$ and $\tau_j \leq \tau_i$, with at least one inequality strict. We claim a dominated model is never the Agent's best response.

Suppose $\tau_j \le \tau_i$. For $\alpha > \tau_i$, both models are active and $\tau_j/\alpha \le \tau_i/\alpha$, so

$$\left(1 - \frac{\tau_j}{\alpha}\right) \ge \left(1 - \frac{\tau_i}{\alpha}\right).$$

Combined with $M_j \ge M_i$, this gives $U_j'(\alpha) \ge U_i'(\alpha)$ for all $\alpha > \tau_i$, by the same argument used for the Flipped Order case above. At $\alpha = \tau_i$, $U_i(\tau_i) = 0$ while $U_j(\tau_i) \ge 0$ (model j is already active, having $\tau_j \le \tau_i$). Since U_j starts weakly ahead of U_i at $\alpha = \tau_i$ and climbs at least as fast for every α beyond it, $U_j(\alpha) \ge U_i(\alpha)$ for all $\alpha \ge \tau_i$; for $\alpha < \tau_i$, $U_i(\alpha) = 0 \le U_j(\alpha)$ trivially. So $U_j(\alpha) \ge U_i(\alpha)$ on all of $[0, 1]$, and model i never wins the envelope.

Removing dominated models, the remaining set $\{1, \dots, K\} \subseteq \mathcal{M}$ has no pair related this way. Sorting by capability, $M_{(1)} < \cdots < M_{(K)}$, we must also have

$$\tau_{(1)} < \cdots < \tau_{(K)},$$

since a violation would mean some pair is still dominated. We refer to this as *capability order*: within the surviving set, more capable models are also more costly to activate.

Capability order alone does not imply the Agent moves through models $1, \dots, K$ one at a time as α increases. For $i < j$ define $D_{ij}(\alpha) = U_j(\alpha) - U_i(\alpha)$; as before, D_{ij} is convex with at most one root θ_{ij}. For three models $i < j < k$,

$$D_{ik}(\alpha) = D_{ij}(\alpha) + D_{jk}(\alpha).$$

If $\theta_{ij} \le \theta_{jk}$, this forces $\theta_{ik} \in [\theta_{ij}, \theta_{jk}]$ and model j is optimal on that interval, as expected. But capability order does not guarantee $\theta_{ij} \le \theta_{jk}$; if instead $\theta_{ij} > \theta_{jk}$, model j is never optimal on $[0, 1]$ even though it is undominated and correctly placed in capability order, since the Agent prefers switching directly from i to k. For example, consider

$$(M_1, \tau_1) = (10, 0.1), \quad (M_2, \tau_2) = (11, 0.5), \quad (M_3, \tau_3) = (50, 0.6),$$

which are in capability order and pairwise undominated. Direct computation gives $D_{12}(1) \approx -5.01 < 0$, so $\theta_{12} > 1$; and $D_{23}(1) \approx 2.99 > 0$, so $\theta_{23} \in (0, 1)$. Since $\theta_{12} > \theta_{23}$, model 2 is never the Agent's choice for any $\alpha \in [0, 1]$: it would only become optimal at a contract share past the allowed range.

Staircase structure

Proposition. Let models $1, \dots, K$ be undominated and in capability order. If the adjacent switching points satisfy

$$\theta_{1,2} < \theta_{2,3} < \cdots < \theta_{K-1,K},$$

then $m^*(\alpha) = i$ for $\alpha \in (\theta_{i-1,i}, \theta_{i,i+1})$, and the $K - 1$ adjacent thresholds fully determine the envelope; the remaining pairwise comparisons are unnecessary.

The case $K = 2$ is the base model. For the inductive step, suppose the ordering holds up to model $i - 1$. Since $D_{i-1,i}$ is convex with a single root at $\theta_{i-1,i}$, once model i overtakes model $i - 1$ it remains ahead for all larger α. Any earlier model $j < i - 1$ is, by the induction hypothesis, already behind model $i - 1$ once α exceeds $\theta_{i-2,i-1} < \theta_{i-1,i}$, and hence remains

behind model i as well by the same convexity argument applied to $D_{j,i}$. So checking neighbors suffices.

When the ordering condition fails, as in the example above, the envelope must be computed from all pairwise thresholds rather than adjacent ones alone; this is the standard problem of finding the upper envelope of pairwise-crossing curves.

Principal's Problem

Given the ordering condition of the Proposition, the Principal's problem extends directly. On each interval $(\theta_{i-1,i}, \theta_{i,i+1})$ where model i is the Agent's choice, $V_i(\alpha)$ remains single-peaked at $\sqrt{\tau_i}$ exactly as in (6)-(7), so

$$\alpha_i^\dagger = \text{clip}(\sqrt{\tau_i},\ \theta_{i-1,i},\ \theta_{i,i+1}), \qquad \alpha^* = \arg\max_i V_i(\alpha_i^\dagger).$$

No new derivation is required beyond the base model once the interval structure is known.