

Truthful and Expected-Payment-Optimal Model Routing in Graph-Based Agentic Workflows

Jiayuan Liu^{1,2} Mingyu Guo³ Jiarui Gan⁴ Vincent Conitzer^{1,2}

¹Carnegie Mellon University ²Foundations of Cooperative AI Lab (FOCAL)

³Adelaide University ⁴University of Oxford

jiayuan4@cs.cmu.edu mingyu.guo@adelaide.edu.au jiarui.gan@cs.ox.ac.uk conitzer@cs.cmu.edu

Abstract

Assigning routine tasks to cheaper large language model (LLM) providers can reduce generation cost, but switching providers may require processing accumulated context again. Keeping dependent tasks with the same provider can instead reuse eligible context. Graph-based workflows therefore couple routing decisions across tasks. Independent providers may also privately know their real-time unit costs and strategically misreport them. We formulate this routing problem as a single-parameter procurement auction and propose STAR (Strategy-proof Topology-aware Agent Routing), which minimizes virtual-cost-weighted graph workloads. Its allocation rule and payments, computed by integrating workload over possible cost reports, make truthful reporting utility-maximizing regardless of other providers' reports and cover each truthful provider's execution cost. With independent cost distributions and non-decreasing virtual costs, STAR additionally minimizes the user's expected total payment to providers. Controlled synthetic experiments separate the execution-cost benefit of topology awareness from the payment benefit of virtual-cost optimization: the former grows with fan-out and transferred payload, while the latter increases gradually along the tested prior-shape heterogeneity path. A paired sensitivity study quantifies how payment savings vary with prior misspecification while strategy-proofness is preserved.

Keywords: Mechanism design, procurement auctions, strategy-proofness, LLM routing, graph-based agentic workflows, context reuse.

1 Introduction

Large language model (LLM) systems increasingly organize model calls into dependent workflows. Frameworks such as Graph of Thoughts [2], MetaGPT [7], and ChatDev [12] use structured interactions to address complex tasks. Routing can reserve more capable models for demanding tasks and assign routine tasks to cheaper models that meet their requirements, reducing the cost of executing the workflow [3, 11, 20].

We consider a user who needs a workflow executed and selects a model provider for each task node. Providers in-

dependently supply model services and seek to maximize profit. Before they submit bids, the orchestrator identifies the providers that meet each task's requirements and specifies how their inference work is measured. Provider i privately knows its unit cost c_i : the cost of one unit of this measured work. Multiplying c_i by its assigned workload gives its execution cost. The user seeks to minimize the total payment to providers while satisfying the workflow's service requirements. Two features shape this routing problem.

First, a task's cost depends on the providers assigned to neighboring tasks. Its provider must generate output and process the context received from upstream tasks. Keeping adjacent tasks with the same provider can reuse cached input; switching providers requires fresh processing. For example, a coding assistant may use a more capable, more expensive model for planning, switch to a cheaper, less capable model for a routine edit it can handle, and return to the more capable model for verification. With separate serving contexts and no eligible cache reuse for the transferred context, the cheaper model processes the accumulated context, and the more capable model processes the updated context on return. These two context-processing charges can exceed the saving on the routine edit. Globally economical routing therefore balances cheaper task execution against the cost of context switches, especially when a large context feeds many downstream tasks.

Second, providers may privately know their instantaneous unit execution costs. These costs can vary with hardware utilization, available capacity, electricity prices, and local demand.¹ If paid their submitted bids, providers have incentives to quote costs above their actual costs. A truthful mechanism jointly designs task assignments and payments so that reporting the actual cost maximizes each provider's utility.

We propose STAR, a strategy-proof mechanism that assigns workflow tasks and pays providers using their reported unit costs and public graph-dependent workloads. It scores

¹Capacity-sensitive compute pricing is already illustrated by AWS EC2 Spot Instances (<https://aws.amazon.com/ec2/spot/>) and Google Cloud Spot VMs (<https://cloud.google.com/solutions/spot-vms>), which advertise discounts of up to 90% and 91%, respectively, relative to regular on-demand prices.

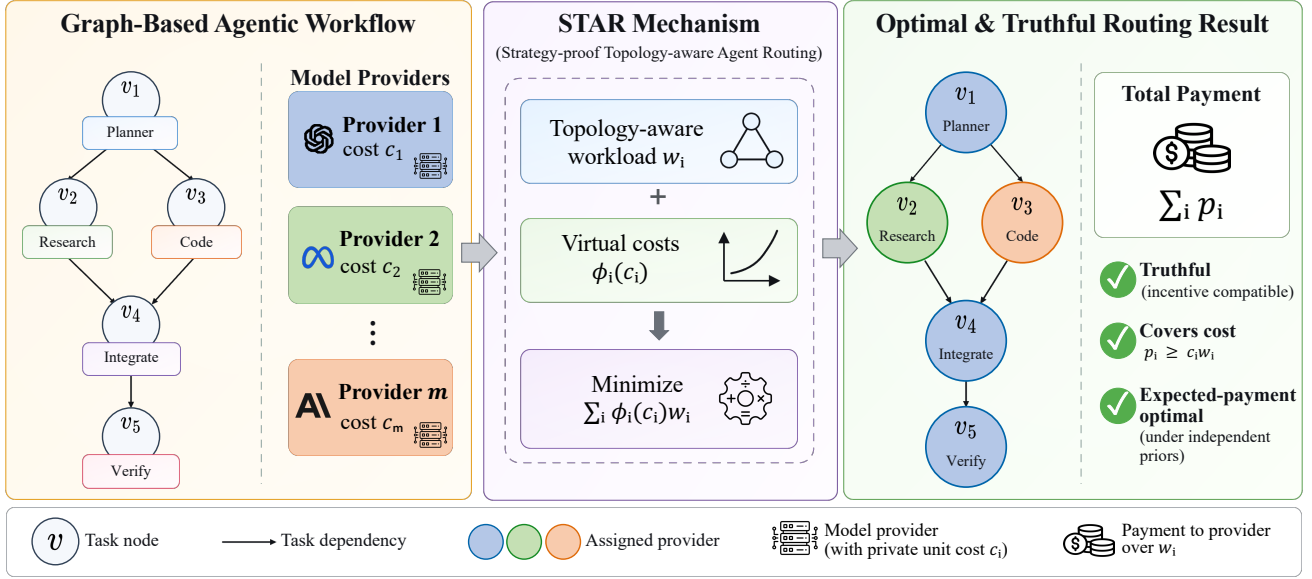


Figure 1: STAR overview. STAR assigns a fixed workflow’s tasks to eligible model providers and determines their payments. Providers privately know unit costs c_i ; assignment-dependent workloads include generation and upstream-context processing with eligible reuse. STAR uses reported costs b_i to optimize virtual-cost-weighted workloads and compute payments that make truthful reporting utility-maximizing and cover truthful providers’ execution costs. The figure uses $w_i = q_i(A)$ and, because of strategy-proofness, depicts truthful cost reports $b_i = c_i$. With independent, correctly specified cost priors and non-decreasing virtual costs, STAR minimizes the user’s expected total payment among truthful, cost-covering mechanisms.

providers by the procurement analogue of Myerson’s virtual value: the *virtual cost* $\phi_i(c) = c + F_i(c)/f_i(c)$, where F_i and f_i are the cumulative distribution function (CDF) and density of provider i ’s cost [9]. Its cost prior is the probability distribution of its private unit cost. The added term accounts for the payment incentive associated with that distribution. Under normalized truthful payments, expected total payment equals expected virtual-cost-weighted workload. Minimizing this objective for each reported cost profile therefore minimizes the user’s expected total payment under independent regular priors, meaning independent cost distributions with non-decreasing virtual costs.

STAR is *dominant-strategy incentive compatible* (DSIC): reporting the true unit cost maximizes a provider’s utility for any reports by the others. This strategy-proofness follows from a simple allocation property: increasing a provider’s reported unit cost can only decrease or preserve its assigned workload. Comparing the optimal assignments before and after the increase proves this property even when neighboring assignments and cache reuse change together. Integrating the allocated workload over possible cost reports yields truthful payments and *ex-post individual rationality* (IR): each truthful provider receives at least its execution cost [1]. Virtual-cost scoring selects the expected-payment-optimal allocation within this truthful framework.

Figure 1 summarizes this direct-report mechanism: providers submit their unit costs, and STAR jointly computes task assignments and payments.

We distinguish two sources of economic improvement.

Topology awareness can reduce realized execution cost by avoiding repeated fresh-input processing. Separately, virtual-cost optimization can reduce the user’s expected payment relative to a graph-aware raw-score truthful mechanism. This second effect reflects differences in providers’ cost distributions: provider-specific priors induce different virtual-cost markups and can change the globally preferred workflow allocation.

Contributions. First, we model task assignment and provider payments through graph-dependent workloads that capture the trade-off between cheaper models and repeated context processing. Second, we establish strategy-proofness and expected-payment optimality using the single-parameter framework, give exact allocation formulations, and derive an efficient payment formula for affine scores. Third, our controlled experiments quantify execution savings from topology awareness, payment savings from heterogeneous priors, and their sensitivity to prior specification, together with the runtime of the complete mechanism.

2 Related Work

We provide a condensed overview here. Appendix A gives a comprehensive discussion.

LLM orchestration and routing. Chain-, tree-, and graph-structured prompting and multi-agent frameworks organize LLM calls into dependent workflows [2, 7, 12, 14, 15, 17]. Cost-aware routers and cascades select among heterogeneous models [3, 11, 20]. MasRouter jointly selects

collaboration modes, roles, and LLMs [18]; BAMAS selects model pools and collaboration topologies under a budget [16]. Our setting adds strategic provider cost reports and truthful payments to routing on a public workflow.

Mechanism design. Our mechanism uses procurement virtual costs from Myerson’s optimal-auction framework [9] and the single-parameter monotonicity/payment characterization of Archer and Tardos [1]. Prior work studies truthful cloud-resource and crowdsourcing procurement [13, 19], and frugal procurement over public set systems including paths and vertex covers [4]. Our model accounts for workload changes across the complete task assignment and optimizes the user’s expected payment. Public workload functions connect these graph interactions to the single-parameter incentive framework.

Context reuse. PagedAttention, SGLang, and prompt caching reduce repeated context processing [6, 8, 21]. We represent the applicable reuse discount by a public edge-level cached-input fraction $\kappa_{u,v}$, determined by cache eligibility and the serving configuration.

3 System Model

3.1 Workflow and Feasible Allocations

We represent a workflow as a finite directed acyclic graph (DAG) $\mathcal{G} = (\mathcal{V}, \mathcal{E})$. Let $v_{\text{user}} \in \mathcal{V}$ be a virtual source node representing the initial user input, and let $\mathcal{V}_T = \mathcal{V} \setminus \{v_{\text{user}}\}$ be the set of task nodes. There are N providers $\mathcal{M} = \{1, \dots, N\}$. For every task node v , the public set $\mathcal{M}_v \subseteq \mathcal{M}$ specifies the providers that are capable of executing v . An allocation is a mapping $A : \mathcal{V}_T \rightarrow \mathcal{M}$ such that $A(v) \in \mathcal{M}_v$ for every $v \in \mathcal{V}_T$. Let $\mathcal{A}$ denote the bid-independent set of all feasible allocations. For notational convenience, we set $A(v_{\text{user}}) = 0$, where $0 \notin \mathcal{M}$. We partition the dependency edges into source and internal edges:

$$\mathcal{E}_{\text{src}} = \{(v_{\text{user}}, v) \in \mathcal{E}\}, \quad \mathcal{E}_{\text{int}} = \mathcal{E} \setminus \mathcal{E}_{\text{src}}. \quad (1)$$

All token quantities below are measured in millions of tokens. Let L_v^{out} denote the expected generated output at task node v , and let $L_{u,v}^{\text{trans}}$ denote the expected payload transmitted along edge (u, v) . In particular, $L_{v_{\text{user}},v}^{\text{trans}}$ is the initial user input provided to a root task.

3.2 Public Graph-Dependent Workloads

For allocation A , define $x_{v,i}(A) = \mathbf{1}\{A(v) = i\}$. We set $x_{v_{\text{user}},i}(A) = 0$. For every edge (u, v) , define

$$\delta_{u,v,i}(A) = x_{v,i}(A)(1 - x_{u,i}(A)). \quad (2)$$

Thus, $\delta_{u,v,i}(A) = 1$ exactly when provider i executes v but does not execute its parent u .

Provider i has a public input-to-output workload coefficient ρ_i . For each edge (u, v) , let $\kappa_{u,v} \in [0, 1]$ denote its public cached-input fraction: $\kappa_{u,v} = 0$ represents free cached input, whereas $\kappa_{u,v} = 1$ represents no same-provider discount. We set $\kappa_{u,v} = 1$ for edges that are not cache eligible.

Running example. Consider a hub h and two children fixed to provider H . Each outgoing edge carries payload L , with discount κ . Assigning h to H contributes $2\rho_H\kappa L$ to the children’s input workload; assigning it to another provider contributes $2\rho_H L$. The saved input work, $2\rho_H(1 - \kappa)L$, must be weighed against the hub’s output generation and incoming-input processing. Proposition 6 accounts for both in the corresponding score threshold.

The total standardized workload assigned to provider i under allocation A is

$$q_i(A) = \sum_{v \in \mathcal{V}_T} L_v^{\text{out}} x_{v,i}(A) + \rho_i \sum_{(u,v) \in \mathcal{E}} L_{u,v}^{\text{trans}} [\kappa_{u,v} x_{v,i}(A) + (1 - \kappa_{u,v}) \delta_{u,v,i}(A)]. \quad (3)$$

For an internal edge whose endpoints are both assigned to provider i , the edge contributes $\rho_i \kappa_{u,v} L_{u,v}^{\text{trans}}$. If the provider changes, the same edge contributes the full fresh-input workload $\rho_i L_{u,v}^{\text{trans}}$. Since the virtual source is not assigned to any provider, every source-to-root payload is charged as fresh input.

Equation (3) measures input-processing work using public cache eligibility and discounts: $\kappa_{u,v} = 1$ charges full fresh-input work, while smaller values represent eligible reuse.

3.3 Private Costs, Reports, and Utilities

Provider i privately knows its unit cost, a single number $c_i \in [a_i, d_i]$ where $[a_i, d_i]$ is its publicly known range of possible costs. This number is the cost per unit of the publicly measured workload $q_i(A)$, so its realized execution cost under allocation A is $C_i(A; c_i) = c_i q_i(A)$. Provider i submits a report $b_i \in [a_i, d_i]$. Given payment $p_i(\mathbf{b})$, its quasi-linear utility is

$$u_i(\mathbf{b}; c_i) = p_i(\mathbf{b}) - c_i q_i(A(\mathbf{b})). \quad (4)$$

The provider’s private information is the single number c_i . Its complete workload function $q_i : \mathcal{A} \rightarrow \mathbb{R}_{\geq 0}$, token quantities, ρ_i , cache eligibility, $\kappa_{u,v}$, and feasible-provider sets $\mathcal{M}_v$ are public and fixed before reports are submitted. These public service specifications define the single-parameter model used throughout the incentive analysis.

4 STAR: Strategy-Proof Topology-Aware Routing

4.1 Score-Based Graph Allocation

We first define a general score-based allocation rule. For each provider i , let $s_i : [a_i, d_i] \rightarrow \mathbb{R}$ be a public non-decreasing score function. Given reports $\mathbf{b} = (b_1, \dots, b_N)$, define

$$A^s(\mathbf{b}) \in \arg \min_{A \in \mathcal{A}} \sum_{j \in \mathcal{M}} s_j(b_j) q_j(A). \quad (5)$$

When multiple allocations attain the same primary objective, the mechanism uses a fixed deterministic tie-breaking order that is independent of all reports.

For STAR, provider costs are independently drawn from provider-specific distributions with CDF F_i and density f_i on $[a_i, d_i]$. The procurement virtual cost is defined as

$$\phi_i(c) = c + \frac{F_i(c)}{f_i(c)}. \quad (6)$$

We assume regularity, so that ϕ_i is non-decreasing. STAR instantiates Eq. (5) with $s_i = \phi_i$:

$$A^{\text{STAR}}(\mathbf{b}) \in \arg \min_{A \in \mathcal{A}} \sum_i \phi_i(b_i) q_i(A). \quad (7)$$

As a special case, for the uniform prior $c_i \sim \text{Uniform}[a_i, d_i]$ used in our reference configuration, $\phi_i(c) = 2c - a_i$.

The term $F_i(c)/f_i(c)$ adjusts the allocation score for the provider's cost distribution. The payment rule determines its realized information rent, the payment in excess of execution cost: $p_i - c_i q_i(A)$.

4.2 Workload Monotonicity

We prove workload monotonicity by comparing the optimal assignments at two reported costs. The graph determines the workload functions and thus the objective minimized in each comparison, incorporating all neighboring assignments and cache effects.

PROPOSITION 1 (Workload Monotonicity). *Fix provider i and the other reports $\mathbf{b}_{-i}$. Suppose that s_i is non-decreasing and that Eq. (5) is solved exactly using the fixed report-independent tie-breaking rule. Then*

$$q_i(A^s(b_i, \mathbf{b}_{-i})) \quad (8)$$

is non-increasing in b_i .

Proof. Take two reports $z < z'$ and let

$$A = A^s(z, \mathbf{b}_{-i}), \quad A' = A^s(z', \mathbf{b}_{-i}).$$

Define the contribution of all other providers as

$$R_i(B) = \sum_{j \neq i} s_j(b_j) q_j(B). \quad (9)$$

Optimality of A at report z gives

$$s_i(z) q_i(A) + R_i(A) \leq s_i(z) q_i(A') + R_i(A'). \quad (10)$$

Optimality of A' at report z' gives

$$s_i(z') q_i(A') + R_i(A') \leq s_i(z') q_i(A) + R_i(A). \quad (11)$$

Adding Eqs. (10) and (11) yields

$$(s_i(z') - s_i(z)) (q_i(A') - q_i(A)) \leq 0. \quad (12)$$

If $s_i(z') > s_i(z)$, then $q_i(A') \leq q_i(A)$. If the two scores are equal, the two optimization problems have the same objective, and the fixed report-independent tie-breaking rule selects the same allocation. Hence the selected workload is non-increasing in the report. $\square$

The proof covers changes to several neighboring assignments at once. Their effects on cache reuse and provider workloads are captured by the functions $q_j(A)$ and the contribution $R_i(A)$ of the other providers.

4.3 Payments, Strategy-Proofness, and Individual Rationality

For fixed $\mathbf{b}_{-i}$, abbreviate $q_i(z) = q_i(A^s(z, \mathbf{b}_{-i}))$. For any public non-decreasing score functions, we compute a provider's payment by adding its reported execution cost to the area under its workload curve over higher possible cost reports:

$$p_i(\mathbf{b}) = b_i q_i(b_i) + \int_{b_i}^{d_i} q_i(z) dz. \quad (13)$$

Equation (13) pays each marginal unit of workload at its critical cost report: the highest reported cost at which that unit remains assigned to the provider. The integral is an area under a workload curve; payments are monetary amounts.

THEOREM 2 (DSIC and Ex-Post Individual Rationality). *Under the conditions of Proposition 1, the score-based allocation rule in Eq. (5) together with the payment rule in Eq. (13) is dominant-strategy incentive compatible and ex-post individually rational.*

Proof. Fix provider i , its true cost c_i , and the other reports $\mathbf{b}_{-i}$. Its truthful utility is

$$u_i(c_i, \mathbf{b}_{-i}; c_i) = \int_{c_i}^{d_i} q_i(z) dz \geq 0, \quad (14)$$

which establishes ex-post individual rationality.

For an alternative report $r_i > c_i$,

$$\begin{aligned} u_i(c_i, \mathbf{b}_{-i}; c_i) - u_i(r_i, \mathbf{b}_{-i}; c_i) \\ = \int_{c_i}^{r_i} (q_i(z) - q_i(r_i)) dz \geq 0, \end{aligned} \quad (15)$$

because q_i is non-increasing. For $r_i < c_i$,

$$\begin{aligned} u_i(c_i, \mathbf{b}_{-i}; c_i) - u_i(r_i, \mathbf{b}_{-i}; c_i) \\ = \int_{r_i}^{c_i} (q_i(r_i) - q_i(z)) dz \geq 0. \end{aligned} \quad (16)$$

Thus truthful reporting maximizes provider i 's utility for every fixed $\mathbf{b}_{-i}$. $\square$

Theorem 2 is the standard single-parameter payment characterization applied to the graph-dependent workload in Eq. (3) [1].

4.4 Expected-Payment Optimality

The preceding truthfulness result applies to any non-decreasing score function. The particular choice $s_i = \phi_i$ provides the additional expected-payment guarantee.

THEOREM 3 (Expected-Payment Optimality). *Suppose costs are independent, provider i has support $[a_i, d_i]$, every virtual cost ϕ_i is non-decreasing, and payments use the normalization $u_i(d_i, \mathbf{c}_{-i}; d_i) = 0$. Then STAR minimizes $\mathbb{E}_{\mathbf{c}} [\sum_i p_i(\mathbf{c})]$ among deterministic DSIC and ex-post individually rational mechanisms whose allocations lie in $\mathcal{A}$.*

Proof. Fix provider i and the other costs $\mathbf{c}_{-i}$, and write $q_i(c) = q_i(A(c, \mathbf{c}_{-i}))$. Independence leaves i 's conditional cost distribution equal to F_i . Taking the expectation of the payment integral and exchanging the order of integration gives

$$\begin{aligned}\mathbb{E}_{c_i} \left[\int_{c_i}^{d_i} q_i(z) dz \right] &= \int_{a_i}^{d_i} q_i(z) \left(\int_{a_i}^z f_i(c) dc \right) dz \\ &= \int_{a_i}^{d_i} F_i(z) q_i(z) dz.\end{aligned}\quad (17)$$

Thus the expected payment, conditional on $\mathbf{c}_{-i}$, is

$$\begin{aligned}\mathbb{E}_{c_i} [p_i(c_i, \mathbf{c}_{-i})] &= \int_{a_i}^{d_i} [c f_i(c) + F_i(c)] q_i(c) dc \\ &= \mathbb{E}_{c_i} [\phi_i(c_i) q_i(c_i)].\end{aligned}\quad (18)$$

Taking expectation over the other costs and summing over providers yields

$$\mathbb{E}_{\mathbf{c}} \left[\sum_i p_i(\mathbf{c}) \right] = \mathbb{E}_{\mathbf{c}} \left[\sum_i \phi_i(c_i) q_i(A(\mathbf{c})) \right].$$

Equation (7) minimizes the expression inside the expectation pointwise for every type profile. Regularity makes the virtual scores non-decreasing, so Proposition 1 and Theorem 2 make this minimizing allocation truthful and individually rational. For any DSIC allocation, the payment characterization adds the provider's utility at its upper cost endpoint d_i to the normalized payment. Ex-post individual rationality makes this boundary utility nonnegative, so setting it to zero minimizes payments for the same allocation. Together these facts establish expected-payment optimality. Appendix B records the normalization details. $\square$

Execution cost, virtual objective, and payment. We distinguish three economic quantities:

$$C_{\text{exec}}(A; \mathbf{c}) = \sum_i c_i q_i(A), \quad (19)$$

$$J_\phi(A; \mathbf{b}) = \sum_i \phi_i(b_i) q_i(A), \quad (20)$$

$$P(\mathbf{b}) = \sum_i p_i(\mathbf{b}). \quad (21)$$

STAR minimizes J_ϕ for each reported profile and hence minimizes the user's expected total payment $\mathbb{E}[P]$ under the correct priors. Execution cost C_{exec} measures providers' resource expenditure, whereas P measures the money the user pays them; the payment includes compensation for execution and incentive rent.

COROLLARY 4 (Raw-Score DSIC). *Setting $s_i(b_i) = b_i$ in Eq. (5) and using Eq. (13) yields a DSIC and ex-post individually rational mechanism. Under truthful reports, its allocation minimizes realized execution cost:*

$$A^{\text{raw}}(\mathbf{c}) \in \arg \min_{A \in \mathcal{A}} \sum_i c_i q_i(A). \quad (22)$$

COROLLARY 5 (Misspecified Monotone Scores). *Any public non-decreasing score functions preserve DSIC and ex-post IR with the payment rule in Eq. (13). Using the true virtual costs additionally gives expected-payment optimality under Theorem 3.*

4.5 Exact Allocation

We give a mixed-integer linear program (MILP) for finite directed acyclic graphs (DAGs), used in all reported experiments, followed by an exact dynamic program for rooted trees and chains. Appendix G extends dynamic programming to graphs of bounded treewidth [5].

Using the edge partition in Eq. (1), let $x_{v,i} \in \{0, 1\}$ indicate that task node $v \in \mathcal{V}_T$ is assigned to provider $i \in \mathcal{M}$. For every edge $(u, v) \in \mathcal{E}$, let $\delta_{u,v,i} \in [0, 1]$ indicate that v is assigned to provider i while its parent u is not assigned to i .

For generic provider scores $s_i(b_i)$, the allocation problem is

$$\begin{aligned}\min_{\mathbf{x}, \delta} \quad & \sum_{i \in \mathcal{M}} s_i(b_i) \left[\sum_{v \in \mathcal{V}_T} L_v^{\text{out}} x_{v,i} \right. \\ & \left. + \rho_i \sum_{(u,v) \in \mathcal{E}} L_{u,v}^{\text{trans}} \left(\kappa_{u,v} x_{v,i} + (1 - \kappa_{u,v}) \delta_{u,v,i} \right) \right]\end{aligned}\quad (23)$$

$$\text{s.t.} \quad \sum_{i \in \mathcal{M}_v} x_{v,i} = 1, \quad v \in \mathcal{V}_T, \quad (24)$$

$$x_{v,i} = 0, \quad v \in \mathcal{V}_T, \quad i \notin \mathcal{M}_v, \quad (25)$$

$$\delta_{u,v,i} \geq x_{v,i} - x_{u,i}, \quad (u, v) \in \mathcal{E}_{\text{int}}, \quad i \in \mathcal{M}, \quad (26)$$

$$\delta_{u,v,i} \leq x_{v,i}, \quad (u, v) \in \mathcal{E}_{\text{int}}, \quad i \in \mathcal{M}, \quad (27)$$

$$\delta_{u,v,i} \leq 1 - x_{u,i}, \quad (u, v) \in \mathcal{E}_{\text{int}}, \quad i \in \mathcal{M}, \quad (28)$$

$$\delta_{v_{\text{user}}, v, i} = x_{v,i}, \quad (v_{\text{user}}, v) \in \mathcal{E}_{\text{src}}, \quad i \in \mathcal{M}, \quad (29)$$

$$x_{v,i} \in \{0, 1\}, \quad v \in \mathcal{V}_T, \quad i \in \mathcal{M},$$

$$0 \leq \delta_{u,v,i} \leq 1, \quad (u, v) \in \mathcal{E}, \quad i \in \mathcal{M}. \quad (30)$$

The objective is the score-weighted workload in Eq. (3); Eqs. (24)–(25) assign each task to exactly one eligible provider. Constraints (26)–(28) enforce $\delta_{u,v,i} = x_{v,i}(1 - x_{u,i})$. Thus a shared provider processes the cached-input fraction $\kappa_{u,v}$, while a provider switch charges full fresh-input workload. Equation (29) charges every source-to-root payload as fresh input.

STAR uses $s_i(b_i) = \phi_i(b_i)$ and Raw-DSIC uses $s_i(b_i) = b_i$. Both optimize the same feasible assignments and graph-dependent workloads, differing only in their provider scores.

Equal-objective allocations are resolved by the lexicographic order of public node ordering and provider identifiers. This tie order is fixed before reports are submitted.

All reported allocations are computed with the COIN-OR Branch-and-Cut (CBC) optimization solver through the Python modeling library PuLP, requesting zero relative and absolute optimality gap. We retain solutions with solver status `Optimal`, using documented floating-point tolerances.

Exact score minimization supplies the workload monotonicity required by Proposition 1; an approximation-based extension would likewise need a monotone allocation rule to preserve strategy-proofness.

Exact optimization on rooted trees. For rooted trees and chains, exact dynamic programming combines optimal child assignments from leaves to root, accounting for their connecting edges and fresh user input. It computes the optimal objective in $O(|\mathcal{V}_T|N^2)$ operations; Appendix F gives the recurrence, correctness proof, assignment reconstruction, and extension to forests.

4.6 A Computational Special Case: Affine Scores

For affine scores, which are linear functions of the reported cost plus a constant, $s_i(b) = \alpha_i b + \gamma_i$ with $\alpha_i > 0$, define

$$J_i(t; \mathbf{b}_{-i}) = \min_{A \in \mathcal{A}} \left[tq_i(A) + \sum_{j \neq i} s_j(b_j)q_j(A) \right]. \quad (31)$$

The payment in Eq. (13) then equals

$$p_i(\mathbf{b}) = b_i q_i(A^s(\mathbf{b})) + \frac{J_i(s_i(d_i); \mathbf{b}_{-i}) - J_i(s_i(b_i); \mathbf{b}_{-i})}{\alpha_i}. \quad (32)$$

This formula evaluates the payment integral through repeated optimization of the same task-assignment problem. We first find the optimal assignment at the submitted costs; its objective value is reused for every provider’s payment. For each provider assigned positive workload, we then recompute the optimum with that provider’s report set to its upper endpoint d_i , holding the other reports fixed. Thus all payments need one shared optimization and at most one additional optimization per active provider, plus tie-breaking computations. Raw, uniform, and power-prior scores have slopes 1, 2, and $1 + 1/k_i$, respectively. Appendix C proves the optimal-value identity and specifies these score families.

4.7 When Topology Awareness Matters

PROPOSITION 6 (Broadcaster Threshold). *Consider a hub node h with m outgoing edges of equal payload L^{trans} and cached-input fraction κ to children assigned to provider H . Hold every other task’s assignment fixed, and compare assigning h to eligible providers L and H , with scores s_L and s_H . Let I_L and I_H be the score-weighted costs of processing all input arriving at h under these two choices, including their respective cache discounts, and write $\Delta_{\text{in}} = I_H - I_L$. Assigning h to H is strictly preferred exactly when*

$$m(1 - \kappa)\rho_H L^{\text{trans}} s_H > L_h^{\text{out}}(s_H - s_L) + \Delta_{\text{in}}. \quad (33)$$

Proof. Assigning h to L contributes $s_L L_h^{\text{out}} + m\rho_H L^{\text{trans}} s_H + I_L$, whereas assigning it to H contributes $s_H L_h^{\text{out}} + m\kappa\rho_H L^{\text{trans}} s_H + I_H$. The latter is smaller exactly under Eq. (33). $\square$

The left side is the saved input-processing cost at the children; the right side is the extra generation and incoming-input cost at the hub. When h has no incoming input, $\Delta_{\text{in}} = 0$.

At equality, the fixed tie-breaking rule applies. The outgoing-context saving grows linearly with fan-out and payload and is zero at $\kappa = 1$. Both STAR and graph-aware raw-score routing capture this trade-off; virtual-cost scoring further accounts for provider-specific payment incentives.

5 Experiments

We validate the incentive and payment implementation, separate execution-cost gains from payment gains, and test prior-shape heterogeneity, deliberate prior misspecification, and payload/cache controls. Appendix I.7 reports computational costs.

5.1 Experimental Setup

Structural-prior stress suite. Our primary payment study uses a frozen broadcaster-market construction and three pre-specified regular-prior profiles. Each provider follows the power family

$$F_i(c) = \left(\frac{c - a_i}{d_i - a_i} \right)^{k_i}, \quad c \in [a_i, d_i], \quad (34)$$

whose virtual cost is given in Appendix C. The shape parameter $k_i > 0$ controls how probability is distributed within the provider’s cost range: $k_i = 1$ is uniform, $k_i > 1$ places more probability near the upper endpoint, and $k_i < 1$ places more near the lower endpoint. The mild and strong profiles use fixed provider-specific k_i vectors. For each profile, we evaluate the same 24 pre-specified structural cells with 50 paired holdout trials per cell. Provider cost ranges, shape vectors, capability tiers, and the structural grid are specified in Appendix I.2.

Scope and workflows. Four hand-constructed synthetic DAG families inspired by creative writing, software engineering, web development, and legal analysis provide the S1–S4 reference. They vary fan-out, fan-in, and high-bandwidth broadcaster motifs, with eligibility specified by manually assigned capability tiers. Appendix H gives each workflow’s graph and task requirements.

Synthetic workflow reference. S1–S4 use nine providers in three capability tiers, with fixed base-cost anchors in a 1:5:25 progression and uniform cost draws within provider-specific ranges. We set $\rho_i = 1/3$, $\kappa_{u,v} = 0.1$ on cache-eligible edges, and $\kappa_{u,v} = 1$ otherwise. Nominal token quantities receive lognormal variation with $\sigma = 0.3$; the S2 source context follows a truncated-Pareto distribution on 5k–100k tokens. Appendix H gives the cost and token distributions and base-cost anchors.

Mechanisms and routing diagnostics. STAR minimizes the full graph-dependent virtual-cost objective and uses Eq. (13), evaluated by Eq. (32) for affine scores. Raw-DSIC minimizes $\sum_i b_i q_i(A)$ over the same allocation space and applies the corresponding payment rule; under truthful reports, its allocation minimizes realized execution cost and coincides with the true-cost oracle. The True-Cost Oracle directly

Prior profile	Reduction range	Positive cells
Uniform	1.49%–3.34%	2/24
Heterogeneous mild	10.23%–10.30%	24/24
Heterogeneous strong	21.99%–31.48%	24/24

Table 1: STAR payment reductions across prior profiles. Ranges are the minimum and maximum point estimates over all 24 pre-specified structural cells; all cells are retained. A positive cell has a paired-bootstrap 95% interval strictly above zero.

minimizes $\sum_i c_i q_i(A)$, serving as an execution-cost benchmark. Myopic independently selects the cheapest feasible provider at each node. We compare STAR and Raw-DSIC by payments to providers and execution costs. The true-cost oracle provides the execution-cost benchmark, while Myopic is evaluated by execution cost only.

Metrics. We separately report realized execution cost $C_{\text{exec}} = \sum_i c_i q_i(A)$, procurement payment $P = \sum_i p_i$, and information rent $R = P - C_{\text{exec}}$. If C_{oracle} denotes the true-cost oracle objective, then

$$P_{\text{STAR}} - C_{\text{oracle}} = \underbrace{C_{\text{STAR}} - C_{\text{oracle}}}_{\text{allocation distortion}} + \underbrace{P_{\text{STAR}} - C_{\text{STAR}}}_{\text{information rent}}. \quad (35)$$

We use the same random draws for each mechanism in a comparison and report paired-bootstrap 95% confidence intervals (CIs).

5.2 Implementation Validation

The theorems establish DSIC and ex-post individual rationality; exhaustive small-instance enumeration and dense bid sweeps validate the implementation. The final suite contains 159 passing tests. Across all reported exact experiment rows, CBC returns status `Optimal` with zero reported gap; when providers report their actual costs, Raw-DSIC assigns each task to exactly the provider selected by the execution-cost oracle; the minimum truthful provider utility is zero; and all payment-decomposition checks pass up to floating-point tolerance. The monotonicity case study evaluates 901 bids and contains four workload plateaus. Workload never increases with the bid, and no sampled misreport gives higher utility than reporting the actual cost.

5.3 Payment Savings under Heterogeneous Regular Priors

We first isolate the condition that distinguishes STAR from Raw-DSIC: provider-specific regular cost distributions. Both mechanisms optimize the same graph-dependent workload and use the DSIC payment rule, with Raw-DSIC scoring provider i by b_i and STAR scoring it by $\phi_i(b_i)$. We evaluate three pre-specified prior profiles over the same 24 frozen structural cells, using 50 paired holdout trials per cell and retaining every cell regardless of effect size.

Table 1 shows consistent payment savings across all 24 structural cells for the tested mild and strong prior-shape

profiles. Under the strong profile, STAR reduces payment relative to Raw-DSIC by 21.99%–31.48%, and all 24 paired 95% confidence intervals lie strictly above zero. The mild profile yields reductions of 10.23%–10.30%, again with positive intervals in all 24 cells. The uniform profile yields 1.49%–3.34%, with a strictly positive interval in 2 of 24 cells.

In the representative degree-100 high-payload cell with $\rho = 1/3$, the payment reductions are 1.49% with 95% CI: $[-0.39\%, 3.58\%]$ under the uniform profile, 10.23%, $[3.44\%, 16.73\%]$ under the mild profile, and 31.48%, $[22.56\%, 40.73\%]$ under the strong profile. The fixed synthetic distributions enable a paired comparison of prior shape while preserving the same graph and workload rules.

5.4 Continuous Prior-Shape Heterogeneity

To test whether the heterogeneous-prior gain depends on a selected endpoint, we hold each provider’s support $[a_i, d_i]$ fixed and vary only its regular prior shape. For $\lambda \in \{0, 0.1, \dots, 1\}$, define

$$k_i(\lambda) = \left(k_i^{\text{strong}}\right)^\lambda, \quad (36)$$

$$F_i(c; \lambda) = \left(\frac{c - a_i}{d_i - a_i}\right)^{k_i(\lambda)}.$$

The corresponding virtual cost is

$$\phi_i(c; \lambda) = \left(1 + \frac{1}{k_i(\lambda)}\right)c - \frac{a_i}{k_i(\lambda)}. \quad (37)$$

We use 200 paired holdout trials per λ and common latent cost quantiles throughout the sweep.

Figure 2B shows a gradual increase in payment savings across the prior-shape path. The payment reduction rises from 2.11% at $\lambda = 0$ to 27.22% at $\lambda = 1$. The point estimate increases at all ten adjacent grid steps and forms an approximately smooth curve, with the final step adding 2.14 percentage points.

For each cost draw, a rank reversal occurs when a pair of providers eligible for the same task have opposite orderings under raw and virtual costs. Counting each eligible pair once, the fraction with a reversal increases from 0.072 to 0.171 along the path. Across the eleven grid points, payment reduction has Spearman correlation 0.998 with this rate and 0.936 with the fraction of draws in which STAR and Raw-DSIC choose different task assignments. These descriptive associations summarize how allocation decisions and payment savings co-vary along the tested synthetic path.

At $\lambda = 0$, all providers have uniform-shaped priors with different lower support endpoints a_i . The virtual scores $\phi_i(c) = 2c - a_i$ therefore contain provider-specific offsets, which can change the preferred allocation and produce a nonzero STAR/Raw-DSIC difference at this endpoint.

At $\lambda = 1$, mean (P, C_{exec}, R) is (1.8382, 1.4530, 0.3851) for STAR and (2.5256, 1.2587, 1.2670) for Raw-DSIC; the oracle execution cost is 1.2587. Hence the absolute payment difference is 0.6875: STAR reduces information rent by 0.8818

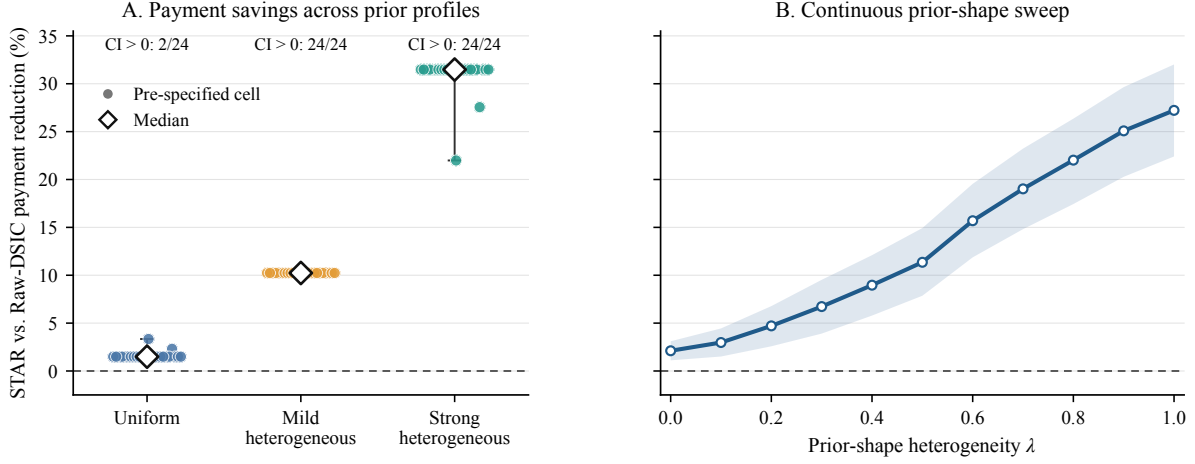


Figure 2: Payment savings under heterogeneous regular priors. A: Each point is one of the 24 pre-specified structural cells for the indicated prior profile; the larger marker and range summarize the cell-level point estimates, and the annotation reports how many paired 95% intervals are strictly above zero. B: In the representative degree-100 high-payload graph, STAR’s payment reduction along the continuous prior-shape path. Provider supports, graph semantics, and latent cost quantiles remain fixed; the band denotes a paired-bootstrap 95% confidence interval.

True cost distribution	Correct-prior STAR reduction (95% CI)	Negative-direction error $ \delta $	Positive-direction error $\delta = - \delta $ (95% CI)	Positive-direction error $\delta = + \delta $ (95% CI)
Uniform	0.79% [0.31,1.25]	0.05	0.85% [0.28,1.38]	0.72% [0.26,1.15]
		0.1	0.71% [0.07,1.32]	0.56% [0.10,0.97]
		0.2	0.34% [-0.54,1.16]	0.20% [-0.31,0.69]
Heterogeneous power priors	30.04% [27.88,32.21]	0.05	30.01% [27.84,32.14]	29.98% [27.79,32.09]
		0.1	30.15% [28.02,32.29]	29.92% [27.76,32.03]
		0.2	30.16% [28.06,32.27]	30.13% [27.93,32.23]

Table 2: User-payment savings (%) under prior-estimation errors. All three result columns and their bracketed paired 95% bootstrap confidence intervals show the user’s total-payment savings relative to Raw-DSIC in percent: $100(1 - \bar{P}_{\text{STAR}}/\bar{P}_{\text{Raw}})\%$. Positive values mean lower user payments. The perturbation magnitude $|\delta|$ is dimensionless. Only assumed shapes change: $\hat{k}_i = k_i e^{\delta r_i}$, with fixed $r = (1, -1, 0, 1, -1)$. Uniform uses $k_i = 1$ for every provider; heterogeneous power priors use $k = (0.5, 2, 0.2, 5, 1)$. Each three-row group shares 1,000 cost draws and correct-prior and Raw-DSIC baselines; the merged correct-prior entry applies to all three magnitudes. True distributions, graph, and cost ranges stay fixed. Appendix I.7 provides payment details.

while increasing execution cost by 0.1944. The payment saving is primarily lower incentive rent, partially offset by allocation distortion.

The degree-10, degree-20, and degree-100 continuum instances replicate the same consumer structure. Their coincident normalized payment-reduction and unique-pair rank-reversal curves provide a consistent view of the prior-shape sweep at three scales. Increasing degree scales the absolute saving from 0.276 at degree 10 to 0.687 at degree 100 when $\lambda = 1$.

5.5 Sensitivity to Prior-Estimation Error Magnitude

We test how STAR’s payment advantage changes as errors in estimated provider cost distributions increase. True costs follow the power family in Eq. (34): the uniform profile has

$k_i = 1$ for every provider, while the heterogeneous power-prior profile has provider-specific shapes $k = (0.5, 2, 0.2, 5, 1)$ (Appendix I.2). We change only the assumed shapes, $\hat{k}_i = k_i e^{\delta r_i}$, with fixed $r = (1, -1, 0, 1, -1)$ and both signs of $|\delta| \in \{0.05, 0.1, 0.2\}$. These magnitudes permit shape increases of at most 5.13%, 10.52%, and 22.14%, respectively. We pre-specify all six perturbations and 1,000 new cost vectors per true profile. Each condition shares the same true costs, truthful reports, graph, eligibility, and cost ranges with correct-prior STAR and Raw-DSIC.

Table 2 reports correct-prior savings of 0.79% (uniform) and 30.04% (heterogeneous). Under heterogeneous power priors, all six perturbations retain payment reductions of 29.92%–30.16%, with every paired 95% interval strictly above zero. Under uniform priors, all six sample-mean reductions

Control	Execution cost reduction	Paired 95% CI
Low payload (ratio 1)	2.13%	[1.16%, 3.24%]
Higher payload (ratio 16)	67.47%	[66.12%, 68.48%]
No cache advantage ($\kappa = 1$)	0.00%	[0.00%, 0.00%]

Table 3: Degree-100 fan-out controls ($\rho = 1/3$, uniform priors). Ratio is transferred context per edge divided by each child’s 20,000-token output: ratios 1 and 16 transfer 20,000 and 320,000 tokens, respectively, with $\kappa = 0.1$; the no-cache row transfers 1,000,000 tokens with $\kappa = 1$. Hub output is 1,000,000 tokens throughout. Table 8 instead uses ratio 50 and $\kappa = 0$. Reductions compare graph-aware Raw-DSIC execution cost with Myopic; intervals are paired-bootstrap 95% confidence intervals. The $\kappa = 1$ zeros are exact, not rounded: without reuse discounts and with common ρ , Raw-DSIC and Myopic attain the same execution cost.

remain positive: 0.72%–0.85% at $|\delta| = 0.05$, 0.56%–0.71% at $|\delta| = 0.1$, and 0.20%–0.34% at $|\delta| = 0.2$. Both directions at $|\delta| = 0.05$ and 0.1 have intervals strictly above zero. At $|\delta| = 0.2$, both intervals span zero, indicating less precise evidence for these smaller savings. The grid characterizes the payment benefit and its precision as prior-estimation error grows. All tested assumed scores are increasing and used consistently in allocation and payments, preserving strategy-proofness. Correctly specified priors additionally yield the expected-payment optimum in Theorem 3. At $|\delta| = 0.2$, the entire task assignment matches correct-prior routing in at least 93% of uniform draws and 96.9% of heterogeneous draws in each error direction. Appendix I.7 reports the full assignment-stability and payment-decomposition results.

5.6 Topology Stress Tests and Structural Controls

Table 3 isolates the roles of payload and reuse at fixed degree 100 and $\rho = 1/3$, matching Proposition 6. Raising the transferred-context/child-output ratio from one to sixteen increases execution-cost reduction from 2.13% to 67.47%; removing reuse with $\kappa = 1$ makes it exactly zero. The paired intervals identify repeated fresh-input processing as the source of the large topology gains.

Topology awareness and virtual-cost optimization generate distinct effects. In the tested degree-100 high-payload fan-out construction, graph-aware raw-cost routing reduces realized execution cost by 90.20%–96.91% relative to Myopic routing, depending on ρ and the prior profile. Comparing STAR with the already topology-aware Raw-DSIC then isolates the additional payment benefit of provider-specific virtual-cost scoring.

Appendices I.1, I.3, I.4, and I.6 report the synthetic S1–S4 uniform-prior reference, the complete stress grid, prior-continuum diagnostics, and the support-width comparison, respectively.

6 Conclusion and Future Directions

STAR combines topology-aware workflow assignment with truthful provider payments. Workload monotonicity and the workload-integrating payment rule establish DSIC and ex-post IR; independent, correctly specified regular cost priors yield expected-payment optimality. Controlled synthetic experiments distinguish execution savings from avoided context processing and payment savings from virtual costs, with prior-shape sweeps and paired misspecification tests characterizing the latter.

Affine scores enable efficient payment computation. Learned cost priors, measured serving workloads, and richer private service attributes provide directions for future work.

Acknowledgments

Jiayuan Liu and Vincent Conitzer thank the Cooperative AI Foundation, Macroscopic Ventures (formerly Polaris Ventures / the Center for Emerging Risk Research) and Jaan Tallinn’s donor-advised fund at Founders Pledge for financial support.

References

- [1] A. Archer and É. Tardos. Truthful mechanisms for one-parameter agents. In *Proceedings 42nd IEEE Symposium on Foundations of Computer Science*, pages 482–491. IEEE, 2001.
- [2] M. Besta, N. Blach, A. Kubicek, R. Gerstenberger, M. Podstawski, L. Gianinazzi, J. Gajda, T. Lehmann, H. Niewiadomski, P. Nyczyk, et al. Graph of thoughts: Solving elaborate problems with large language models. In *Proceedings of the AAAI conference on artificial intelligence*, volume 38, pages 17682–17690, 2024.
- [3] L. Chen, M. Zaharia, and J. Zou. FrugalGPT: How to use large language models while reducing cost and improving performance. *Transactions on Machine Learning Research*, 2024. ISSN 2835-8856. URL <https://openreview.net/forum?id=cSimKw5p6R>. Featured Certification.
- [4] N. Chen, E. Elkind, N. Gravin, and F. Petrov. Frugal mechanism design via spectral techniques. In *2010 IEEE 51st Annual Symposium on Foundations of Computer Science*, pages 755–764. IEEE, 2010.
- [5] M. Cygan, F. V. Fomin, Ł. Kowalik, D. Lokshtanov, D. Marx, M. Pilipczuk, M. Pilipczuk, and S. Saurabh. *Parameterized algorithms*, volume 5. Springer, 2015.
- [6] I. Gim, G. Chen, S.-s. Lee, N. Sarda, A. Khandelwal, and L. Zhong. Prompt cache: Modular attention reuse for low-latency inference. *Proceedings of Machine Learning and Systems*, 6:325–338, 2024.
- [7] S. Hong, M. Zhuge, J. Chen, X. Zheng, Y. Cheng, J. Wang, C. Zhang, Z. Wang, S. K. S. Yau, Z. Lin, et al. Metagpt: Meta programming for a multi-agent collaborative framework. In *The Twelfth International Conference on Learning Representations*, 2023.
- [8] W. Kwon, Z. Li, S. Zhuang, Y. Sheng, L. Zheng, C. H. Yu, J. Gonzalez, H. Zhang, and I. Stoica. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the 29th symposium on operating systems principles*, pages 611–626, 2023.
- [9] R. B. Myerson. Optimal auction design. *Mathematics of operations research*, 6(1):58–73, 1981.
- [10] N. Nisan, T. Roughgarden, E. Tardos, and V. V. Vazirani, editors. *Algorithmic Game Theory*. Cambridge University Press, 2007.
- [11] I. Ong, A. Almahairi, V. Wu, W.-L. Chiang, T. Wu, J. E. Gonzalez, M. W. Kadous, and I. Stoica. RouteLLM: Learning to route LLMs with preference data. *arXiv preprint arXiv:2406.18665*, 2024.
- [12] C. Qian, W. Liu, H. Liu, N. Chen, Y. Dang, J. Li, C. Yang, W. Chen, Y. Su, X. Cong, et al. Chatdev: Communicative agents for software development. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 15174–15186, 2024.
- [13] A. Singla and A. Krause. Truthful incentives in crowdsourcing tasks using regret minimization mechanisms. In *Proceedings of the 22nd international conference on World Wide Web*, pages 1167–1178, 2013.
- [14] J. Wei, X. Wang, D. Schuurmans, M. Bosma, F. Xia, E. Chi, Q. V. Le, D. Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 35, pages 24824–24837, 2022.
- [15] Q. Wu, G. Bansal, J. Zhang, Y. Wu, B. Li, E. Zhu, L. Jiang, X. Zhang, S. Zhang, J. Liu, et al. Autogen: Enabling next-gen LLM applications via multi-agent conversations. In *First Conference on Language Modeling*, 2024.
- [16] L. Yang, J. Luo, X. Liu, Y. Lou, and Z. Chen. Bamas: Structuring budget-aware multi-agent systems. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pages 29802–29810, 2026.
- [17] S. Yao, D. Yu, J. Zhao, I. Shafran, T. Griffiths, Y. Cao, and K. Narasimhan. Tree of thoughts: Deliberate problem solving with large language models. *Advances in neural information processing systems*, 36:11809–11822, 2023.
- [18] Y. Yue, G. Zhang, B. Liu, G. Wan, K. Wang, D. Cheng, and Y. Qi. Masrouter: Learning to route llms for multi-agent systems. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 15549–15572, 2025.
- [19] S. Zaman and D. Grosu. A combinatorial auction-based mechanism for dynamic vm provisioning and allocation in clouds. *IEEE Transactions on Cloud Computing*, 1(2): 129–141, 2013.
- [20] Y.-K. Zhang, S. Lu, Q.-G. Chen, W. Luo, D.-C. Zhan, and H.-J. Ye. Let the LLM stick to its strengths: Learning to route economical LLM. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025.
- [21] L. Zheng, L. Yin, Z. Xie, J. Huang, C. Sun, C. Yu, S. Cao, C. Kozyrakis, I. Stoica, J. E. Gonzalez, et al. Efficiently programming large language models using sglang. 2023.

A Detailed Related Work

Structured LLM workflows and economical routing. Chain-, tree-, and graph-based reasoning and multi-agent systems structure dependent model calls [2, 7, 12, 14, 15, 17]. Frugal-GPT, RouteLLM, and economical model routing select among models using performance and cost information [3, 11, 20]. Building on this motivation for economical model selection, our formulation incorporates independently compensated providers and incentives for truthful cost reporting.

Joint multi-agent configuration. MasRouter selects collaboration modes, roles, and LLMs in a cascaded router [18]. BAMAAS jointly selects a model pool and collaboration topology subject to a budget [16]. These systems demonstrate joint optimization of multi-agent configurations. Our complementary procurement setting fixes the workflow and eligibility sets before bidding, then uses strategic provider cost reports to determine task assignments and payments.

Single-parameter procurement. Our incentive analysis applies the single-parameter characterization for providers whose private information is one unit-cost value [1]. For any bid-independent feasible set and public quantities $q_i(A)$, exact minimization of $\sum_i s_i(b_i)q_i(A)$ with monotone scores and fixed tie-breaking yields workload monotonicity. Public quantity functions incorporate graph interactions directly into this argument. Procurement virtual costs then provide the expected-payment objective under independent regular priors [9, 10]. We develop the graph-dependent workload formulation, exact allocation methods, payment computation, and economic comparisons within this framework.

Graph and infrastructure procurement. Truthful cloud-resource allocation and crowdsourcing procurement study related incentive problems [13, 19]. Frugal mechanism design on public set systems includes path and vertex-cover procurement [4]; its overpayment benchmarks and worst-case frugality objectives differ from our correct-prior expected-payment objective. A provider here can execute multiple tasks, and its public amount of work changes with neighboring assignments and input-reuse terms. Our public quantity functions represent this allocation-dependent work while preserving a single private unit cost per provider. The same framework also accommodates public setup- and placement-dependent workload rules.

Context processing and caching. PagedAttention, SGLang, and prompt caching motivate attention to context-processing work [6, 8, 21]. Actual reuse depends on the model, serving instance, cache state, and token-prefix structure. Our workload model encodes the applicable eligibility and discount in public edge parameters, with $\kappa_{u,v} = 1$ charging fresh input. The synthetic experiments fix these parameters to isolate the routing and payment effects of repeated context processing.

B Expected-Payment Identity

Section 4.4 proves the expected-payment identity by exchanging the order of integration in the payment rule. This

appendix records how the choice of upper-type utility enters that identity. For fixed other types $\mathbf{c}_{-i}$, write $q_i(z) = q_i(A(z, \mathbf{c}_{-i}))$ and let $U_i^{\text{top}}(\mathbf{c}_{-i})$ be provider i 's utility when its true cost and report are both d_i . The single-parameter payment characterization gives

$$p_i(c_i, \mathbf{c}_{-i}) = c_i q_i(c_i) + \int_{c_i}^{d_i} q_i(z) dz + U_i^{\text{top}}(\mathbf{c}_{-i}). \quad (38)$$

Ex-post individual rationality implies $U_i^{\text{top}}(\mathbf{c}_{-i}) \geq 0$. The normalized rule sets it to zero and minimizes payment for the given allocation. With a general upper-type utility, the identity proved in the main text becomes $\mathbb{E}[p_i] = \mathbb{E}[\phi_i(c_i)q_i(A(\mathbf{c}))] + \mathbb{E}[U_i^{\text{top}}(\mathbf{c}_{-i})]$. This separates the allocation-dependent virtual-cost term from the nonnegative normalization term.

C Derivation of the Affine-Score Payment Shortcut

Equation (13) defines the general truthful payment for any monotone score-based allocation rule. We now give a computational specialization for the case in which provider i 's score is affine in its report: $s_i(b) = \alpha_i b + \gamma_i$, $\alpha_i > 0$. Affine scores permit the payment integral to be evaluated using optimal-value differences, while Theorem 2 establishes the incentive guarantee for general monotone scores. For fixed $\mathbf{b}_{-i}$, define

$$J_i(t; \mathbf{b}_{-i}) = \min_{A \in \mathcal{A}} [t q_i(A) + \sum_{j \neq i} s_j(b_j) q_j(A)]. \quad (39)$$

Then the general integral payment in Eq. (13) has the exact affine-score representation

$$p_i(\mathbf{b}) = b_i q_i(A^s(\mathbf{b})) + \frac{J_i(s_i(d_i); \mathbf{b}_{-i}) - J_i(s_i(b_i); \mathbf{b}_{-i})}{\alpha_i}. \quad (40)$$

All score families used in our reported experiments are affine. In particular:

$$\text{Raw score: } s_i(b) = b, (\alpha_i, \gamma_i) = (1, 0), \quad (41)$$

$$\text{Uniform prior: } \phi_i(b) = 2b - a_i, (\alpha_i, \gamma_i) = (2, -a_i), \quad (42)$$

$$\begin{aligned} \text{Power prior: } \phi_i(b) &= \left(1 + \frac{1}{k_i}\right) b - \frac{a_i}{k_i}, \\ (\alpha_i, \gamma_i) &= \left(1 + \frac{1}{k_i}, -\frac{a_i}{k_i}\right). \end{aligned} \quad (43)$$

The allocation is solved once at the submitted profile. For each provider with positive allocated workload, we solve one additional allocation problem with that provider's score set to its upper-support value and apply Eq. (32). If the current workload is zero, monotonicity implies that all higher-report workloads are also zero, so the provider's normalized payment is zero.

For a general regular prior with a non-affine virtual cost, Eq. (13) directly supplies the truthful payment used in Theorems 2 and 3. Our implementation and reported experiments

use the affine families above, enabling the optimal-value shortcut in Eq. (32).

The main paper defines the general normalized payment as

$$p_i(\mathbf{b}) = b_i q_i(A^s(\mathbf{b})) + \int_{b_i}^{d_i} q_i(A^s(z, \mathbf{b}_{-i})) dz. \quad (44)$$

This section derives a computational shortcut for the special case

$$s_i(b) = \alpha_i b + \gamma_i, \quad \alpha_i > 0. \quad (45)$$

For fixed $\mathbf{b}_{-i}$, define

$$J_i(t; \mathbf{b}_{-i}) = \min_{A \in \mathcal{A}} \left[t q_i(A) + \sum_{j \neq i} s_j(b_j) q_j(A) \right]. \quad (46)$$

As the pointwise minimum of affine functions of t , J_i is concave, piecewise linear, and absolutely continuous. At almost every t , its derivative is the workload of an optimal allocation:

$$\frac{\partial J_i(t; \mathbf{b}_{-i})}{\partial t} = q_i(A_i(t; \mathbf{b}_{-i})), \quad (47)$$

where

$$A_i(t; \mathbf{b}_{-i}) \in \arg \min_{A \in \mathcal{A}} \left[t q_i(A) + \sum_{j \neq i} s_j(b_j) q_j(A) \right].$$

The fixed report-independent tie rule determines the selected optimizer at breakpoints; these finitely many points do not affect the integral.

Therefore,

$$\begin{aligned} J_i(s_i(d_i); \mathbf{b}_{-i}) - J_i(s_i(b_i); \mathbf{b}_{-i}) &= \int_{s_i(b_i)}^{s_i(d_i)} q_i(A_i(t; \mathbf{b}_{-i})) dt \\ &= \alpha_i \int_{b_i}^{d_i} q_i(A^s(z, \mathbf{b}_{-i})) dz, \end{aligned} \quad (48)$$

where the final equality uses $t = s_i(z) = \alpha_i z + \gamma_i$ and $dt = \alpha_i dz$. Substituting Eq. (48) into Eq. (44) yields

$$p_i(\mathbf{b}) = b_i q_i(A^s(\mathbf{b})) + \frac{J_i(s_i(d_i); \mathbf{b}_{-i}) - J_i(s_i(b_i); \mathbf{b}_{-i})}{\alpha_i}. \quad (49)$$

This identity evaluates the general truthful payment exactly for affine scores. For arbitrary monotone scores, the workload integral in Eq. (44) remains the applicable payment formula.

D Relation of Raw-Score DSIC to Vickrey–Clarke–Groves Payments

The raw-score mechanism selects

$$A^{\text{raw}}(\mathbf{b}) \in \arg \min_{A \in \mathcal{A}} \sum_i b_i q_i(A) \quad (50)$$

and uses the normalized single-parameter integral payment. The Vickrey–Clarke–Groves (VCG) rule compensates

a provider according to the reduction in the other providers' reported costs made possible by its participation.

When provider i can be excluded and reporting its upper type induces zero workload,

$$q_i(A^{\text{raw}}(d_i, \mathbf{b}_{-i})) = 0, \quad (51)$$

the payment can be written as

$$\begin{aligned} p_i^{\text{raw}}(\mathbf{b}) &= \min_{A: q_i(A)=0} \sum_{j \neq i} b_j q_j(A) \\ &\quad - \sum_{j \neq i} b_j q_j(A^{\text{raw}}(\mathbf{b})), \end{aligned} \quad (52)$$

which is the Clarke-pivot VCG payment for the efficient allocation.

A provider may also retain unavoidable workload at its upper type, for example when it is the sole feasible provider for a task. The top-type-normalized payment accommodates this case and preserves DSIC and ex-post individual rationality, using the upper-type assignment as its normalization reference.

E Equivalent Breakpoint Representation

We give an equivalent mathematical breakpoint representation of the general workload-integrating payment. The reported affine-score experiments compute the same payment through optimal-value differences.

For fixed $\mathbf{b}_{-i}$, the workload $q_i(z)$ is a non-increasing step function. When $b_i < d_i$, partition the integration range at every interior workload change:

$$b_i = t_0 < t_1 < \dots < t_K = d_i.$$

Let Q_k be the constant workload on the open interval (t_{k-1}, t_k) . Then Eq. (13) becomes

$$p_i(\mathbf{b}) = b_i q_i(b_i) + \sum_{k=1}^K Q_k (t_k - t_{k-1}). \quad (53)$$

The first term uses the workload actually selected by the fixed tie-breaking rule at the current report, including when b_i is a change point. Values at the other individual change points and at d_i do not affect the integral. When $b_i = d_i$, the integration range is empty and the payment is $d_i q_i(d_i)$. Workload changes can arise from task reassignment or cache-status changes; the number of intervals need not equal the number of tasks currently assigned to provider i .

F Exact Dynamic Programming for Rooted Trees

Section 4.5 uses an exact dynamic program for rooted trees and chains. We give its value recurrence and correctness proof before detailing assignment reconstruction and workflow forests.

For each task node v and eligible provider i , define $D_v(i)$ as the minimum contribution to the allocation objective from

v and all its descendant tasks, when v is assigned to i . It includes their output generation and input processing on edges between those tasks. An edge from v 's parent is accounted for when combining v with that parent; fresh user-context processing is included at the root r . Thus the complete workflow optimum is $\min_{i \in \mathcal{M}_r} D_r(i)$. We compute these values from leaves to root by dynamic programming.

Write $s_i = s_i(b_i)$ and let $\text{ch}(v)$ denote v 's children. The local generation contribution is $g_v(i) = s_i L_v^{\text{out}}$ for $v \neq r$; at the root, $g_r(i) = s_i [L_r^{\text{out}} + \rho_i L_{v_{\text{user}}, r}^{\text{trans}}]$ also includes fresh source input. The edge contribution $\eta_{v,u}(i, j) = s_j \rho_j L_{v,u}^{\text{trans}} [\kappa_{v,u} + (1 - \kappa_{v,u}) \mathbf{1}\{i \neq j\}]$ accounts for processing the parent-child context. The recurrence is

$$D_v(i) = g_v(i) + \sum_{u \in \text{ch}(v)} \min_{j \in \mathcal{M}_u} \{D_u(j) + \eta_{v,u}(i, j)\}. \quad (54)$$

Correctness. With v 's provider fixed, its child branches can be optimized independently. Each minimizing term combines the child's best assignment with its connecting edge, charging that edge once. At a leaf, $D_v(i) = g_v(i)$; induction up the tree establishes every entry and hence the optimal root value. Computing all values takes $O(|\mathcal{V}_T|N^2)$ operations. Backtracking recovers an optimal assignment with the public tie order.

Root choice and reconstruction. Equivalently, the context-processing edge factor can be written as

$$\eta_{v,u}(i, j) = s_j(b_j) \rho_j L_{v,u}^{\text{trans}} [\kappa_{v,u} + (1 - \kappa_{v,u}) \mathbf{1}\{i \neq j\}]. \quad (55)$$

The root entry already includes fresh user-context processing, so the complete objective is

$$\min_{i \in \mathcal{M}_r} D_r(i). \quad (56)$$

A workflow forest with several task roots is handled by applying Eq. (56) independently to each root and summing the resulting objectives.

For reconstruction, each dynamic-programming entry stores its minimizing child-provider choices. Backtracking from an optimal root choice determines the assignment of every task. Among equal-objective subtree assignments, the public node and provider ordering selects the lexicographically smallest assignment; this ordering is fixed before reports are submitted, just as in the main mechanism.

Evaluating the objective values requires

$$O\left(\sum_{v \in \mathcal{V}_T} |\mathcal{M}_v| + \sum_{(v,u) \in \mathcal{E}_{\text{int}}} |\mathcal{M}_v| |\mathcal{M}_u|\right) \quad (57)$$

operations, bounded by $O(|\mathcal{V}_T|N^2)$ for a task tree. The first term accounts for node values, including single-node trees; the second enumerates eligible parent-child provider pairs.

G Exact Tree-Decomposition Dynamic Programming

This section gives an exact junction-tree dynamic program for finite DAGs whose underlying undirected task graph has bounded treewidth. A directed DAG may induce cycles in this underlying undirected graph, for example through multi-parent fan-in. This is distinct from a directed control-flow cycle: workflows with directed loops must first be unrolled into finite DAGs.

Pairwise factorization. Let

$$\mathcal{H} = (\mathcal{V}_T, \mathcal{E}_H)$$

be the underlying undirected task graph, where $\{u, v\} \in \mathcal{E}_H$ whenever either (u, v) or (v, u) is an internal dependency edge. Each task v has assignment domain $\mathcal{M}_v$.

The source-edge and generation contributions can be combined into the unary factor

$$\psi_v(i) = s_i(b_i) \left[L_v^{\text{out}} + \rho_i \sum_{(v_{\text{user}}, v) \in \mathcal{E}_{\text{src}}} L_{v_{\text{user}}, v}^{\text{trans}} \right], \quad i \in \mathcal{M}_v. \quad (58)$$

For every internal directed edge (u, v) , define the pairwise factor

$$\psi_{u \rightarrow v}(i, j) = s_j(b_j) \rho_j L_{u,v}^{\text{trans}} [\kappa_{u,v} + (1 - \kappa_{u,v}) \mathbf{1}\{i \neq j\}], \quad (59)$$

where $i \in \mathcal{M}_u$ and $j \in \mathcal{M}_v$. The complete score objective is therefore

$$\sum_{v \in \mathcal{V}_T} \psi_v(A(v)) + \sum_{(u,v) \in \mathcal{E}_{\text{int}}} \psi_{u \rightarrow v}(A(u), A(v)). \quad (60)$$

Equation (60) is exactly the objective in Eq. (5), expressed as unary and pairwise factors.

Tree decomposition. A tree decomposition of $\mathcal{H}$ is a tree $\mathcal{T} = (\mathcal{I}, \mathcal{F})$ together with bags $\{B_t \subseteq \mathcal{V}_T : t \in \mathcal{I}\}$ satisfying:

1. for every vertex v , the bags containing v form a connected subtree of $\mathcal{T}$;
2. for every edge $\{u, v\} \in \mathcal{E}_H$, at least one bag contains both u and v .

The width is

$$w = \max_{t \in \mathcal{I}} |B_t| - 1.$$

We root $\mathcal{T}$ at an arbitrary bag t_0 . Each unary and pairwise factor in Eq. (60) is assigned to exactly one bag containing all variables in its scope. For example, one may assign each factor to the highest such bag under the rooted decomposition.

Let $\mathcal{Q}_t$ be the factors assigned to bag B_t . For an assignment

$$\mathbf{a}_t \in \prod_{v \in B_t} \mathcal{M}_v,$$

define the local bag cost

$$\Psi_t(\mathbf{a}_t) = \sum_{\psi \in Q_t} \psi(\mathbf{a}_t|_{\text{scope}(\psi)}). \quad (61)$$

Because every factor is assigned to exactly one bag, no subtraction or double-counting correction is required.

For every non-root bag t , let $p(t)$ be its parent and define the separator

$$S_t = B_t \cap B_{p(t)}.$$

The message from bag t to its parent is

$$m_{t \rightarrow p(t)}(\mathbf{a}_{S_t}) = \min_{\mathbf{a}_{B_t \setminus S_t}} \left[\Psi_t(\mathbf{a}_t) + \sum_{c \in \text{ch}(t)} m_{c \rightarrow t}(\mathbf{a}_{B_c \cap B_t}) \right]. \quad (62)$$

At the root bag t_0 , the optimal objective is

$$\min_{\mathbf{a}_{B_{t_0}}} \left[\Psi_{t_0}(\mathbf{a}_{t_0}) + \sum_{c \in \text{ch}(t_0)} m_{c \rightarrow t_0}(\mathbf{a}_{B_c \cap B_{t_0}}) \right]. \quad (63)$$

Backtracking through the minimizing assignments recovers a globally optimal workflow allocation.

Let

$$M_{\max} = \max_{v \in \mathcal{V}_T} |\mathcal{M}_v|.$$

For a supplied tree decomposition of width w , the running time is bounded by

$$O(|\mathcal{I}| M_{\max}^{w+1}) \quad (64)$$

up to polynomial factors for evaluating local potentials and messages [5]. For fixed treewidth, the dynamic program runs in polynomial time; it is fixed-parameter tractable when parameterized jointly by w and $M_{\max}$.

With an available low-width decomposition, this junction-tree formulation provides an exact theoretical allocation algorithm. Its complexity is parameterized by the supplied decomposition width. The reported experimental pipeline uses the MILP in Section 4.5.

H Synthetic Market and Workflow Configurations

All four workflow families are hand-constructed finite DAGs. Nominal token values below are reported in thousands of tokens for readability; the solver converts them to millions of tokens before applying per-million-token costs. Unless otherwise stated, nominal values are sampled using Eq. (67) below.

H.1 Cost and Token Sampling

We simulate nine providers in three capability tiers, with synthetic base-cost anchors following a 1:5:25 progression. This progression provides a controlled separation between

provider tiers. In the synthetic S1–S4 reference configuration,

$$c_i = C_i^{\text{base}} S_i, \quad S_i \sim \text{Uniform}(0.1, 1.0), \quad (65)$$

so provider i has support

$$[a_i, d_i] = [0.1 C_i^{\text{base}}, C_i^{\text{base}}], \quad \phi_i(c) = 2c - a_i. \quad (66)$$

We set $\rho_i = 1/3$ for every provider, use $\kappa_{u,v} = 0.1$ on cache-eligible edges, and set $\kappa_{u,v} = 1$ on non-cacheable edges. The base-cost anchors remain fixed throughout this configuration.

For each nominal token quantity $\bar{L}$, we sample

$$L \sim \text{LogNormal}\left(\log \bar{L} - \frac{\sigma^2}{2}, \sigma^2\right), \quad \sigma = 0.3. \quad (67)$$

The S2 source context is sampled separately from the configured truncated-Pareto distribution on 5k–100k tokens.

H.2 Provider Tiers

A provider in capability tier t can execute tasks of difficulty at most t . In the workflow diagrams, D1, D2, and D3 denote public task-difficulty tiers 1, 2, and 3; “Diff-1” likewise means difficulty tier 1.

Tier	Base-cost anchors	Providers
Tier 1	\$0.10, \$0.20, \$0.30	3
Tier 2	\$0.50, \$1.00, \$1.50	3
Tier 3	\$2.50, \$5.00, \$7.50	3

Table 4: Synthetic base-cost anchors in dollars per million standardized output-workload units.

I Additional Experimental Results

I.1 Synthetic S1–S4 Uniform-Prior Reference

The synthetic reference evaluates the four richer S1–S4 DAG families under provider-specific uniform priors. STAR lowers sample-mean payment relative to Raw-DSIC by 1.44%, 0.36%, 0.91%, and 0.92%, respectively; the paired interval for S2 includes zero. This is a useful boundary case: when the raw and virtual score trade-offs are closely aligned, STAR and Raw-DSIC often select similar allocations, so the additional payment benefit of virtual-cost optimization is modest.

I.2 Heterogeneous-Prior Stress Configuration

Table 7 specifies the fixed provider cost ranges and prior shapes. Experimental provider identifiers are 0–4; capability tier 2 can also execute tier-1 tasks. The structural cells are specified below.

The uniform, mild, and strong profiles use the same frozen structural cells. Every pre-specified cell is retained, including weak or nonsignificant effects. The documented provider-to-prior assignment is fixed across these cells, making the comparison reproducible for each specified market profile.

Setting	Role	Count	Difficulty	Nominal output (k)
S1	Planner	5	1	300 each
	Writers	15 (3 per stage)	2	40 each
	Aggregator	5	1	50 each
	Final Editor	1	1	20
S2	Scanner / Analyzer / Planner	1 each	1 / 2 / 3	3 / 5 / 8
	SpecGenerator	1	1	350
	Modules	10	$1 \times 3, 7 \times 2, 2 \times 1$	25 each
	File tasks	30	$10 \times 3, 20 \times 1$	15 each
	Merger / Tester	1 each	1 / 2	15 / 8
S3	Architect / SpecWriter	1 each	2 / 1	10 / 300
	Frontend / Backend / Interactive Lead	1 each	2 / 2 / 2	50 / 40 / 30
	Frontend developers	5	$3 \times 1, 2 \times 2$	15 each
	Backend developers	4	$1 \times 1, 3 \times 2$	20 each
	Interactive developers	3	$1 \times 2, 2 \times 3$	15 each
S4	Expert Assessment / Case Retriever	1 each	3 / 1	5 / 400
	Analysts	5	$3 \times 1, 2 \times 2$	40, 35, 30, 20, 18
	Aggregator	1	1	80
	Strategists	3	2	20 each
	Expert Synthesis	1	3	10

Table 5: Nominal synthetic workflow configurations. S1 repeats the Planner–Writers–Aggregator stage five times. In S2, the Scanner root context is sampled from the configured truncated-Pareto range, and the Merger receives all 30 file outputs, giving approximately 450k nominal fan-in tokens. The S3 Architect root input has 5k nominal tokens. The S4 Expert Assessment root input has 50k nominal tokens.

Workflow	Payment reduction	Paired 95% CI
S1	1.44%	[0.14%, 2.72%]
S2	0.36%	[-0.39%, 1.13%]
S3	0.91%	[0.07%, 1.79%]
S4	0.92%	[0.11%, 1.71%]

Table 6: Synthetic S1–S4 uniform-prior boundary reference. Intervals are paired-bootstrap 95% confidence intervals for STAR relative to graph-aware Raw-DSIC.

I.3 Complete Heterogeneous-Prior Results

For the representative degree-100 high-payload cell with $\rho = 1/3$, the two distinct effects are visible in the same comparison:

The graph-aware execution reduction remains near 90% across the three prior profiles, whereas the payment advantage varies strongly with prior shape. Thus, graph coupling and prior heterogeneity affect different comparisons.

I.4 Complete Prior-Shape Continuum

The normalized curves coincide across degrees 10, 20, and 100 because the frozen construction replicates identical consumer branches while preserving the set of competing provider pairs. Degree increases absolute payment savings. The mild profile serves as a separate off-path reference: its least-squares log- k projection is $\lambda = 0.338$, with a non-negligible residual.

I.5 Structural Negative Controls

Table 3 in Section 5.6 reports all three structural controls. The no-cache control makes graph-aware and myopic allocations agree row by row. The payload controls therefore support the mechanism in Proposition 6: fan-out creates a large topology advantage only when each repeated edge carries substantial avoidable fresh-input workload.

I.6 Support-Width Heterogeneity as a Boundary Case

The region-inspired experiment varies synthetic support-width profiles while holding normalized provider means fixed, allowing aggregate dispersion to change with the treatment. In the tested degree-100 and S4 cells, the additional treatment-control payment reduction is modest and its 95% interval includes zero. These paired estimates characterize the payment effect and uncertainty for the documented synthetic profiles, complementing the prior-shape comparisons above.

I.7 Prior Misspecification and Runtime Details

Prior-estimation-error magnitude grid (E2). The experiment uses the original five provider cost ranges, eligibility rules, degree-10 fan-out graph, $\rho = 1/3$, and cache semantics. The uniform profile has $k_i = 1$ for all providers; the heterogeneous power-prior profile uses $k = (0.5, 2, 0.2, 5, 1)$, the same provider-specific shapes used in Table 2. Appendix I.2 gives their cost ranges. For each profile, we evaluate eight scoring conditions: correct-prior STAR, six signed

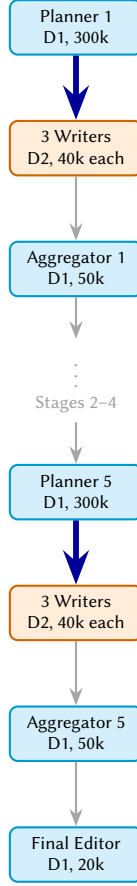


Figure 3: S1: Repeated Broadcasters. Five repeated stages each contain one Diff-1 planner broadcasting a 300k-token plan to three Diff-2 writers, followed by a Diff-1 aggregator. The grouped writer boxes represent three parallel nodes per stage; stages 2–4 are omitted visually but are instantiated in the generator.

Provider ID	Capability tier	Unit-cost range (\$/million work units)		Uniform k_i	Mild k_i	Strong k_i
		a_i	d_i			
0	1	0.05	0.50	1.0	0.5	0.5
1	1	0.10	0.80	1.0	2.0	2.0
2	2	0.25	1.40	1.0	0.5	0.2
3	2	0.35	1.10	1.0	1.0	5.0
4	2	0.45	1.25	1.0	2.0	1.0

Table 7: Fixed provider cost ranges and distribution shapes for the structural-prior stress suite. Work units are the standardized workloads in Eq. (3). The cost ranges stay fixed across profiles; the listed shape parameters determine the virtual costs $\phi_i(c) = (1 + 1/k_i)c - a_i/k_i$.

shape perturbations $\widehat{k}_i = k_i e^{\delta r_i}$ with $r = (1, -1, 0, 1, -1)$ and $\delta \in \{-0.2, -0.1, -0.05, +0.05, +0.1, +0.2\}$, and Raw-DSIC. These controlled parameter errors leave each provider’s true cost and truthful report unchanged.

Before execution, we fix 1,000 new draw identifiers (30,000–30,999) per profile and retain every draw and signed perturbation. Each draw’s latent quantiles are shared across the two true profiles, and within a profile all scoring conditions share the identical true cost vector. The shapes

obey $\widehat{k}_i/k_i \in [e^{-m}, e^m]$ at magnitude $m = |\delta|$. For $m = 0.05, 0.1, 0.2$, the maximum decreases are 4.88%, 9.52%, and 18.13%, and the maximum increases are 5.13%, 10.52%, and 22.14%, respectively. The fixed direction permits a matched comparison of both signs as the error magnitude grows.

Table 16 gives all condition-level payment decompositions and changes from correct-prior routing. All 95% intervals use 2,000 resamples of the 1,000 paired draw identifiers, with scoring conditions resampled together. The new cohort is

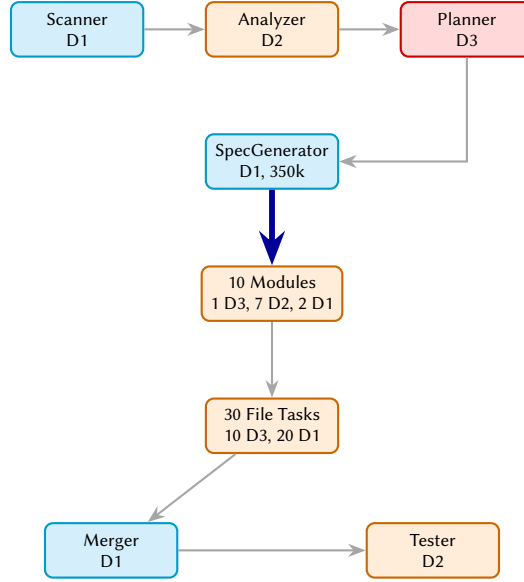


Figure 4: S2: Nested Fan-Out. A Diff-1 SpecGenerator broadcasts a 350k-token specification to ten modules, which decompose into thirty file tasks. All thirty file outputs feed the Merger, producing approximately 450k nominal fan-in tokens.

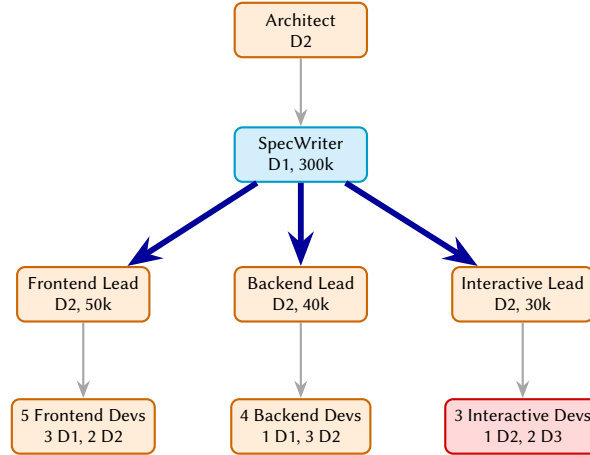


Figure 5: S3: Two-Level Team Fan-Out. A Diff-1 SpecWriter broadcasts a 300k-token specification to three Diff-2 team leads, which route work to twelve developers with heterogeneous capability requirements.

analyzed separately from earlier runs; its means are not pooled with earlier draws.

Optimization runtime (E1). These central processing unit (CPU) measurements time the task-assignment and payment optimizations required to run STAR on a supplied workflow and provider reports. Synthetic graphs provide controlled inputs to those same computations. The reported mechanism time includes constructing and solving the allocation problems, report-independent tie-breaking, and every active provider’s payment computation; graph generation is outside this timer.

On one Apple M2 Pro processor, all 36 chain/sparse-DAG instances complete within the time limits. We test 8, 32, and 64 task nodes, with 3 or 5 providers and three seeds. The largest complete-mechanism time is 0.767 seconds. Peak

sampled memory is 268.9 mebibytes (MiB), measured as the summed resident set size (RSS, physical memory occupied) of the Python worker and all CBC solver processes it launches, including their subprocesses. Memory is sampled every 20 milliseconds.

CBC uses one thread, zero requested relative and absolute optimality gap, and a 30-second limit for each allocation optimization. Each complete mechanism has a 180-second limit. Table 17 summarizes the timings by workflow size, and Table 18 gives the graph-family and provider-count breakdown.

1.8 Experiment Coverage and Validation

The evaluation includes every pre-specified synthetic structural-stress and prior-continuum cell. E1 completes

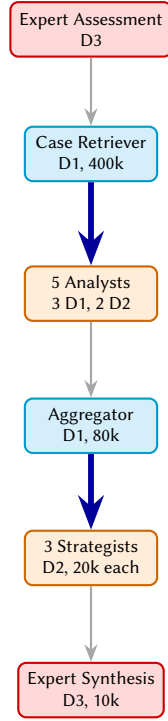


Figure 6: S4: Hourglass. A Diff-1 Case Retriever broadcasts a 400k-token payload to five analysts. Their outputs contract at a Diff-1 aggregator, expand to three Diff-2 strategists, and reconverge at the final Diff-3 synthesis node.

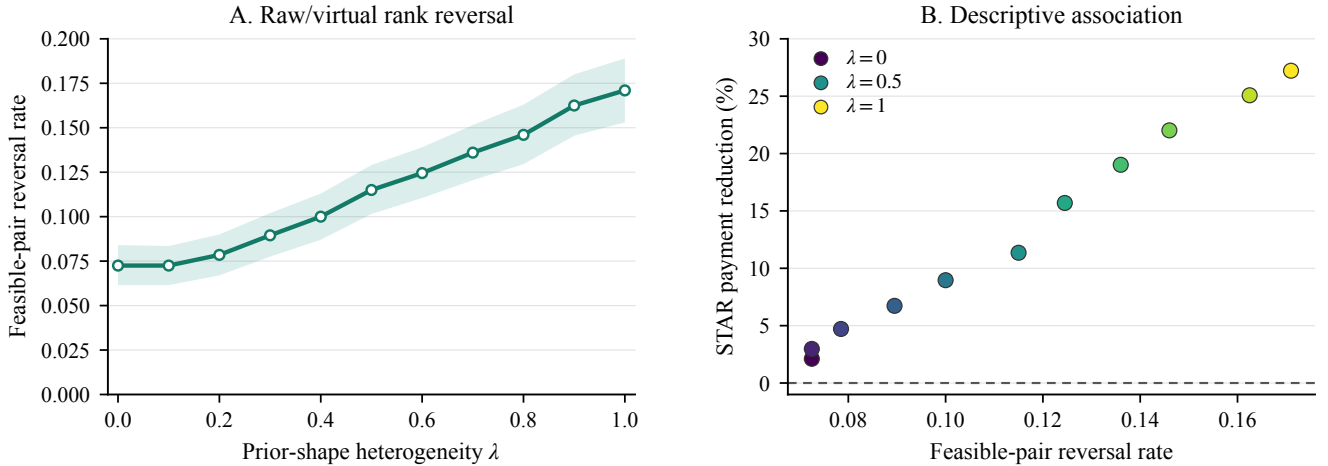


Figure 7: Prior-continuum diagnostics for degree 100. The left panel reports the feasible-provider raw/virtual rank-reversal rate across λ ; the right panel summarizes its descriptive association with STAR’s payment reduction along the fixed synthetic path.

36/36 instances, and E2 includes all 16,000 pre-specified condition-level results across both true profiles and all eight scoring conditions per profile. Every allocation optimization reports status Optimal with zero reported gap; the E2 minimum truthful provider utility is zero and payment-decomposition errors are at numerical precision. In every checked instance, when providers report their actual unit costs, Raw-DSIC assigns each task to exactly the same provider as the execution-cost-minimizing oracle.

Cell	Topology	Degree	ρ	Paired trials
C01	Fanout	1	1/3	50
C02	Fanout	1	1.00	50
C03	Fanout	10	1/3	50
C04	Fanout	10	1.00	50
C05	Fanout	30	1/3	50
C06	Fanout	30	1.00	50
C07	Fanout	100	1/3	50
C08	Fanout	100	1.00	50
C09	Fanin	1	1/3	50
C10	Fanin	1	1.00	50
C11	Fanin	10	1/3	50
C12	Fanin	10	1.00	50
C13	Fanin	30	1/3	50
C14	Fanin	30	1.00	50
C15	Fanin	100	1/3	50
C16	Fanin	100	1.00	50
C17	Hourglass	1	1/3	50
C18	Hourglass	1	1.00	50
C19	Hourglass	10	1/3	50
C20	Hourglass	10	1.00	50
C21	Hourglass	30	1/3	50
C22	Hourglass	30	1.00	50
C23	Hourglass	100	1/3	50
C24	Hourglass	100	1.00	50

Table 8: The 24 frozen structural cells used separately under each of the uniform, mild, and strong prior profiles. The fixed settings are $\kappa = 0.0$, a transferred payload of 1,000,000 tokens, a generated hub payload of 1,000,000 tokens, and child outputs of 20,000 tokens.

Topology	Degree	ρ	STAR–Raw payment	Paired 95% CI	Raw–Myopic execution	Paired 95% CI
Fanout	1	1/3	3.34%	[0.52%, 6.07%]	1.77%	[0.57%, 3.31%]
Fanout	1	1.00	1.49%	[-0.36%, 3.41%]	24.15%	[20.89%, 27.28%]
Fanout	10	1/3	1.49%	[-0.44%, 3.49%]	67.79%	[66.31%, 68.86%]
Fanout	10	1.00	1.49%	[-0.39%, 3.48%]	87.84%	[87.16%, 88.34%]
Fanout	30	1/3	1.49%	[-0.46%, 3.48%]	84.60%	[83.79%, 85.19%]
Fanout	30	1.00	1.49%	[-0.37%, 3.41%]	94.47%	[94.12%, 94.71%]
Fanout	100	1/3	1.49%	[-0.39%, 3.58%]	91.11%	[90.63%, 91.48%]
Fanout	100	1.00	1.49%	[-0.38%, 3.41%]	96.88%	[96.68%, 97.02%]
Fanin	1	1/3	1.49%	[-0.47%, 3.50%]	9.07%	[7.47%, 10.74%]
Fanin	1	1.00	1.49%	[-0.53%, 3.50%]	24.15%	[20.77%, 27.45%]
Fanin	10	1/3	1.49%	[-0.49%, 3.41%]	10.09%	[8.56%, 11.62%]
Fanin	10	1.00	1.49%	[-0.52%, 3.42%]	25.07%	[21.80%, 28.15%]
Fanin	30	1/3	1.49%	[-0.37%, 3.53%]	10.17%	[8.68%, 11.76%]
Fanin	30	1.00	1.49%	[-0.48%, 3.47%]	25.14%	[22.02%, 28.34%]
Fanin	100	1/3	1.49%	[-0.49%, 3.43%]	10.19%	[8.68%, 11.74%]
Fanin	100	1.00	1.49%	[-0.55%, 3.52%]	25.17%	[21.90%, 28.29%]
Hourglass	1	1/3	2.31%	[0.58%, 4.11%]	6.28%	[5.01%, 7.65%]
Hourglass	1	1.00	1.49%	[-0.54%, 3.48%]	24.51%	[21.19%, 27.61%]
Hourglass	10	1/3	1.49%	[-0.43%, 3.47%]	24.98%	[23.36%, 26.52%]
Hourglass	10	1.00	1.49%	[-0.40%, 3.54%]	52.57%	[50.53%, 54.19%]
Hourglass	30	1/3	1.49%	[-0.42%, 3.42%]	28.30%	[26.73%, 29.65%]
Hourglass	30	1.00	1.49%	[-0.50%, 3.50%]	55.01%	[53.12%, 56.48%]
Hourglass	100	1/3	1.49%	[-0.40%, 3.47%]	29.49%	[28.00%, 30.82%]
Hourglass	100	1.00	1.49%	[-0.46%, 3.44%]	55.88%	[53.94%, 57.45%]

Table 9: Complete frozen structural-grid results for the uniform prior. Payment columns compare STAR with graph-aware Raw-DSIC; execution columns compare graph-aware Raw with Myopic. All 24 pre-specified cells are retained, and intervals are paired-bootstrap 95% confidence intervals.

Topology	Degree	ρ	STAR–Raw payment	Paired 95% CI	Raw–Myopic execution	Paired 95% CI
Fanout	1	1/3	10.30%	[3.26%, 16.70%]	2.27%	[0.71%, 4.17%]
Fanout	1	1.00	10.23%	[3.42%, 16.73%]	22.65%	[18.73%, 26.62%]
Fanout	10	1/3	10.23%	[3.26%, 16.47%]	67.75%	[66.33%, 68.85%]
Fanout	10	1.00	10.23%	[2.85%, 16.82%]	87.92%	[87.29%, 88.35%]
Fanout	30	1/3	10.23%	[3.45%, 16.98%]	84.68%	[83.88%, 85.21%]
Fanout	30	1.00	10.23%	[3.25%, 16.81%]	94.52%	[94.22%, 94.72%]
Fanout	100	1/3	10.23%	[3.44%, 16.73%]	91.19%	[90.67%, 91.50%]
Fanout	100	1.00	10.23%	[3.50%, 16.60%]	96.91%	[96.72%, 97.02%]
Fanin	1	1/3	10.23%	[3.53%, 16.63%]	8.31%	[6.44%, 10.26%]
Fanin	1	1.00	10.23%	[3.21%, 16.91%]	22.65%	[18.64%, 26.61%]
Fanin	10	1/3	10.23%	[3.27%, 16.88%]	9.38%	[7.61%, 11.21%]
Fanin	10	1.00	10.23%	[3.54%, 16.95%]	23.61%	[19.60%, 27.38%]
Fanin	30	1/3	10.23%	[3.68%, 16.96%]	9.47%	[7.72%, 11.24%]
Fanin	30	1.00	10.23%	[3.28%, 16.66%]	23.69%	[19.99%, 27.49%]
Fanin	100	1/3	10.23%	[3.73%, 17.08%]	9.49%	[7.73%, 11.35%]
Fanin	100	1.00	10.23%	[3.17%, 16.94%]	23.71%	[19.82%, 27.42%]
Hourglass	1	1/3	10.26%	[4.23%, 16.00%]	6.06%	[4.49%, 7.70%]
Hourglass	1	1.00	10.23%	[3.21%, 16.57%]	22.99%	[19.09%, 26.89%]
Hourglass	10	1/3	10.23%	[3.40%, 16.92%]	24.51%	[22.74%, 26.19%]
Hourglass	10	1.00	10.23%	[3.45%, 16.79%]	52.16%	[50.08%, 54.01%]
Hourglass	30	1/3	10.23%	[3.23%, 16.66%]	27.96%	[26.39%, 29.50%]
Hourglass	30	1.00	10.23%	[3.27%, 16.39%]	54.66%	[52.74%, 56.47%]
Hourglass	100	1/3	10.23%	[3.49%, 16.92%]	29.20%	[27.70%, 30.64%]
Hourglass	100	1.00	10.23%	[3.50%, 16.69%]	55.55%	[53.69%, 57.16%]

Table 10: Complete frozen structural-grid results for the heterogeneous mild prior. Payment columns compare STAR with graph-aware Raw-DSIC; execution columns compare graph-aware Raw with Myopic. All 24 pre-specified cells are retained, and intervals are paired-bootstrap 95% confidence intervals.

Topology	Degree	ρ	STAR–Raw payment	Paired 95% CI	Raw–Myopic execution	Paired 95% CI
Fanout	1	1/3	21.99%	[14.04%, 29.82%]	2.52%	[0.80%, 4.58%]
Fanout	1	1.00	31.48%	[21.92%, 40.29%]	21.82%	[17.46%, 25.88%]
Fanout	10	1/3	31.48%	[21.96%, 40.40%]	65.32%	[62.29%, 67.52%]
Fanout	10	1.00	31.48%	[21.78%, 40.66%]	86.63%	[84.98%, 87.72%]
Fanout	30	1/3	31.48%	[21.78%, 40.41%]	83.13%	[81.04%, 84.43%]
Fanout	30	1.00	31.48%	[21.60%, 40.12%]	93.88%	[93.05%, 94.40%]
Fanout	100	1/3	31.48%	[22.56%, 40.73%]	90.20%	[88.91%, 91.03%]
Fanout	100	1.00	31.48%	[22.30%, 40.32%]	96.53%	[96.05%, 96.84%]
Fanin	1	1/3	31.48%	[22.27%, 40.39%]	8.02%	[6.05%, 9.96%]
Fanin	1	1.00	31.48%	[21.98%, 40.82%]	21.82%	[17.44%, 26.01%]
Fanin	10	1/3	31.48%	[21.82%, 40.50%]	8.96%	[7.09%, 11.02%]
Fanin	10	1.00	31.48%	[21.65%, 40.27%]	22.70%	[18.51%, 26.69%]
Fanin	30	1/3	31.48%	[21.87%, 40.61%]	9.03%	[7.13%, 10.98%]
Fanin	30	1.00	31.48%	[21.48%, 40.63%]	22.76%	[18.55%, 26.72%]
Fanin	100	1/3	31.48%	[21.84%, 40.35%]	9.06%	[7.19%, 11.00%]
Fanin	100	1.00	31.48%	[22.18%, 40.90%]	22.78%	[18.55%, 26.99%]
Hourglass	1	1/3	27.55%	[18.85%, 35.28%]	5.87%	[4.17%, 7.77%]
Hourglass	1	1.00	31.48%	[21.52%, 41.01%]	22.15%	[17.71%, 26.42%]
Hourglass	10	1/3	31.48%	[22.28%, 40.38%]	22.85%	[20.33%, 25.12%]
Hourglass	10	1.00	31.48%	[22.19%, 40.72%]	49.70%	[45.98%, 52.66%]
Hourglass	30	1/3	31.48%	[21.86%, 40.62%]	26.03%	[23.33%, 28.28%]
Hourglass	30	1.00	31.48%	[21.73%, 40.69%]	52.17%	[48.63%, 54.82%]
Hourglass	100	1/3	31.48%	[21.53%, 40.47%]	27.17%	[24.50%, 29.44%]
Hourglass	100	1.00	31.48%	[21.85%, 40.53%]	53.05%	[49.61%, 55.75%]

Table 11: Complete frozen structural-grid results for the heterogeneous strong prior. Payment columns compare STAR with graph-aware Raw-DSIC; execution columns compare graph-aware Raw with Myopic. All 24 pre-specified cells are retained, and intervals are paired-bootstrap 95% confidence intervals.

Prior profile	Raw graph-aware vs. Myopic execution reduction	STAR vs. Raw-DSIC payment reduction
Uniform	91.11% [90.63, 91.48]	1.49% [−0.39, 3.58]
Heterogeneous mild	91.19% [90.67, 91.50]	10.23% [3.44, 16.73]
Heterogeneous strong	90.20% [88.91, 91.03]	31.48% [22.56, 40.73]

Table 12: Representative degree-100 high-payload cell at $\rho = 1/3$. The middle column isolates topology awareness, whereas the right column compares two graph-aware DSIC mechanisms and isolates the additional effect of virtual-cost scoring.

λ	Payment reduction	Paired 95% CI	Pair-reversal rate	Allocation-disagreement rate
0.0	2.11%	[1.11%, 3.10%]	0.072	0.060
0.1	2.97%	[1.51%, 4.44%]	0.072	0.090
0.2	4.71%	[2.58%, 6.78%]	0.079	0.100
0.3	6.73%	[3.91%, 9.52%]	0.089	0.140
0.4	8.96%	[5.79%, 12.10%]	0.100	0.210
0.5	11.35%	[7.86%, 14.93%]	0.115	0.250
0.6	15.69%	[11.87%, 19.54%]	0.124	0.235
0.7	19.02%	[14.81%, 23.22%]	0.136	0.220
0.8	22.02%	[17.44%, 26.35%]	0.146	0.240
0.9	25.08%	[20.28%, 29.65%]	0.163	0.255
1.0	27.22%	[22.41%, 32.02%]	0.171	0.285

Table 13: Complete degree-100 prior-shape continuum. Supports and high-payload graph semantics are fixed; the mild prior serves as a separate off-path reference. Rank reversal and allocation disagreement are reported as rates, and payment intervals are paired-bootstrap 95% confidence intervals.

Degree	Absolute saving (\$)	Reduction
10	0.276	27.22%
20	0.322	27.22%
100	0.687	27.22%

Table 14: Degree scaling at $\lambda = 1$. Absolute savings grow with replicated workload, while normalized reductions coincide under the frozen replication-scaled construction.

Structural cell	η	Treatment [95% CI]	Control [95% CI]	Treatment-control [95% CI]	CI > 0
Degree-100 fan-out	0.25	0.000% [0.000%, 0.000%]	0.000% [0.000%, 0.000%]	0.000% [0.000%, 0.000%]	No
Degree-100 fan-out	0.50	0.096% [-1.718%, 1.804%]	-0.323% [-2.167%, 1.088%]	0.419% [-0.972%, 1.723%]	No
Degree-100 fan-out	1.00	2.538% [-0.658%, 5.790%]	1.658% [-0.545%, 4.082%]	0.879% [-2.050%, 3.744%]	No
S4 cached input	0.25	0.005% [0.000%, 0.012%]	0.000% [0.000%, 0.000%]	0.005% [0.000%, 0.012%]	No
S4 cached input	0.50	0.256% [-0.919%, 1.429%]	-0.316% [-1.393%, 0.595%]	0.572% [-0.387%, 1.477%]	No
S4 cached input	1.00	3.385% [0.973%, 5.752%]	1.748% [-0.232%, 3.741%]	1.637% [-0.271%, 3.602%]	No

Table 15: Region-inspired support-width contrasts. Treatment and control preserve provider means while varying support widths and aggregate dispersion. The parameter η scales each cost range’s half-width around its fixed mean: treatment half-widths are 20%, 50%, or 90% of the mean, multiplied by η ; control half-widths are uniformly 50% $\times \eta$. The S4 cell uses cached-input fraction $\kappa = 0.1$. Each paired-bootstrap interval resamples $n = 50$ seed blocks, retaining all six profile assignments together within each block. The pre-specified materiality assessment uses a 5-percentage-point threshold; the final column checks whether the treatment-control interval lies strictly above zero.

True prior	Scoring condition	P	C_{exec}	R	$P - P_{\text{correct}}$ (95% CI)	Allocation disagreement
Uniform	Correct prior	0.939	0.686	0.254	+0.0000 [+0.0000,+0.0000]	0.0%
Uniform	$\delta = -0.2$	0.944	0.692	0.252	+0.0043 [-0.0012,+0.0096]	7.0%
Uniform	$\delta = -0.1$	0.940	0.688	0.252	+0.0008 [-0.0019,+0.0035]	3.5%
Uniform	$\delta = -0.05$	0.939	0.686	0.253	-0.0005 [-0.0019,+0.0008]	1.0%
Uniform	$\delta = +0.05$	0.940	0.685	0.255	+0.0007 [-0.0006,+0.0020]	1.9%
Uniform	$\delta = +0.1$	0.942	0.685	0.256	+0.0023 [-0.0002,+0.0047]	3.4%
Uniform	$\delta = +0.2$	0.945	0.686	0.260	+0.0056 [+0.0013,+0.0102]	6.8%
Uniform	Raw-DSIC	0.947	0.682	0.265	+0.0075 [+0.0029,+0.0119]	8.9%
Heterogeneous	Correct prior	0.694	0.564	0.130	+0.0000 [+0.0000,+0.0000]	0.0%
Heterogeneous	$\delta = -0.2$	0.693	0.557	0.136	-0.0012 [-0.0052,+0.0025]	3.1%
Heterogeneous	$\delta = -0.1$	0.693	0.560	0.133	-0.0010 [-0.0042,+0.0018]	1.7%
Heterogeneous	$\delta = -0.05$	0.694	0.563	0.132	+0.0003 [-0.0013,+0.0018]	0.7%
Heterogeneous	$\delta = +0.05$	0.695	0.565	0.129	+0.0006 [-0.0009,+0.0027]	0.5%
Heterogeneous	$\delta = +0.1$	0.695	0.567	0.128	+0.0013 [-0.0010,+0.0037]	1.1%
Heterogeneous	$\delta = +0.2$	0.693	0.567	0.126	-0.0008 [-0.0045,+0.0028]	2.0%
Heterogeneous	Raw-DSIC	0.992	0.476	0.516	+0.2980 [+0.2754,+0.3195]	26.6%

Table 16: Complete decomposition for the prior-estimation-error magnitude grid. All rows use the same 1,000 paired cost draws per true profile as Table 2. Uniform denotes $k_i = 1$ for all providers; heterogeneous denotes the provider-specific vector $k = (0.5, 2, 0.2, 5, 1)$ in Table 2. Only assumed shapes vary, with $\delta = \pm 0.05, \pm 0.1, \pm 0.2$. Payment is $P = C_{\text{exec}} + R$ before rounding. Payment changes and assignment disagreement are relative to correct-prior STAR; intervals use 2,000 paired bootstrap resamples. Disagreement is the fraction of draws with at least one task assigned differently.

$ \mathcal{V}_T $	Completed instances	Task-assignment time (seconds)	All-provider payment time (seconds)	Maximum total time (seconds)
8	12/12	0.069	0.118	0.285
32	12/12	0.094	0.219	0.512
64	12/12	0.134	0.361	0.767

Table 17: Full-mechanism CPU scaling. Times aggregate the chain and original sparse-DAG generators, provider counts $N \in \{3, 5\}$, and three seeds. Allocation and all-provider payment columns are medians; max total retains the slowest completed instance.

family	$ \mathcal{V}_T $	N	solved	allocation (s)	payments (s)	peak RSS (MiB)
chain	8	3	3/3	0.124	0.120	183.3
chain	8	5	3/3	0.067	0.115	182.0
chain	32	3	3/3	0.078	0.214	183.7
chain	32	5	3/3	0.104	0.262	187.2
chain	64	3	3/3	0.113	0.278	187.8
chain	64	5	3/3	0.194	0.421	268.9
sparse DAG	8	3	3/3	0.068	0.101	182.6
sparse DAG	8	5	3/3	0.070	0.119	181.6
sparse DAG	32	3	3/3	0.090	0.175	187.0
sparse DAG	32	5	3/3	0.122	0.245	189.1
sparse DAG	64	3	3/3	0.134	0.296	193.1
sparse DAG	64	5	3/3	0.201	0.513	204.3

Table 18: Optimization runtime by graph family, task count, and provider count. Each row contains three fixed seeds. N is the number of available providers; task eligibility follows the capability requirements. Memory is the maximum sampled sum of resident set sizes (RSS) of the Python worker and all solver processes it launches, sampled at 20-millisecond intervals.

J Disclosure of AI Assistance

OpenAI Codex assisted with language editing, $\text{\LaTeX}$ formatting, and implementing, running, and checking experiment code under the authors' direction. The authors are responsible for the final manuscript, experiments, and references.