WHICH MODEL FOR WHICH ROLE? COST-EFFICIENT PARETO RECOMMENDATION FOR LLM AGENTIC WORKFLOWS

Anonymous authors
Paper under double-blind review

ABSTRACT

LLM agentic workflows increasingly employ different models for specialized roles. Choosing among these assignments requires balancing task quality against per-query API cost. Finding good assignments incurs a separate search cost from evaluating multiple candidates. We introduce *Cost-Coupled Gittins* (CC-GITTINS), which exploits a shared cost structure: assignments that cost more to run also cost more to evaluate. Gittins controllers for quality and cost share observations and independently relax their evaluation-cost penalties to allocate further evaluations. We evaluate CC-GITTINS by offline replay on eight quality–cost matrices from workflows with two to five roles and up to 1,681 candidate assignments. Across the eight quality–cost benchmarks, CC-GITTINS recovers a substantial fraction of the true Pareto frontier with few false positives at roughly 10% of the exhaustive-evaluation cost, and recovers most of the frontier by 30%.

1 INTRODUCTION

Large language model (LLM) agents increasingly combine multiple interacting roles, such as planning, solving, searching, and verification (Hua et al., 2026; Kim et al., 2026; Jiang et al., 2026). These roles need not use the same underlying model: a stronger model may be useful for one role, while a cheaper model may suffice for another. This flexibility creates a *model assignment problem*—given a set of candidate models for each role, which assignments provide the best trade-offs between end-to-end task quality and deployment cost? Since one role's output can affect the inputs and behavior of subsequent roles, an assignment's quality generally cannot be inferred by evaluating each model in isolation. The whole workflow must be treated as the unit of evaluation.

Existing work has not yet treated *budgeted Pareto recommendation* as the search objective for agent model assignments. TeamBench and AgentCARD benchmark heterogeneous agent teams and report empirical cost–quality frontiers over the assignments they evaluate (Kim et al., 2026; Jiang et al., 2026). AgentOpt (Hua et al., 2026) likewise reports the frontier over evaluated combinations, while its search algorithms optimize a fixed weighted utility. In these studies, the frontier follows from the chosen evaluations rather than guiding which costly assignment to evaluate next. SCOPE actively searches for a low-cost multi-module assignment subject to a specified quality threshold (Huang et al., 2026), while VineLM adapts model choices at individual stage invocations under request-level cost or latency budgets (Pagonas et al., 2026). These constraint-conditioned methods target an operating point, rather than a Pareto set of complete role assignments. What remains open is how to use a limited API evaluation budget to progressively identify and recommend nondominated assignments when the preferred quality–cost trade-off is not fixed in advance.

This matters since different users and deployment settings may prefer different points on the quality–cost trade-off. Exhaustive evaluation is costly: the number of complete assignments grows multiplicatively with the model choices available to different roles, and each candidate must be run on queries to estimate its quality and deployment cost. We therefore seek an anytime recommendation of nondominated model assignments while spending only a fraction of the API cost required to exhaustively evaluate the configuration space.

This problem has an additional structure that is easy to overlook. There are two distinct notions of cost. The *deployment cost* of a configuration is the mean API cost per query when that assignment is

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

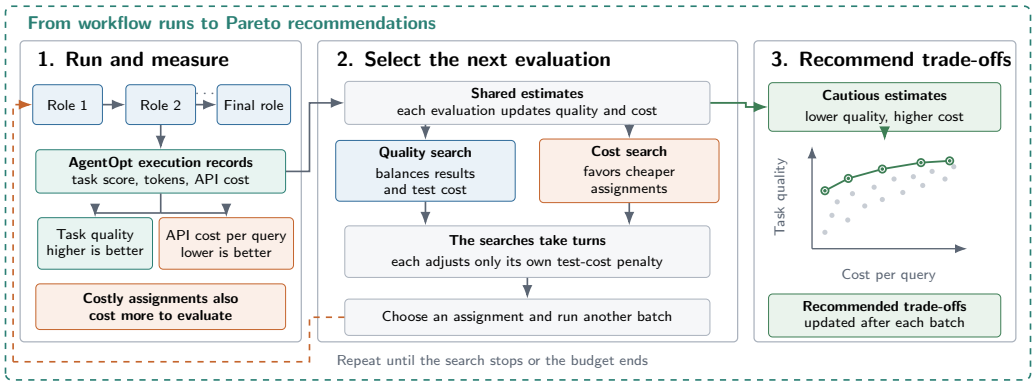

Figure 1: From workflow measurements to Pareto recommendations. Our client-side system records task scores and token use for each role assignment; model prices convert token use to per-query API cost. Costly assignments also cost more to evaluate. Quality-focused and cost-focused searches share these measurements, take turns choosing batches, and independently adjust their evaluation penalties after local stopping signals. After each batch, cautious estimates update the recommended trade-offs (green) among candidate assignments (gray).

eventually used. The *search cost* is the cumulative API cost spent evaluating candidate assignments before producing a recommendation, including configurations that are never deployed. These costs are naturally coupled: evaluating a candidate requires running essentially the same workflow that would later be deployed. For comparable benchmark and deployment queries, the cost of evaluating a batch of B examples for configuration i is therefore approximately $\kappa_i \approx BD_i$, where D_i is its mean deployment cost. Consequently, an expensive configuration is not only expensive to deploy; it is also expensive to learn about.

We exploit this structure with *Cost-Coupled Gittins* (CC-GITTINS), a sequential evaluation framework for Pareto recommendation. CC-GITTINS uses complementary scalar controllers associated with quality and deployment cost. The controllers share every workflow observation—each evaluated query provides information about both objectives—but independently adapt how strongly they penalize the cost of acquiring additional information. Intuitively, the quality controller searches for configurations whose additional deployment cost may be justified by improved task performance, while the cost controller preserves exploration of the inexpensive end of the frontier. As observations accumulate, CC-GITTINS maintains an anytime conservative Pareto recommendation rather than committing to a single configuration.

The role of cost coupling is particularly transparent in a deterministic limiting case. When uncertainty has vanished and configurations have matched remaining evaluation horizons, the quality controller ranks assignments according to $Q_i - \lambda D_i$ for an effective cost weight λ, whereas the cost controller ranks assignments by increasing deployment cost. Varying the effective information price can therefore expose different supported quality–cost trade-offs. This interpretation motivates the two complementary controllers, while not implying finite-budget recovery of every Pareto-optimal configuration. Figure 1 summarizes how these measurements lead to further evaluations and updated recommendations.

We evaluate CC-GITTINS by offline replay on eight workflow instances. We implement a client-side system to collect complete 10×10 assignment matrices for HotpotQA and MathQA. Six additional instances are derived from SCOPE workflows with three to five interacting roles and up to 1,681 retained configurations. Across these benchmarks, CC-GITTINS recovers useful portions of the exhaustive Pareto frontier at substantially lower evaluation cost. On HotpotQA, for example, recommendations at the 10% exhaustive-cost checkpoint achieve mean precision 0.93 and recall 0.65; at 30%, these increase to 0.98 and 0.90, respectively. Our experiments further compare against multi-objective search baselines and isolate the effects of cost coupling, controller directions, adaptive cost relaxation, and scheduling through controlled ablations.

Contributions. Our main contributions are as follows:

- We formulate model assignment in multi-role LLM workflows as a cost-aware Pareto recommendation problem under a limited API evaluation budget.
- We introduce CC-GITTINS, which exploits coupled deployment and evaluation costs through shared observations and independently adapting quality and cost controllers. A deterministic analysis characterizes their complementary quality–cost preferences.
- We implement a client-side agent optimization system and collect complete HotpotQA and MathQA assignment matrices with per-query task scores and token-derived API costs.
- We evaluate CC-GITTINS on eight quality–cost workflow instances at matched API budgets, comparing its Pareto recommendations with five search baselines and testing its components through controlled ablations.

Related work is discussed in Appendix A.

2 PARETO RECOMMENDATION AND EVALUATION

Configurations and objectives. A workflow has roles $1, \dots, L$, with candidate model set $\mathcal{M}_\ell$ for role ℓ. Let $\mathcal{C} = \mathcal{M}_1 \times \cdots \times \mathcal{M}_L$ be the set of complete model assignments and $K = |\mathcal{C}|$ its size. We number these configurations $1, \dots, K$; throughout, i identifies one complete assignment. Configuration i has task quality Q_i, to maximize, and mean deployment API cost per query D_i, to minimize. Since role outputs affect subsequent execution, the full assignment is the unit of evaluation. The target configuration set is

$$\mathcal{P}^\star = \{i : \nexists j \text{ with } Q_j \geq Q_i,\ D_j \leq D_i \text{ and at least one strict inequality}\}. \tag{1}$$

For offline replay, Q_i, D_i are the exhaustive means on the fixed N-question benchmark. They characterize that matrix; no population or distribution-shift guarantee is implied.

Geometry and evaluation spend. For geometry, map each objective to a fixed higher-is-better coordinate in $[0,1]^d$, using the origin as the hypervolume reference. In the quality–cost case, positive affine maps of $(Q_i, -D_i)$ give $y_i = (q_i, c_i) \in [0,1]^2$. Write $P^\star = \{y_i : i \in \mathcal{P}^\star\}$. Let $\widehat{\mathcal{P}}_t$ be the recommended configurations and $A_t = \{y_i : i \in \widehat{\mathcal{P}}_t\}$ their *true* objective vectors. These vectors are available to the offline evaluator, not the selector. Total search cost $S_t = \sum_{s \leq t} \kappa_s$ sums the batch charges κ_s across all evaluated configurations. For exhaustive search cost S_{full}, the recommendation at budget fraction b is

$$\widehat{\mathcal{P}}(b) = \widehat{\mathcal{P}}_{t(b)}, \qquad t(b) = \max\{t : S_t \leq bS_{\text{full}}\}. \tag{2}$$

We report a checkpoint only when a recommendation is available within that budget, including its warm-start cost.

Evaluation metrics. For an offline evaluator, let A be the nonempty set of recommended assignments at their true normalized coordinates, and let $d(x, S)$ be the Euclidean distance from x to the nearest point in S. Generational distance (GD) averages how far reported assignments lie from the true frontier. Inverted generational distance (IGD) averages how far true frontier points lie from the recommendation. Hypervolume regret measures the volume dominated by the true frontier but not by the recommendation. We use

$$\text{GD}(A, P^\star) = |A|^{-1} \sum_{a \in A} d(a, P^\star), \tag{3}$$

$$\text{IGD}(A, P^\star) = |P^\star|^{-1} \sum_{p^\star \in P^\star} d(p^\star, A), \tag{4}$$

$$R_{\text{HV}}(A) = \text{HV}(P^\star) - \text{HV}(A). \tag{5}$$

These average or volume measures assess recommendations at a given search spend; they do not require exact frontier recovery or limit the number of returned assignments. They are not worst-case guarantees for individual assignments or frontier regions. The stated distance convention retains *all* reported assignments; the existing replay plots use a filtered convention, as noted in Appendix G.

3 COST-COUPLED GITTINS

3.1 DEPLOYMENT COST ALSO PRICES EVALUATION

An additional benchmark batch for candidate i costs κ_i, contributing to total search spend S_t. For comparable benchmark and deployment queries in batches of size B, $\kappa_i \approx BD_i$. Realized batch charges can vary with question difficulty, output length, and the final batch size. The controller uses a positive warm-start estimate $\widehat{\kappa}_i$; budget accounting uses realized charges. This coupling need not hold when evaluation cost is unrelated to the deployment objectives, as in a generic quality–fairness problem.

In the two-objective specialization, the posterior models raw quality and per-query API cost (USD). Acquisition uses the affine desirability

$$Y_i = (Q_i, C_i), \qquad C_i = 1 - D_i/R, \qquad v_C = (0,1), \quad v_Q = (1,0), \tag{6}$$

where the positive scale R is frozen from the warm start. R need not bound all costs, so these acquisition coordinates need not lie in the unit cube. They are distinct from the fixed normalized coordinates used for offline evaluation in Section 2. The axes select C_i and Q_i directly, without radial utilities or division by direction components.

3.2 SHARED POSTERIOR AND SCALAR CONTROLLERS

Each batch supplies both objectives and updates a single state per configuration, shared by both controllers. With independent Gaussian posterior marginals $\theta_{i,j} \mid \mathcal{D}_{i,n} \sim \mathcal{N}(m_{i,j,n}, v_{i,j,n})$ for $j \in \{Q, D\}$, let $S_{i,j,n}$ be the sum of observed outcomes and σ_j^2 the fixed per-question residual variance. The predicted mean and variance of the final benchmark mean are

$$M_{i,j,n} = \frac{S_{i,j,n} + (N-n)m_{i,j,n}}{N}, \qquad U_{i,j,n} = \frac{(N-n)^2 v_{i,j,n} + (N-n)\sigma_j^2}{N^2}. \tag{7}$$

At completion, predictive variance is zero and the mean is the observed benchmark mean (Xie et al., 2026). For the cost axis, the predicted mean and variance are $M^C = 1 - M_D/R$ and $U^C = U_D/R^2$; for the quality axis, they are $M^Q = M_Q$ and $U^Q = U_Q$. Appendix C gives prior calibration.

For a scalar controller $a \in \{C, Q\}$, let k index batches and H_i be the completion stage. With outside option α, centered mean $x = M_{i,k}^a - \alpha$, positive per-batch information price γ, and mean-zero Gaussian increment $Z_{i,k}^a$, the required-completion recursion is

$$V_{i,H_i}^{a,\gamma}(x) = \max\{0, x\}, \tag{8}$$

$$W_{i,k}^{a,\gamma}(x) = -\gamma + \mathbb{E}[V_{i,k+1}^{a,\gamma}(x + Z_{i,k}^a)], \qquad V_{i,k}^{a,\gamma}(x) = \max\{0, W_{i,k}^{a,\gamma}(x)\}. \tag{9}$$

The increment variance is the reduction in U^a between adjacent stages. Intermediate abandonment earns the outside option; terminal utility requires completion. If $r_{i,k}^a(\gamma)$ is the zero of $W_{i,k}^{a,\gamma}$, the index is

$$G_{i,k}^Q = M_{i,k}^Q - r_{i,k}^Q(\eta_Q \widehat{\kappa}_i), \qquad G_{i,k}^C = M_{i,k}^C - r_{i,k}^C(\eta_C \widehat{\kappa}_i). \tag{10}$$

Root tables depend on the scalar variance schedule, horizon, and effective cost. We use separate positive multipliers η_Q, η_C, initially one. Scalar Gittins theory motivates each fixed-price controller (Xie et al., 2026); shared observations, interleaving, and the set-valued readout are additional design choices.

3.3 ASYNCHRONOUS COST RELAXATION

Controllers alternate *decision opportunities* in the order C, Q. On controller a's visit, compare the largest unfinished index with its best completed terminal value I_a (set $I_a = -\infty$ if no configuration is complete). If an unfinished configuration is preferred, acquire one batch. Otherwise, its fixed-η_a scalar rule gives a local stopping signal. The visited controller relaxes only its own information-price multiplier when possible; Appendix C specifies the boundary, continuation, and replay stopping rules. Thus 1:1 round-robin balances visits, not purchased batches or API expenditure; "asynchronous" refers to independent relaxation events, not concurrent execution.

3.4 WHY COST COUPLING EXPOSES QUALITY–COST TRADE-OFFS

At a fixed uncertainty schedule and remaining horizon, raising a positive evaluation price lowers the scalar index. When uncertainty states match, higher predicted quality and no greater evaluation cost likewise give no lower quality-axis index (Appendix B). Different uncertainty states can reverse this ordering, so it is not a dominance-pruning rule.

Proposition 1 (Deterministic roles of the two exact axes)**.** *Suppose uncertainty has vanished, each unfinished configuration requires the same h remaining full batches of size B to collect terminal utility, and each batch costs exactly $\kappa_i = BD_i$. Among unfinished configurations,*

$$\begin{aligned} G_i^Q(\eta_Q) &= Q_i - \lambda_Q D_i, & \lambda_Q &= \eta_Q hB, \\ G_i^C(\eta_C) &= 1 - \lambda_C D_i, & \lambda_C &= R^{-1} + \eta_C hB > 0. \end{aligned} \tag{11}$$

Thus the quality axis ranks a quality–cost scalarization, while the cost axis ranks configurations by increasing deployment cost.

The deterministic stopping root equals the remaining completion cost $h\eta_a\kappa_i$ (Appendix B). Coupling evaluation cost to deployment cost gives the quality-axis information price a deployment-cost weight, which can favor interior *supported* trade-offs: frontier points maximizing $Q_i - \lambda D_i$ for some $\lambda \geq 0$. The cost axis targets the low-cost endpoint through its terminal utility C_i; coupling changes its coefficient but, with matched horizons, not its ordering. This motivates the two exact axes but does not establish that they recover the full frontier.

Equation (11) is exact only for this limiting model. In the Bayesian policy, value of information, unequal horizons, and estimated or varying batch costs prevent an exact scalarization interpretation. Starting at $\eta_Q = 1$ and halving visits only a discrete subset of nonnegative scalarization weights and does not guarantee discovery of every supported point.

3.5 ANYTIME CONSERVATIVE PARETO RECOMMENDATION

Coverage alone does not exclude poor extra recommendations. Following the scalar LCB-style readout of Xie et al. (2026), use

$$Q_{i,n}^- = M_{i,Q,n} - \beta\sqrt{U_{i,Q,n}}, \qquad D_{i,n}^+ = M_{i,D,n} + \beta\sqrt{U_{i,D,n}}, \qquad \beta = 1. \tag{12}$$

Return all configurations nondominated when maximizing Q^- and minimizing D^+. Partially evaluated configurations remain eligible, and each is compared using its own conservative point. There is no restriction to one recommendation per controller. At exhaustive completion the readout equals the true benchmark frontier.

This adjustment aims to reduce unstable off-front recommendations but is not a simultaneous or time-uniform confidence certificate. It does not guarantee that every reported assignment lies near the true frontier. Neither this readout nor asynchronous relaxation confers Bayes optimality for a multi-objective Pareto-set utility.

4 EVALUATION PROTOCOL AND DIAGNOSTICS

Benchmarks. Our quality–cost evaluation uses eight exhaustive or replay-complete matrices across five workflow families. We evaluate $10^2 = 100$ assignments on all 200 questions of each of HotpotQA (Yang et al., 2018) (planner and search-enabled solver) and MathQA (Amini et al., 2019) (answerer and critic). Their ten candidate models use AWS Bedrock Application Inference Profiles; Appendix E.2 lists invocation prices. Six further instances use SCOPE's released query-level evaluations (Huang et al., 2026), covering three-to-five-role text-to-SQL, data-transformation, and data-imputation workflows. These configurations form subsets of the nominal assignment spaces, each evaluated on every benchmark question. All methods replay the same complete score–cost matrices (Appendix E.1).

Replay and metrics. The selector sees only purchased observations. The reported two-objective replay uses four warm questions per configuration, batches of up to four unseen questions, the exact axes $(0, 1), (1, 0)$, and independent per-controller relaxation. Runs that meet the configured stopping

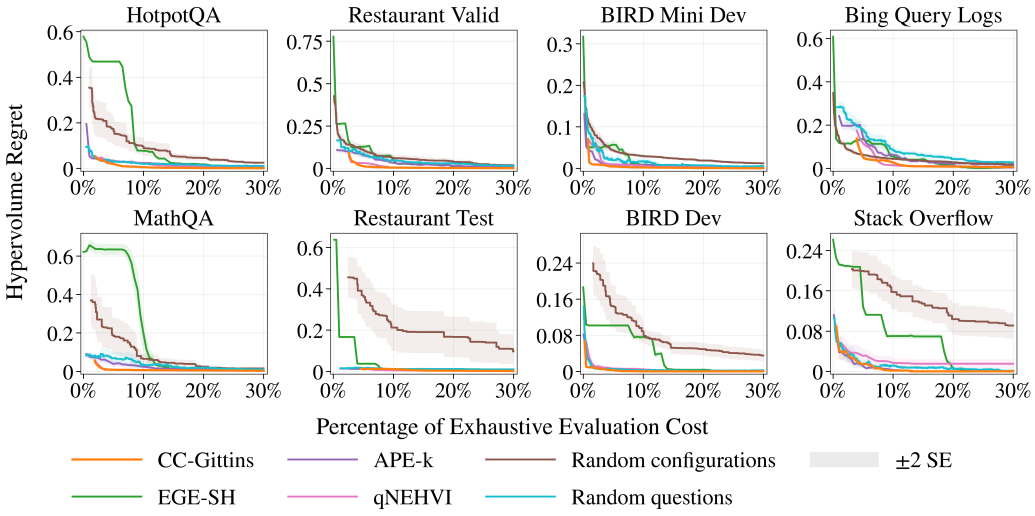

Figure 2: Hypervolume regret over 0–30% of exhaustive evaluation cost (lower is better). Curves aggregate 20 matched seeds, with qNEHVI exceptions in Appendix J; bands show ±2 standard errors. After stopping, CC-GITTINS carries its last recommendation forward without further sampling or spend.

condition carry their last recommendation to later budget targets; Appendix C gives the numerical rule. Comparisons use matched question-order seeds and account for the token-based API cost of selected observations, including warm-up. Equation (2) aligns checkpoints without overspending. We evaluate recommendations with GD, IGD, and HV regret, with precision, recall, and set size as supplementary exact-set diagnostics. Metric normalization is fixed across methods and seeds. Relative HV regret, if reported, divides the absolute regret by $\mathrm{HV}(P^\star)$. The existing GD/IGD plots use a filtered recommendation set; Appendix G explains this limitation.

What weighted scores can represent. In Proposition 1's deterministic setting, changing the quality controller's evaluation penalty changes its preference between quality and deployment cost, while the cost controller targets the cheapest end of the frontier. We test the geometry behind this choice using the complete benchmark matrices. For every nonnegative weight λ, we find the Pareto-optimal assignments that maximize $Q_i - \lambda D_i$; together they form the *supported subset*. Zero weight favors quality alone, while larger weights favor cheaper assignments. A nonconvex part of a discrete Pareto frontier may contain points that no weight selects. This offline oracle considers all weights and known outcomes, so it describes what weighted scores could represent, not what CC-GITTINS finds within a budget.

Across all eight instances, the supported subset loses less than 0.10% of full-frontier hypervolume, although it omits some Pareto points. Thus weighted scores can represent most dominated area on these matrices, but not every trade-off. This full-data geometry diagnostic does not guarantee what the policy will find with finite evaluations and a discrete sequence of prices. Appendix F gives the per-instance counts, distances, and hypervolume losses.

Baselines. We compare CC-GITTINS with EGE-SH for fixed-budget Pareto-set identification (Kone et al., 2024), the adaptive Pareto-exploration sampling rule of APE-k (Kone et al., 2023), and qNEHVI, a Bayesian optimization method that selects evaluations by noisy expected hypervolume improvement (Daulton et al., 2021). Two random search baselines, *random questions* and *random configurations*, sample either shared benchmark questions or assignments. Appendix E.4 gives their replay protocols.

Quality–cost comparisons. Figures 2, 3, and 4 show CC-GITTINS over 0–30% budget. Appendix H reports runtime; Appendices I and J give further recommendation plots.

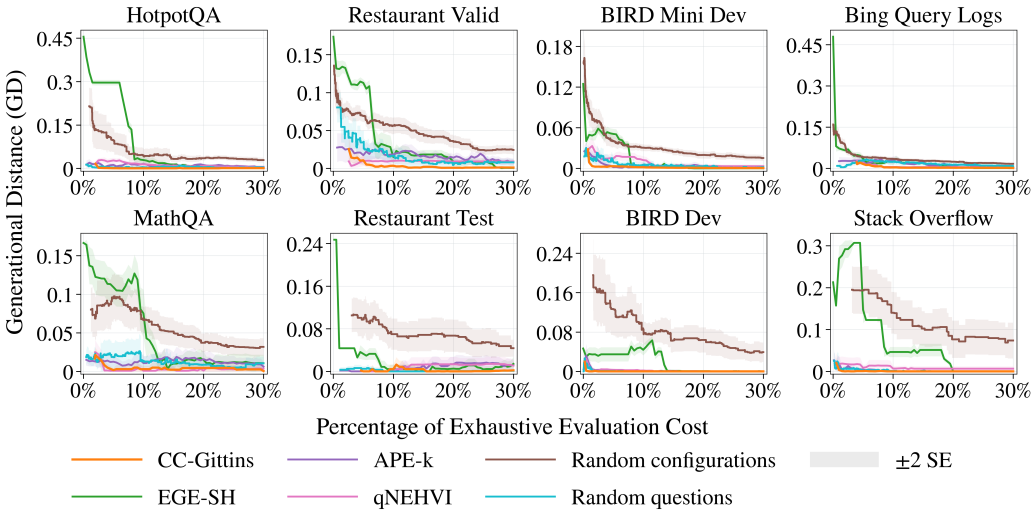

Figure 3: Generational distance on the same eight benchmarks over 0–30% budget. Bands show ±2 standard errors over the seed sets in Fig. 2. CC-GITTINS recommendations are carried forward after stopping; distances use the nondominated-subset convention in Appendix G.

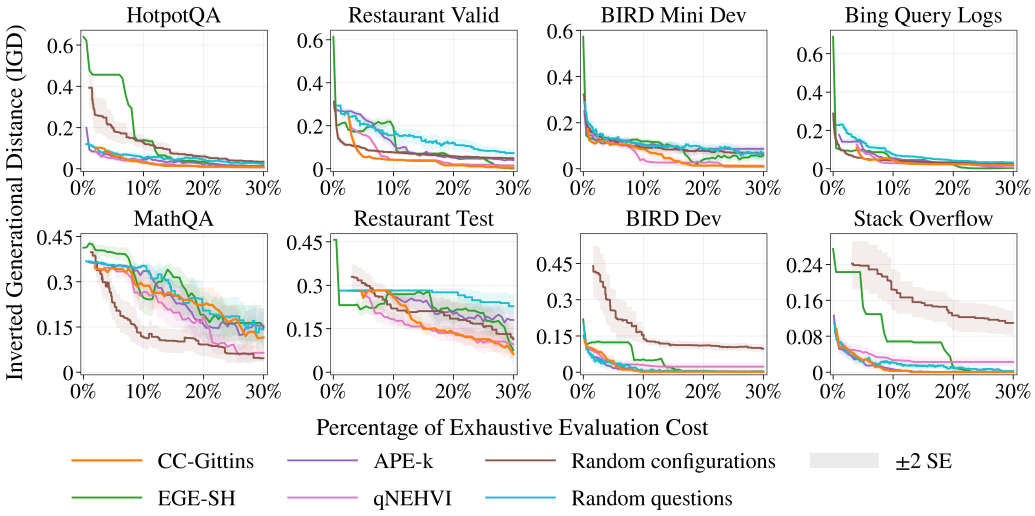

Figure 4: Inverted generational distance on the same eight benchmarks over 0–30% budget. Bands show ±2 standard errors over the seed sets in Fig. 2. CC-GITTINS recommendations are carried forward after stopping; distances use the nondominated-subset convention in Appendix G.

4.1 RECOMMENDATION QUALITY IMPROVES WITH SEARCH BUDGET

Figures 2, 3, and 4 plot recommendation quality through the 30% budget. At each checkpoint, they describe the choices available for the API cost spent, rather than requiring recovery of every frontier point. Falling HV regret shows increasing coverage of valuable trade-offs, low GD means the retained recommended configurations lie near the exhaustive frontier, and decreasing IGD shows that fewer frontier regions are missed.

Across the eight instances, recommendation quality generally improves over the displayed budget range, but the amount of frontier coverage at a fixed budget depends on the matrix. On the complete HotpotQA and MathQA matrices, the latest recommendations within 10% cost have mean precision of 0.93 and 0.94 and mean recall of 0.65 and 0.49, respectively. At 30%, precision rises to 0.98 and

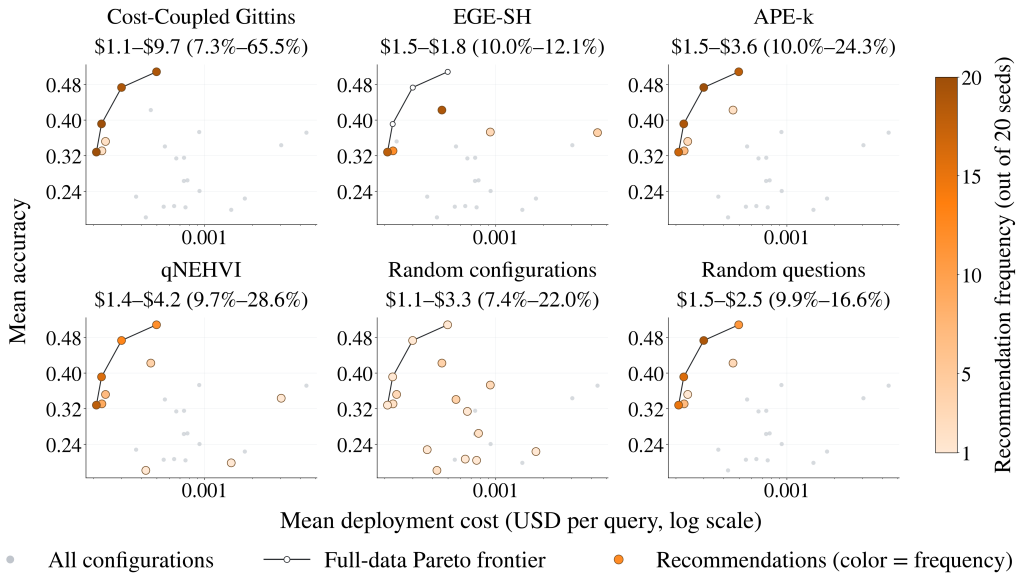

Figure 5: Stack Overflow Pareto recommendations at the 10% search-cost checkpoint for CC-GITTINS and five baselines. Positions show exhaustive mean accuracy and deployment cost. Gray points represent all configurations; black lines mark the full-data Pareto frontier. Marker color shows recommendation frequency across 20 matched seeds. Each method's label gives the mean start–end interval over which its recommendation remains stable, in USD and as a fraction of exhaustive search cost. The supplementary video shows the six methods' evolving recommendations on Stack Overflow over 0–50% of exhaustive API evaluation cost.

0.95 and recall to 0.90 and 0.66. The trajectories show the corresponding reductions in HV regret, while GD and IGD distinguish proximity from coverage.

Since the recommendation considers every configuration, it can contain more points than there are acquisition directions. New evidence can add trade-offs or remove configurations that become dominated. Figure 5 compares CC-GITTINS with all five Pareto baselines on Stack Overflow at the 10% budget target; the panels use the same full-data landscape and show recommendation frequency across 20 seeds. The supplementary video shows how the recommended Pareto sets of these six methods evolve on Stack Overflow over 0–50% of exhaustive API evaluation cost, allowing a visual comparison of recommendation quality at matched search budgets.

Checkpoint convention. In the reported replay, some runs stop acquiring evaluations before later budget targets and carry their last recommendation forward without further spend or observations. The horizontal coordinate is therefore available budget, not necessarily realized spend. Reported GD and IGD retain each recommendation's true-coordinate nondominated subset, as detailed in Appendix G.

5 DISCUSSION AND CONCLUSIONS

CC-GITTINS prices benchmark evaluation through deployment cost. In the deterministic equal-horizon limit, this makes the quality axis rank $Q - \lambda D$, while the cost axis ranks by deployment cost. The result motivates the exact-axis pair but does not guarantee full-front recovery under finite-budget stochastic search. Across eight replay matrices, recommendation quality generally improves as the available API search budget rises through 30% of exhaustive evaluation cost, although frontier coverage varies by instance. Appendix D reports a limited exploratory replay with latency as a third objective on HotpotQA and MathQA.

Reproducibility statement

The main text specifies the acquisition and stopping rules and evaluation protocol. Appendix E documents the benchmark matrices, experimental settings, and baseline details; Appendix C describes numerical index computation. Code and the video accompanying Figure 5 are provided in the supplementary material.

AI use statement

The authors used generative AI tools, including LLM-based assistants, to draft and revise text, assist with mathematical derivations, and format the manuscript in LaTeX. For Proposition 1, the authors supplied the underlying idea and a proof sketch, and AI tools helped develop the detailed derivation and proof. The authors checked the cited sources and AI-assisted mathematical arguments and text, revised them as needed, and take full responsibility for the paper's contents.

Separately, LLMs are components of the agent configurations evaluated in this work. The quality–cost experiments replay HotpotQA and MathQA response matrices collected with our client-side system and six SCOPE-derived instances. The replay makes no new model API calls. Appendix E.1 documents data sources and processing.

References

Aida Amini, Saadia Gabriel, Shanchuan Lin, Rik Koncel-Kedziorski, Yejin Choi, and Hannaneh Hajishirzi. MathQA: Towards interpretable math word problem solving with operation-based formalisms. In *Proceedings of NAACL-HLT*, pp. 2357–2367, 2019. doi: 10.18653/v1/N19-1245.

Lingjiao Chen, Matei Zaharia, and James Zou. FrugalGPT: How to use large language models while reducing cost and improving performance. *arXiv preprint arXiv:2305.05176*, 2023.

Samuel Daulton, Maximilian Balandat, and Eytan Bakshy. Differentiable expected hypervolume improvement for parallel multi-objective Bayesian optimization. In *Advances in Neural Information Processing Systems*, volume 33, pp. 9851–9864, 2020. URL https://papers.neurips.cc/paper_files/paper/2020/hash/6fec24eac8f18ed793f5eaad3dd7977c-Abstract.html.

Samuel Daulton, Maximilian Balandat, and Eytan Bakshy. Parallel Bayesian optimization of multiple noisy objectives with expected hypervolume improvement. In *Advances in Neural Information Processing Systems*, volume 34, pp. 2187–2200, 2021. URL https://proceedings.neurips.cc/paper/2021/hash/11704817e347269b7254e744b5e22dac-Abstract.html.

Qitian Jason Hu, Jacob Bieker, Xiuyu Li, Nan Jiang, Benjamin Keigwin, Gaurav Ranganath, Kurt Keutzer, and Shriyash Kaustubh Upadhyay. RouterBench: A benchmark for multi-LLM routing system. *arXiv preprint arXiv:2403.12031*, 2024.

Wenyue Hua, Sripad Karne, Qian Xie, Armaan Agrawal, Nikos Pagonas, Kostis Kaffes, and Tianyi Peng. AgentOpt v0.1 technical report: Client-side optimization for LLM-based agent. *arXiv preprint arXiv:2604.06296*, 2026.

Yiqian Huang, Shiqi Zhang, Tianyuan Jin, and Xiaokui Xiao. SCOPE: Cost-efficient model selection for compound AI systems under quality constraints. In *Proceedings of the 32nd ACM SIGKDD Conference on Knowledge Discovery and Data Mining V. 2*, pp. 1922–1931. Association for Computing Machinery, 2026. doi: 10.1145/3770855.3818067.

Yinsicheng Jiang, Liang Cheng, Yeqi Huang, Yufan Zhao, Zhan Lu, Li Dong, Wenda Li, Edoardo Ponti, and Luo Mai. Specialize roles, mix deployments: Pushing the cost–accuracy frontier of LLM agent teams. *arXiv preprint arXiv:2606.20629*, 2026.

Yubin Kim, Chanwoo Park, Taehan Kim, Eugene Park, Samuel Schmidgall, Salman Rahman, Chunjong Park, Cynthia Breazeal, Xin Liu, Hamid Palangi, Hae Won Park, and Daniel McDuff. TeamBench: Evaluating agent coordination under enforced role separation. *arXiv preprint arXiv:2605.07073*, 2026.

Cyrille Kone, Emilie Kaufmann, and Laura Richert. Adaptive algorithms for relaxed Pareto set identification. In *Advances in Neural Information Processing Systems*, volume 36, pp. 35190–35201, 2023. doi: 10.52202/075280-1528. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/6e976e7930460b5c3167a104ba8cc39c-Abstract-Conference.html.

Cyrille Kone, Emilie Kaufmann, and Laura Richert. Bandit Pareto set identification: The fixed budget setting. In *Proceedings of the 27th International Conference on Artificial Intelligence and Statistics*, volume 238 of *Proceedings of Machine Learning Research*, pp. 2548–2556. PMLR, 2024. URL https://proceedings.mlr.press/v238/kone24a.html.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model serving with PagedAttention. In *Proceedings of the 29th Symposium on Operating Systems Principles*, pp. 611–626, 2023.

Michael Luo, Xiaoxiang Shi, Colin Cai, Tianjun Zhang, Justin Wong, Yichuan Wang, Chi Wang, Yanping Huang, Zhifeng Chen, Joseph E. Gonzalez, et al. Autellix: An efficient serving engine for LLM agents as general programs. *arXiv preprint arXiv:2502.13965*, 2025.

Zifan Lyu, Chahine Nejma, Tobias Wegel, Fanny Yang, and Florian E. Dorner. Cutting LLM evaluation costs with SySRs: A bandit algorithm that provably exploits model similarity. In *International Conference on Machine Learning*, 2026.

Isaac Ong, Amjad Almahairi, Vincent Wu, Wei-Lin Chiang, Tianhao Wu, Joseph E. Gonzalez, M Waleed Kadous, and Ion Stoica. RouteLLM: Learning to route LLMs from preference data. In *International Conference on Learning Representations*, volume 2025, pp. 34433–34448, 2025.

Nikos Pagonas, Matthew Lou, Tianyi Peng, Dan Rubenstein, and Kostis Kaffes. VineLM: Trie-based fine-grained control for agentic workflows. *arXiv preprint arXiv:2605.23914*, 2026.

Felipe Maia Polo, Ronald Xu, Lucas Weber, Mírian Silva, Onkar Bhardwaj, Leshem Choshen, Allysson Flavio Melo de Oliveira, Yuekai Sun, and Mikhail Yurochkin. Efficient multi-prompt evaluation of LLMs. In *Advances in Neural Information Processing Systems*, volume 37, pp. 22483–22512, 2024. doi: 10.52202/079017-0707.

Qian Xie, Yueli He, and Nairen Cao. Efficient cost-aware LLM evaluation via Bayesian bandit Gittins indices. In *ICML 2026 Workshop on Decision-Making from Offline Datasets to Online Adaptation: Black-Box Optimization to Reinforcement Learning*, 2026. URL https://arxiv.org/abs/2609.25645.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. HotpotQA: A dataset for diverse, explainable multi-hop question answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pp. 2369–2380, 2018. doi: 10.18653/v1/D18-1259.

Lianmin Zheng, Liangsheng Yin, Zhiqiang Xie, Chuyue Sun, Jeff Huang, Cody Hao Yu, Shiyi Cao, Christos Kozyrakis, Ion Stoica, Joseph E. Gonzalez, et al. SGLang: Efficient execution of structured language model programs. *Advances in Neural Information Processing Systems*, 37: 62557–62583, 2024. doi: 10.52202/079017-2000.

Jin Peng Zhou, Christian K. Belardi, Ruihan Wu, Travis Zhang, Carla P. Gomes, Wen Sun, and Kilian Q. Weinberger. On speeding up language model evaluation. In *International Conference on Learning Representations (ICLR)*, 2025.

A Related Work

Resource-aware LLM systems. Server-side systems improve scheduling, memory use, and serving throughput (Kwon et al., 2023; Zheng et al., 2024; Luo et al., 2025). Client-side routing selects a model per request or call (Chen et al., 2023; Hu et al., 2024; Ong et al., 2025). Agent configuration search instead assigns models to interacting roles and observes an end-to-end trajectory reward. TeamBench exhaustively studies the assignments of several providers to Planner, Executor, and Verifier (Kim et al., 2026), while AgentCARD maps deployment-aware cost–accuracy frontiers over multi-role assignments (Jiang et al., 2026). These systems characterize configuration landscapes after evaluation.

Efficient LLM evaluation and constrained selection. BanditEval exploits low-rank response-matrix structure (Zhou et al., 2025), SySRs uses paired comparisons to identify the best model (Lyu et al., 2026), and PromptEval shares information across prompts and examples to estimate multi-prompt performance (Polo et al., 2024). Gittins indices have also been adapted to account for evaluation costs in LLM model selection (Xie et al., 2026). SCOPE and VineLM both consider multiple objectives but formulate selection as constrained optimization rather than Pareto-set recommendation: SCOPE uses per-query partial evaluations to minimize deployment cost subject to a quality threshold (Huang et al., 2026), while VineLM uses sparse profiling and stage-level replanning to maximize request accuracy under cost or latency budgets (Pagonas et al., 2026).

Pareto-set search. General-purpose multi-objective bandits study Pareto-set identification under fixed-confidence or fixed-budget objectives (Kone et al., 2023; 2024). Hypervolume-based Bayesian optimization, including qEHVI and qNEHVI, selects evaluations by expected hypervolume improvement (Daulton et al., 2020; 2021). These methods can be sample-efficient, but their standard acquisition criterion does not explicitly price the assignment-dependent API cost of each evaluation. CC-Gittins includes that price in its scalar indices and updates its recommendation after each batch; we assess those recommendations at matched cumulative API spend.

B Scalar-Index Derivations and Scope

Price monotonicity of the scalar recursion. Fix the transition schedule and terminal stage, and take $0 < \gamma_1 \le \gamma_2$. At the terminal stage $V_H^{\gamma_1} = V_H^{\gamma_2} = x_+$. If $V_{k+1}^{\gamma_1} \ge V_{k+1}^{\gamma_2}$ pointwise, then

$$W_k^{\gamma_1}(x) = -\gamma_1 + \mathbb{E}[V_{k+1}^{\gamma_1}(x + Z_k)] \ge -\gamma_2 + \mathbb{E}[V_{k+1}^{\gamma_2}(x + Z_k)] = W_k^{\gamma_2}(x). \tag{13}$$

Taking the maximum with zero preserves the inequality and proves it by backward induction. Continuation values are nondecreasing in x; consequently the nonnegative-continuation set for the higher price is contained in that for the lower price, and its threshold satisfies $r_k(\gamma_1) \le r_k(\gamma_2)$. With identical schedules,

$$G_i^Q - G_j^Q = (M_i^Q - M_j^Q) + r_k(\eta\widehat{\kappa}_j) - r_k(\eta\widehat{\kappa}_i) \ge 0. \tag{14}$$

We restrict to positive prices. At exactly zero cost, a nondegenerate Gaussian transition can make continuation strictly positive at every finite x, so a finite stopping root need not exist. This also motivates the positive-price and underflow safeguards below. □

Proof of Proposition 1. With zero uncertainty, $Z_k = 0$. For $h = 0$, the excess terminal value is $V_0(x) = x_+$. If $V_{h-1}(x) = \max\{0, x - (h-1)\gamma\}$, then

$$V_h(x) = \max\{0, -\gamma + \max\{0, x - (h-1)\gamma\}\} = \max\{0, x - h\gamma\}. \tag{15}$$

Thus the continuation root is $h\gamma$. With quality terminal utility Q_i, the index is $G_i^Q = Q_i - h\eta_Q\kappa_i = Q_i - \eta_Q hBD_i$. With cost terminal utility $C_i = 1 - D_i/R$, the same recursion gives $G_i^C = C_i - h\eta_C\kappa_i = 1 - (R^{-1} + \eta_C hB)D_i$. Since $R > 0$ and $\eta_C > 0$, the cost-axis coefficient is positive. Required completion is essential: if a known terminal reward could be collected immediately, paying the remaining h batches would be unnecessary. Unequal remaining horizons instead give configuration-specific penalties $\eta_a h_i BD_i$, not a common scalarization weight. □

C Posterior Calibration and Scalar Computation

Warm-start batches use independent seeded question orders for each configuration. The prior means are equal-weight averages of configuration warm-batch means. Following the existing empirical-target implementation, quality prior variance is fixed at $v_{0,Q} = 0.04$. If $\bar{D}_i^{\mathrm{warm}}$ is the warm-batch mean cost and $K > 1$, use

$$v_{0,D} = \max\left\{\frac{1}{K-1}\sum_{i=1}^{K}\left(\bar{D}_i^{\mathrm{warm}} - \frac{1}{K}\sum_{j=1}^{K}\bar{D}_j^{\mathrm{warm}}\right)^2, 10^{-12}R^2\right\}. \tag{16}$$

Freeze calibration after warm-up and include each warm observation in the likelihood once. Empirical-Bayes reuse of warm data and the independent Gaussian approximation are modeling choices, not coverage certificates.

Equation (9) is one-dimensional on either exact axis. Only the active objective's benchmark-mean innovation variance enters its scalar dynamic program; the unused objective is still observed and updates the shared posterior. Root tables are cached by variance schedule, horizon, and positive effective cost. The quality and cost axes therefore do not require the expected-minimum formula or a two-dimensional radial dynamic program. A shorter final batch is a separate stage; exact heterogeneous stage charges can be handled by a stage-dependent penalty. Proposition 1 assumes equal-sized, equal-price remaining batches.

Numerical stopping rule. The replay does not impose a fixed lower bound on η_a. Let $G^a_{\max}$ be the best unfinished index; the implementation treats $G^a_{\max} \le I_a + \tau$ as a local scalar stop. At such a stop, let $\rho_a = \max_{i \text{ unfinished}} h_i\gamma_{i,a}$ be the largest remaining effective completion penalty, where h_i is the number of planned batches remaining and $\gamma_{i,a}$ is the controller's current effective per-batch price. The numerical threshold is

$$\delta_a = \max\{\tau, \epsilon_{\mathrm{mach}}\max(1, |I_a|, |G^a_{\max}|)\}, \tag{17}$$

where ϵ_{mach} is double-precision machine epsilon and the available implementation defaults to $\tau = 10^{-9}$. If $\rho_a > \delta_a$ and halving keeps the multiplier and effective costs positive, the visited controller halves η_a and yields without an evaluation. Otherwise, it records a numerical stop, including when the next halving would underflow. Let F collect the current numerical-stop marks. A newly purchased batch clears F. Acquisition ends when both controllers are marked without an intervening observation, the external budget is reached, or all configurations are complete. A numerical stop is not statistical certification. Algorithm 1 gives the replay rule.

Algorithm 1 CC-Gittins replay: quality–cost specialization

Require: configurations, axes $v_C = (0,1), v_Q = (1,0)$, batch size B, external budget, tolerance τ
1: acquire warm batches; freeze the prior, R, and $\widehat{\kappa}_i$; set $\eta_C = \eta_Q = 1$ and $F = \varnothing$
2: publish the conservative Pareto set using Eq. (12)
3: **while** the external budget permits evaluation, unfinished configurations remain, and $F \neq \{C, Q\}$ **do**
4: $a \leftarrow$ next controller in C, Q round-robin order
5: $i \leftarrow \arg\max_{j \text{ unfinished}} G^a_{j,k_j}$; obtain completed incumbent I_a
6: **if** controller a has a local scalar stop **then**
7: compute remaining penalty ρ_a and threshold δ_a
8: **if** $\rho_a > \delta_a$ and the next halving keeps effective costs positive **then**
9: $\eta_a \leftarrow \eta_a/2$; remove a from F
10: **else**
11: add a to F
12: **end if**
13: **continue** to the next visit
14: **end if**
15: acquire i's next batch, charge realized cost, and update both posterior marginals
16: $F \leftarrow \varnothing$; publish the updated conservative Pareto set
17: **end while**
18: **return** the last recommendation within the requested budget

D EXPLORATORY QUALITY–COST–LATENCY REPLAY

This supplementary study adds recorded workflow latency to quality and deployment API cost on the HotpotQA and MathQA matrices. It tests one possible extension on two instances; the main empirical evidence remains the eight quality–cost instances. The results do not establish general higher-dimensional performance or a preferred number of controllers.

Controller choice. Let L_i be mean end-to-end workflow latency and $\ell_i = 1 - L_i/T$, with a positive scale T fixed after warm-up. In acquisition coordinate order (Q, C, ℓ), the round-robin sequence is pure quality $(1, 0, 0)$, balanced quality–latency $(\frac{1}{2}, 0, \frac{1}{2})$, pure latency $(0, 0, 1)$, and pure cost $(0, 1, 0)$. The pure directions use their scalar desirabilities. The balanced direction uses the radial terminal utility

$$U_{\mathrm{bal}}(i) = \min\{Q_i, \ell_i\}. \tag{18}$$

The equal direction weights specify equal scaling before taking the minimum, not an arithmetic average. This fourth controller is an empirical choice. Proposition 1's linear supported-front interpretation does not imply coverage by this radial controller set.

Pilot acquisition. The implementation assumes independent Gaussian posteriors for the latent raw means. Acquisition uses their posterior means and variances: scalar dynamic programs on the pure axes and a two-coordinate radial dynamic program for the balanced utility. Terminal acquisition values are posterior expected latent utilities. This differs from the finite-benchmark acquisition specification in the main text; the finite-benchmark bridge in Eq. (7) is applied separately to construct recommendations. No cross-objective covariance is fitted.

The controllers share every observation, take turns, and independently relax positive evaluation-cost penalties. Acquisition uses warm-estimated API batch costs rounded to the nearest multiplicative bin, with ratio 2 and anchor 10^{-4} USD. Budget accounting uses realized, unrounded API spend. We apply the numerical-stop rule of Appendix C to all active controllers: a new observation clears their stop marks, and numerical termination requires all controllers to be marked without an intervening observation. Runs also end at the external budget or after all configurations are complete. A stopped run carries its final recommendation to later budget targets without additional spend.

Recommendation and offline metrics. For the readout, apply Eq. (7) to quality, cost, and latency. Augment Eq. (12) with $L^+_{i,n} = M_{i,L,n} + \beta\sqrt{U_{i,L,n}}$, and return configurations nondominated when maximizing Q^- and minimizing D^+, L^+. This remains a heuristic conservative readout. The random baselines use empirical three-objective Pareto sets, so comparisons include differences in both acquisition and readout.

For offline evaluation only, map the true means to

$$y_i^{(3)} = \left(Q_i, \frac{R_L}{R_L + L_i}, \frac{R_D}{R_D + D_i}\right), \tag{19}$$

where R_L, R_D are medians of strictly positive full-data assignment means for latency and API cost. These nonlinear reciprocal scales do not enter acquisition. We compute absolute three-dimensional HV regret with the origin reference, and Euclidean GD/IGD after retaining each recommendation's true-coordinate nondominated subset. Thus GD/IGD do not assess every returned assignment. The true three-dimensional frontiers contain 23 points on HotpotQA and 17 on MathQA; no three-dimensional supported-front oracle is reported.

Replay protocol. Each matrix has 100 assignments and 200 questions, with recorded quality, token-priced API cost, and end-to-end latency for every cell. Comparisons use 20 matched seeds. The four-controller variant uses four warm questions per assignment and batches of up to four unseen questions. Recorded latency measures workflow execution time rather than optimizer overhead and is specific to the collection environment. At each budget fraction, we use the latest available recommendation whose realized API spend does not exceed the budget, including warm-up. The random-question baseline evaluates shared question prefixes; random configurations evaluates complete assignment prefixes. Figure 6 compares the four-controller variant with three-objective

EGE-SH, APE-k, qNEHVI, and these two random baselines. Bands are ± 2 sample standard errors, clipped below at zero; GD/IGD are omitted where any seed has no nonempty recommendation within budget.

Exploratory results. Figure 6 shows decreasing HV regret for the tested variant on both matrices over the displayed budget range, while distance-based comparisons are mixed, particularly for IGD on MathQA. This limited replay illustrates the evaluation-and-recommendation procedure with an additional objective; it does not demonstrate uniform superiority across metrics or broader workflow families.

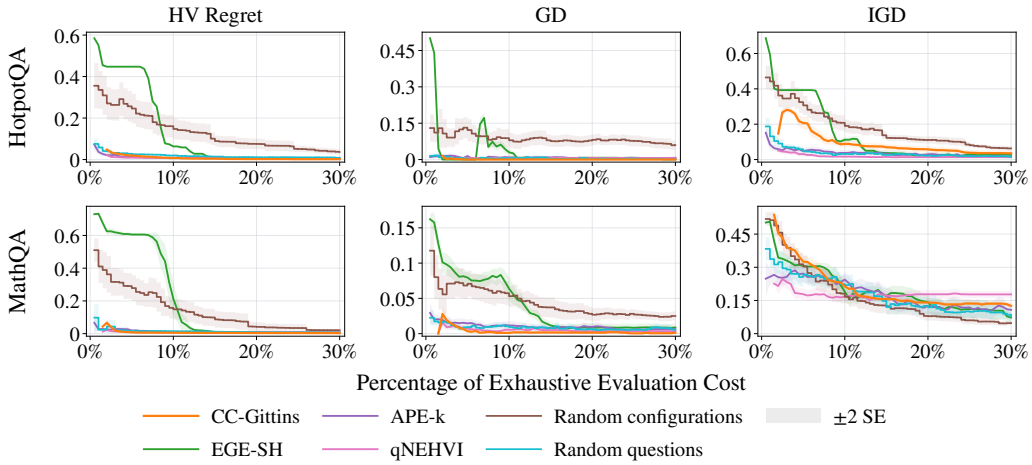

Figure 6: Exploratory three-objective replay on HotpotQA (top) and MathQA (bottom) over 0–30% of exhaustive API evaluation cost. Columns show absolute HV regret, GD, and IGD; lower is better. Curves show means over 20 matched seeds, with ± 2 standard-error bands. The comparison includes four-direction CC-GITTINS, EGE-SH, APE-k, qNEHVI, random questions, and random configurations. Recommendations are carried forward without added spend after a run stops.

Three versus four controllers. Removing the balanced controller leaves the pure Q, ℓ, and C axes. Table 1 compares these controller sets at the 10% and 30% API-cost checkpoints. Adding the fourth controller also changes each pure axis's share of round-robin visits, so this is a controller-set comparison rather than an isolated utility ablation. Four controllers yield lower mean HV regret only on HotpotQA at 10%, and lower mean IGD only on MathQA at 10%. At 30%, the three pure axes have lower means for both metrics on both matrices. These means do not establish a general benefit from the fourth controller; the table does not report uncertainty or a significance test.

Table 1: Exploratory three-objective controller-set ablation over 20 matched seeds. Values are means; relative HV regret is a percentage of full-frontier HV. Lower is better. The four-direction variant adds the balanced radial-min quality–latency controller to the three pure axes.

		Relative HV regret (%)		IGD	
Dataset	Budget	Four directions	Pure axes	Four directions	Pure axes
MathQA	10%	0.65	0.60	0.213	0.287
MathQA	30%	0.39	0.35	0.126	0.104
HotpotQA	10%	1.13	1.45	0.089	0.084
HotpotQA	30%	0.26	0.24	0.036	0.024

E Experimental Setup and Implementation Details

E.1 Benchmark Matrices

Each response matrix has one row per retained role-level model assignment and one column per question. We collect complete HotpotQA and MathQA matrices by evaluating all $10^2 = 100$ assignments of ten candidate models to two roles on all 200 questions, or 20,000 observed cells per benchmark. The two workflow settings follow those described in AgentOpt (Hua et al., 2026). The calculated exhaustive API costs are \$62.58 and \$234.34, respectively. Their full-data Pareto frontiers contain 15 and 6 assignments, and their maximum mean accuracies are 74.66% and 97.50%. The six SCOPE matrices use fully evaluated configuration rows from SCOPE's released per-query cache for three data-processing systems (Huang et al., 2026). Every retained SCOPE configuration has score and token records for all questions in its corresponding benchmark split. These configurations cover a subset of the nominal Cartesian product of role-level model choices; we do not infer results for unobserved configurations. Table 2 reports the retained dimensions used by every compared method. Thus, the exhaustive-evaluation cost and full-data Pareto frontier for each instance refer to its retained matrix; paired development and test instances may contain different assignment sets.

Table 2: Benchmark matrices used in the eight-instance evaluation. We collected HotpotQA and MathQA observations with the client-side system; the other six instances use SCOPE's released evaluation records. "Configurations" counts retained role-level model assignments (the arms of the search problem), and "Questions" counts each instance's evaluation horizon, covered by every retained configuration.

Benchmark instance	Workflow	Roles	Configurations	Questions
HotpotQA	Planner–solver	2	100	200
MathQA	Answerer–critic	2	100	200
BIRD Mini Dev	DIN-SQL	4	1,115	500
BIRD Dev	DIN-SQL	4	75	1,534
Bing Query Logs	UniDM-DT	5	1,681	102
Stack Overflow	UniDM-DT	5	23	710
Restaurant Valid	UniDM-DI	3	972	156
Restaurant Test	UniDM-DI	3	36	86

At the question level, HotpotQA uses a continuous token-level F1 score in $[0, 1]$. Since HotpotQA produces free-form answers, token-level F1 preserves partial credit for token overlap that binary exact match would discard. MathQA uses a binary correctness indicator, and each released per-query evaluation in the six SCOPE-derived instances likewise provides a binary task-success outcome. Consequently, configuration-level mean performance is mean F1 for HotpotQA and the empirical success rate for the other seven instances.

E.2 Model Pricing and Cost Accounting

Table 3 lists the ten models in the HotpotQA and MathQA matrices and the AWS Bedrock US standard on-demand prices used for cost accounting.[1] Each workflow execution's API cost is the sum, across both roles, of its recorded input and output token counts multiplied by the corresponding per-million-token rates. The GPT-5.6 Luna rate uses the US commercial short-context tier (up to 272K input tokens per request). The six SCOPE-derived matrices use the token records and model API prices in their released evaluation cache.

[1] https://aws.amazon.com/bedrock/pricing/; https://docs.aws.amazon.com/bedrock/latest/userguide/model-card-openai-gpt-56-luna.html.

Table 3: AWS Bedrock US standard on-demand input and output prices used for the ten models in our HotpotQA and MathQA evaluations, in USD per million tokens.

Model	Input	Output
Claude Haiku 4.5	1.00	5.00
Claude Opus 4.6	5.00	25.00
DeepSeek R1	1.35	5.40
GPT-5.6 Luna	0.22	1.32
gpt-oss-120b	0.15	0.60
gpt-oss-20b	0.07	0.30
Kimi K2.5	0.60	3.00
Ministral 3 8B	0.15	0.15
Qwen3 32B	0.15	0.60
Qwen3 Next 80B A3B	0.15	1.20

E.3 Two-objective CC-Gittins configuration

In practice, evaluation queries can be run in batches of size B to reduce wall-clock latency through parallel execution; $B = 1$ is also valid.

Table 4: Two-objective CC-Gittins settings for the reported replay.

Component	Setting
Warm start / adaptive batch	4 / up to 4 questions per configuration
Statistical target	Mean of the fixed benchmark
Cost coordinates	Raw API cost (USD); $C = 1 - D/R$ for acquisition
Directions (in visit order)	$(0, 1)$ cost, $(1, 0)$ quality
Scheduler	1:1 decision opportunities, shared observations
Multipliers	$\eta_C = \eta_Q = 1$; independent halving
Numerical safeguard	Precision threshold in Eq. (17); underflow check
Termination	Both local numerical stops, external budget, or exhaustive completion
Recommendation	Conservative Pareto, $\beta = 1$, no set-size cap
Budget accounting	Token-based API cost (USD), including warm start

E.4 Baseline implementations

All five baselines use the same frozen response matrices but follow their own sampling protocols; checkpoints are aligned by realized cumulative API search cost. EGE-SH allocates observations in rounds and uses empirical Pareto gaps to halve the active assignments (Kone et al., 2024). We use the confidence-bound sampling rule of APE-k to direct observations toward uncertain Pareto comparisons; its fixed-confidence stopping condition is disabled, and sampling continues until the replay budget or available observations are exhausted (Kone et al., 2023). qNEHVI uses a Bayesian surrogate over categorical assignment features and chooses evaluations by noisy expected hypervolume improvement (Daulton et al., 2021).

The two random search baselines differ in the sampled unit: *random questions* evaluates all assignments on the same randomly drawn questions, while *random configurations* fully evaluates randomly drawn assignments. Both recommend the empirical Pareto set from their observed outcomes.

F Supported-Front Oracle Diagnostic

Table 5 reports what the full-data weighted-score family can represent on each matrix. For HotpotQA, 6 of the 15 true Pareto points are supported. Keeping those six loses only 0.00066 of absolute hypervolume, although the furthest omitted frontier point is 0.0716 away in normalized coordinates. Stack Overflow and Restaurant Test have no unsupported frontier points. This offline oracle considers all nonnegative weights and known outcomes; it does not measure finite-budget CC-Gittins recommendations.

Table 5: Complete-data supported-front diagnostic. Pareto and Supported count distinct objective outcomes on the true frontier and in its supported subset. Max gap and IGD are the largest and average distances, respectively, from a true Pareto point to the nearest supported point; HV regret is the absolute dominated-area loss from keeping only supported points. Distances and HV use $(Q, 1 - D/(1.05 \max_i D_i))$ with origin reference. This is not a budgeted CC-GITTINS result.

Instance	Pareto	Supported	Max gap	IGD	HV regret
HotpotQA	15	6	0.0716	0.0146	0.00066
MathQA	6	4	0.0232	0.0072	0.00046
BIRD Mini Dev	11	7	0.0598	0.0087	0.00016
BIRD Dev	8	4	0.0520	0.0141	0.00018
Bing Query Logs	16	7	0.0687	0.0234	0.00042
Stack Overflow	4	4	0.0000	0.0000	0.00000
Restaurant Valid	10	7	0.0149	0.0035	0.00010
Restaurant Test	3	3	0.0000	0.0000	0.00000

G CC-GITTINS REPLAY RESULTS

Protocol and scope. CC-GITTINS uses the exact cost and quality axes $(0, 1)$ and $(1, 0)$, real per-configuration acquisition costs, independent direction-local η decay, shared observations, and round-robin controller visits. The evaluation covers eight benchmarks and seeds 42–61. HotpotQA and MathQA use the complete 10×10 assignment matrices. Each run stops when both controllers register numerical stops without an intervening observation, exhausts its external budget, or completes every configuration; a stopped run carries its last recommendation to later plotted targets without additional observations or spend.

For the reported replay metrics, we first remove truly dominated points from each recommendation before computing GD and IGD. These distances therefore describe the recommendation's true-coordinate nondominated subset, whereas Eqs. (3) and (4) define GD and IGD for the full output. Hypervolume is unchanged by this filtering. To assess the full output under the stated convention, GD and IGD must be recomputed without removing those assignments.

G.1 RECOMMENDATIONS ACROSS SEARCH BUDGETS

Figures 8 and 9 in Appendix I show the 10% and 30% recommendations across eight instances. At 10%, the recommendation contains a small set of configurations supported by the available observations. By 30%, more configurations have narrower predictive intervals, and the set traces more of the cost–accuracy frontier. The quality controller can favor interior supported trade-offs through its cost penalty, while the cost controller directly explores the low-cost region. Figures 10 and 11 then compare checkpoint-estimated and exhaustive coordinates for the same seed-42 recommendation sets, separating estimation error from configuration selection.

G.2 ABLATIONS AND INTERPRETATION

Appendix K compares four component-focused families around G0: acquisition-cost coupling, controller directions, continuation, or scheduling. At the 30% checkpoint, G0 has lower mean HV regret, GD, and IGD than every cost-mechanism control. Adding a midpoint or five directions can lower HV regret or GD further, but neither improves all three metrics: G0 retains the lowest mean IGD in that family. Fixed-η continuation without a Gittins stop and sequentially draining one axis both underperform G0 on the three aggregate metrics. We report HV, GD, and IGD together since improvements in dominated area, distance of the retained nondominated subset, and frontier coverage need not coincide.

H RUNTIME RESULTS

Figure 7 compares the total wall-clock runtime of the six methods. qNEHVI has the highest mean runtime on all eight benchmarks, and the gap widens substantially on the three largest SCOPE matrices. Its mean runtime reaches 6.2 hours on Restaurant Valid, 10.3 hours on BIRD Mini Dev, and 22.4 hours on Bing Query Logs, compared with 1.0, 3.3, and 2.8 minutes for CC-GITTINS, respectively. CC-GITTINS finishes within 3.3 minutes on every benchmark. EGE-SH and APE-k also remain below 18 minutes, while the two random baselines finish in approximately 8–14 seconds. These results show that qNEHVI's computational overhead becomes the dominant cost as the configuration matrix grows.

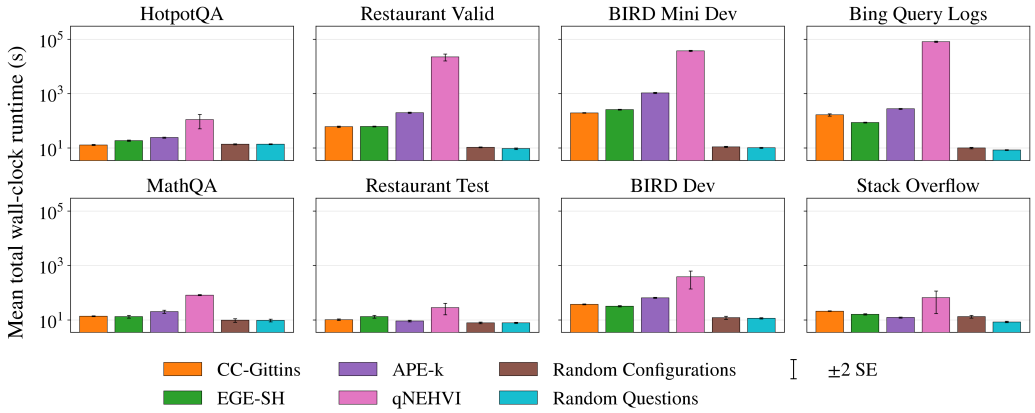

Figure 7: Mean total wall-clock runtime of the six methods across eight benchmark instances. The vertical axis uses a base-10 logarithmic scale, with major ticks marking successive powers of ten. Bars show means and error bars show ± 2 standard errors across seeds.

I CC-GITTINS PARETO PLOTS

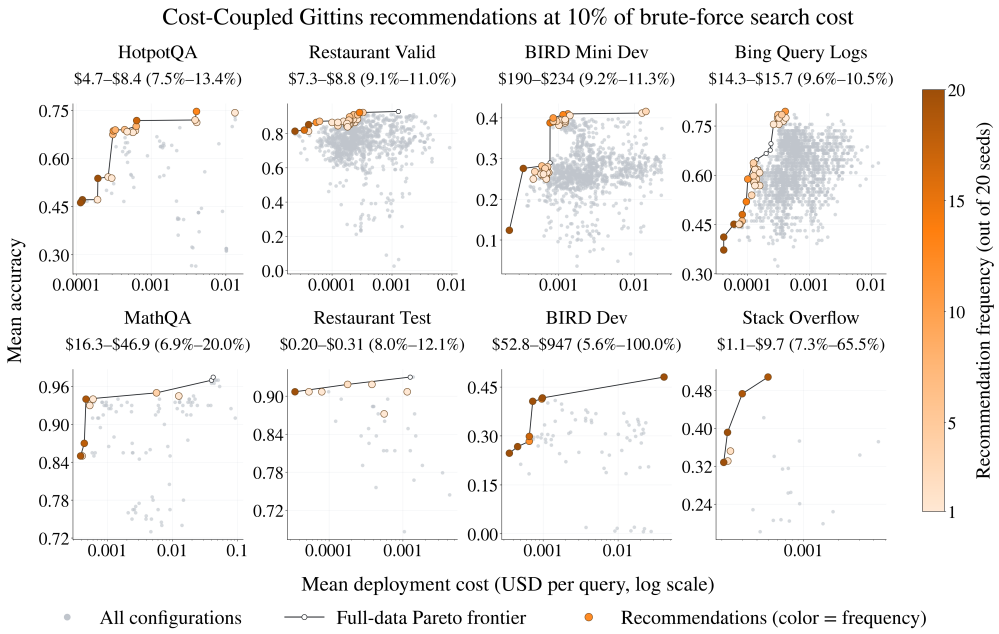

Figure 8: CC-GITTINS recommendations at a 10% search-cost budget across eight benchmark instances and 20 seeds. Gray points show all configurations; the black line shows the full-data Pareto frontier. Orange markers locate recommended configurations at their exhaustive accuracy and deployment cost, with color indicating recommendation frequency across seeds. The line beneath each panel title reports the mean start–end interval over which the checkpoint recommendation remains stable, first as cumulative API cost (USD) and then as a fraction of exhaustive search cost.

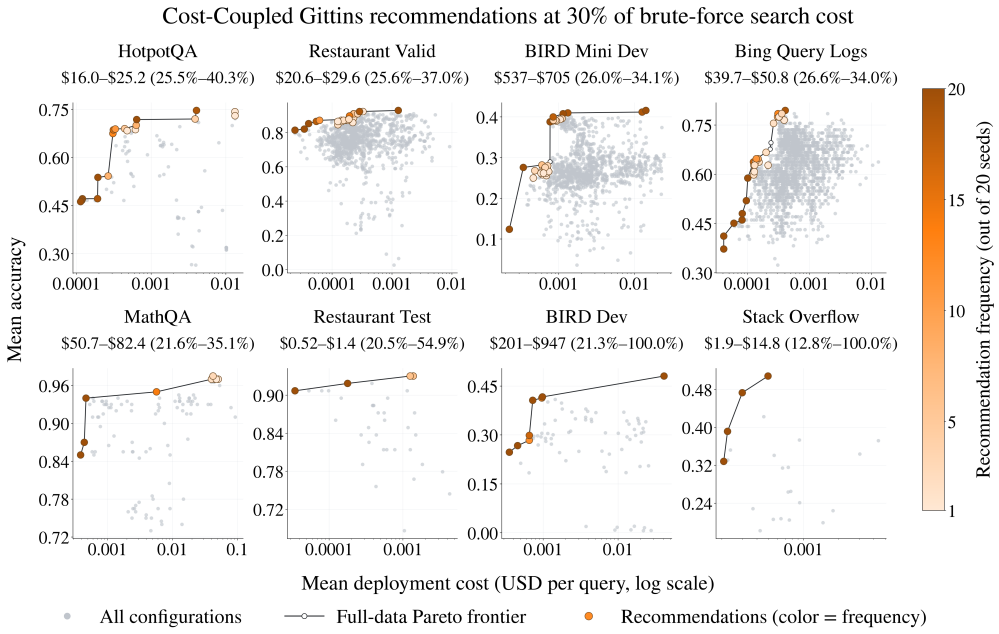

Figure 9: CC-GITTINS recommendations at a 30% search-cost budget, displayed as in Fig. 8. Marker color gives recommendation frequency across 20 seeds; positions use exhaustive accuracy and deployment cost for retrospective comparison.

J FIXED-BUDGET PARETO PLOT PROTOCOL

For every benchmark, seed, and target $b \in \{0.1, 0.3\}$, we select the last recommendation event whose realized cost fraction does not exceed b. If CC-GITTINS has already stopped acquisition, its final recommendation is carried forward with no additional spend. If the warm start is within b but no later event is, we use the warm-start recommendation. A budget below the warm-start spend has no available recommendation. The selected configurations are then placed at exhaustive raw mean accuracy and mean API cost (USD per query). This second step is an offline diagnostic and is not available to the selector.

qNEHVI was prohibitively slow and repeatedly exceeded the available memory on the three largest SCOPE matrices (Restaurant Valid, Bing Query Logs, and BIRD Mini Dev). For these instances, the checkpoint comparisons use the saved qNEHVI recommendations at the 10% and 30% budget targets rather than requiring each run to complete the full cost trajectory. The reported aggregates retain 20, 18, and 19 available seeds for Restaurant Valid, Bing Query Logs, and BIRD Mini Dev, respectively, with uncertainty computed over the available seeds.

For CC-GITTINS, a completed configuration has zero predictive variance for the benchmark mean, so its plotted exhaustive coordinate matches its recommendation estimate. Partially evaluated configurations remain eligible under Eq. (12); their plotted exhaustive coordinates can differ from their online estimates. All methods use the same axis limits within a benchmark.

Figures 10 and 11 isolate estimation error from configuration selection for HotpotQA and Stack Overflow at seed 42. Within each benchmark, the actual and estimated columns contain the same recommended configuration identifiers; only their plotted coordinates change. Thus, displacement between a method's paired panels reflects error in the online estimates available at the checkpoint rather than a change in the returned set.

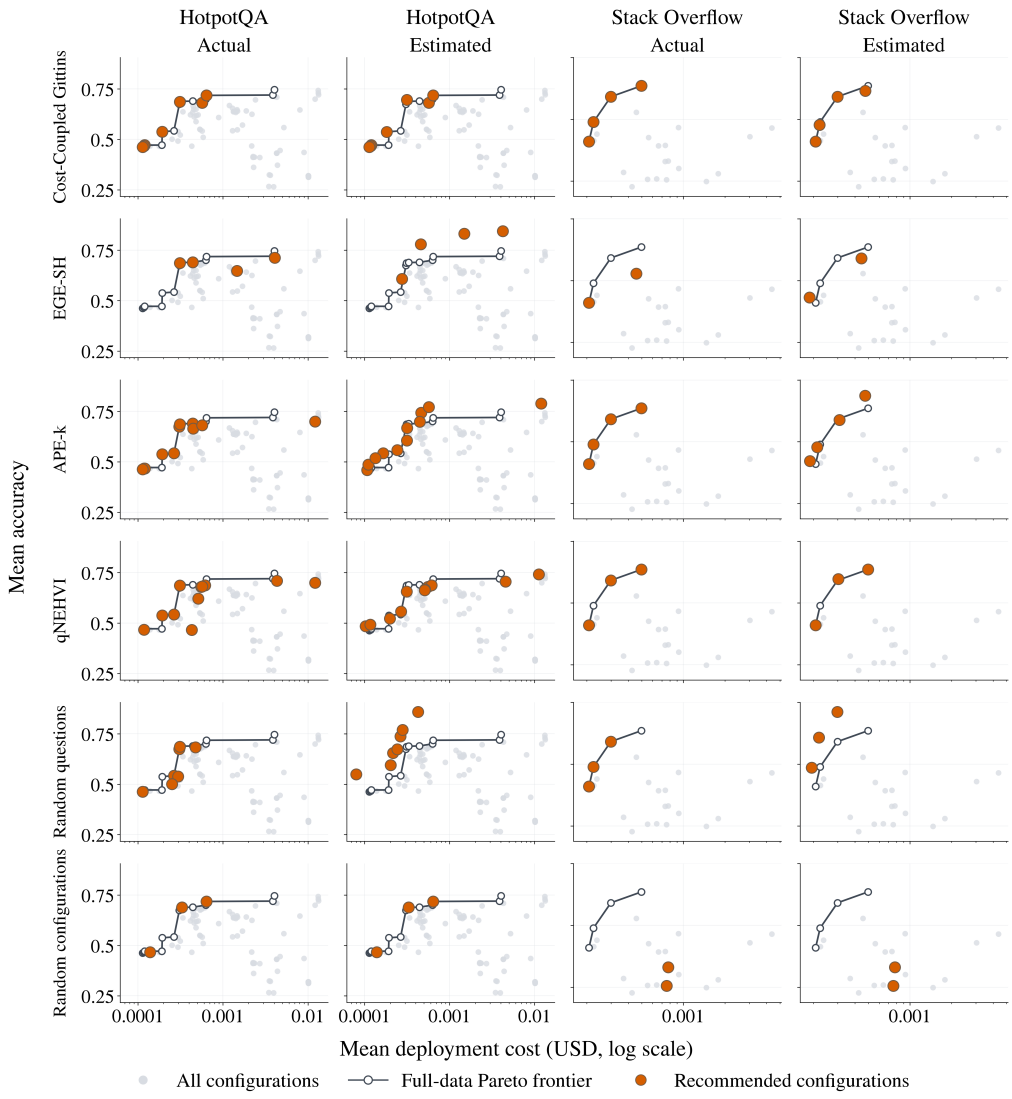

Figure 10: Estimated and actual Pareto recommendations at the 10% search-cost checkpoint for seed 42. Rows show CC-GITTINS and the five baselines; columns show exhaustive (actual) and checkpoint-estimated coordinates for the same recommended configurations on HotpotQA and Stack Overflow. Gray points are all configurations, hollow markers and connecting lines show the exhaustive Pareto frontier, and orange points show the configurations recommended by each method.

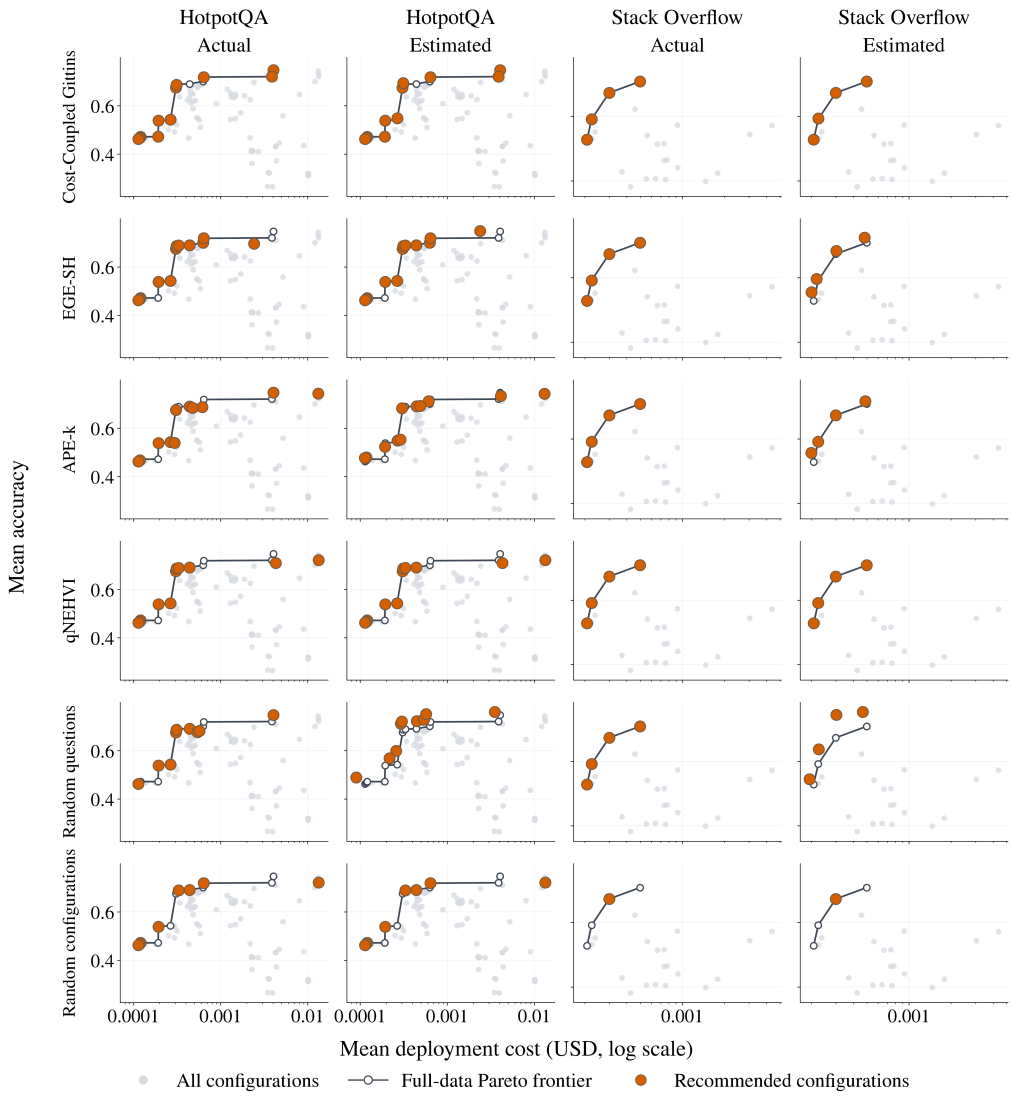

Figure 11: Estimated and actual Pareto recommendations at the 30% search-cost checkpoint for seed 42, using the same layout and visual encoding as Fig. 10. The paired columns separate checkpoint estimation error from the choice of recommended configuration identifiers.

K CONTROLLED CC-GITTINS ABLATIONS

Questions and reference policy. The ablation study asks which parts of CC-GITTINS account for its budget-limited Pareto recommendations: evaluation prices, controller directions, continuation policy, and the order of controller visits. We organize four component-focused families around G0, which uses real configuration-specific acquisition costs, the exact quality and cost axes, independent direction-local η relaxation, and round-robin visits. Table 6 lists all ten configurations and their aggregate results; the following subsections explain each comparison alongside its curves.

Matched comparisons. All configurations use the same eight benchmark matrices, 20 seeds, model ordering, exhaustive truth vectors, recommendation rule, and batch size, giving 1,600 validated benchmark–seed runs. Checkpoints use the last recommendation within each API-cost budget, including warm-up. Unit-cost controls change the prices used inside the acquisition index; actual API charges still determine the evaluation budget. If a run has stopped, its last recommendation is carried forward without further observations or spending. Curves show seed means with ± 2 standard-error bands. Each table row averages 160 matched benchmark–seed snapshots at the 30% checkpoint.

Table 6: Mean ablation metrics at the 30% cost checkpoint. Each row averages 160 matched benchmark–seed snapshots (eight benchmarks and 20 seeds); lower is better. Bold marks the lowest mean in each column across the ten unique configurations.

Family role	Configuration	HV regret	GD	IGD
Reference	G0 CC-Gittins, real cost	0.0016	0.0007	**0.0277**
Cost/direction	G1 quality only, real cost	0.0304	0.0016	0.0627
Cost	G2 quality only, unit cost	0.0387	0.0083	0.0603
Cost	G3 axes, unit cost	0.0086	0.0045	0.0292
Direction	G4 deployment only, real cost	0.0099	0.0016	0.1165
Direction	G5 axes plus midpoint	0.0011	**0.0003**	0.0349
Direction	G6 five directions	**0.0010**	0.0008	0.0321
Continuation	G7 fixed η, no stop	0.0021	0.0015	0.0585
Scheduler	G8 quality then deployment	0.0299	0.0015	0.0611
Scheduler	G9 deployment then quality	0.0023	0.0013	0.0426

How to read the metrics. Lower values are better throughout. HV regret measures missing dominated area; GD measures how far the retained recommendations lie from the true frontier; IGD measures how far true frontier points lie from the retained recommendations. GD and IGD use each recommendation's true-coordinate nondominated subset, so they do not assess every returned assignment. The metrics can therefore favor different configurations: G6 has the lowest aggregate HV regret, G5 the lowest GD, and G0 the lowest IGD. The family-level comparisons below separate these trade-offs from the computational cost of additional controllers.

K.1 Cost mechanism: evaluation prices and the cost controller

Figures 12–14 vary the evaluation prices used by the index and the presence of the cost controller. G0 and G3 both use the exact axes, with real and unit prices respectively; G1 and G2 make the same price comparison with quality-only acquisition. This distinguishes price sensitivity from the benefit of visiting both objectives.

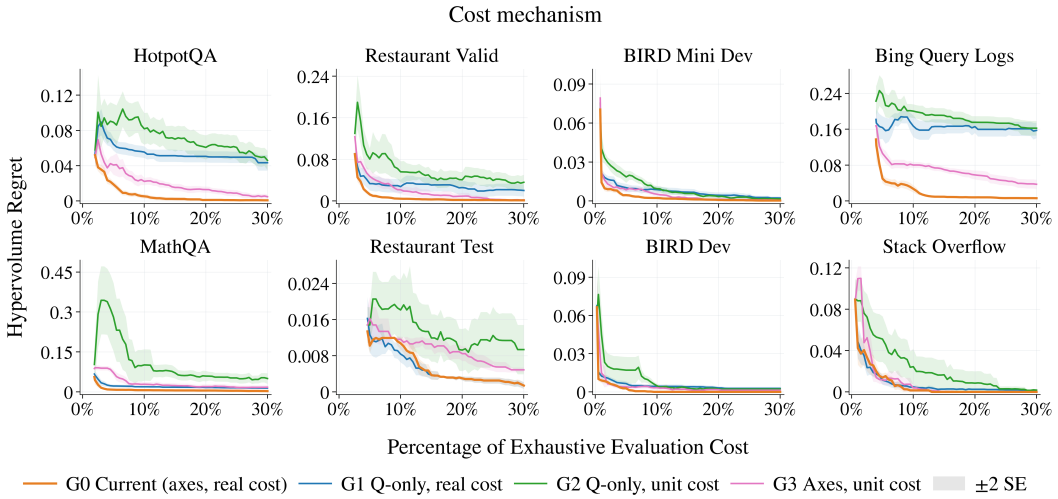

Figure 12: Cost-mechanism ablation: hypervolume regret over 0–30% cost. G0 is the current exact-axis, real-cost protocol; G1–G3 remove the deployment controller, real acquisition costs, or both. Bands show ±2 standard errors over 20 matched seeds.

With both axes fixed, unit prices (G3) raise the 30% mean HV regret by 5.4× and GD by 6.1× relative to G0. Under quality-only acquisition, G2 likewise has 27% higher HV regret and more than 5× higher GD than G1. These contrasts support using configuration-specific evaluation prices in the acquisition rule.

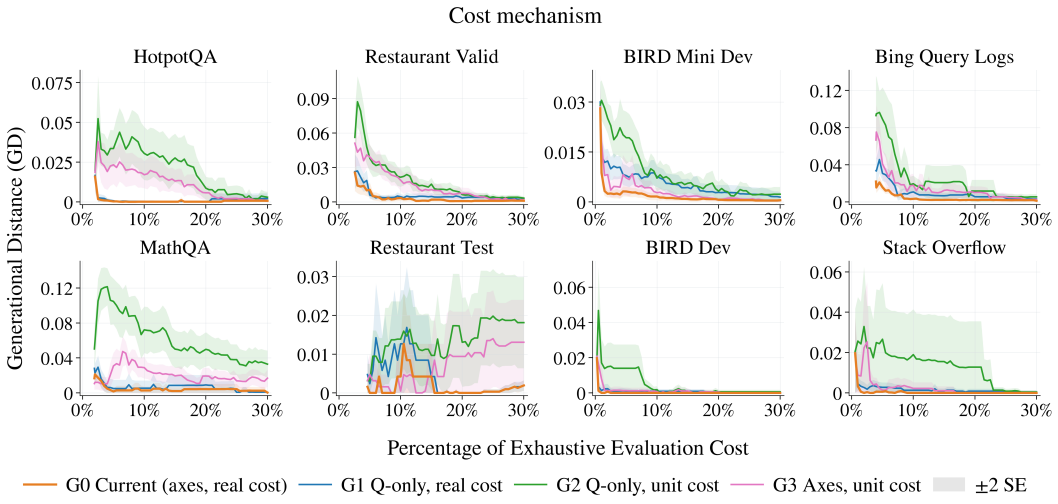

Figure 13: Cost-mechanism ablation: generational distance for the configurations and matched seeds in Fig. 12.

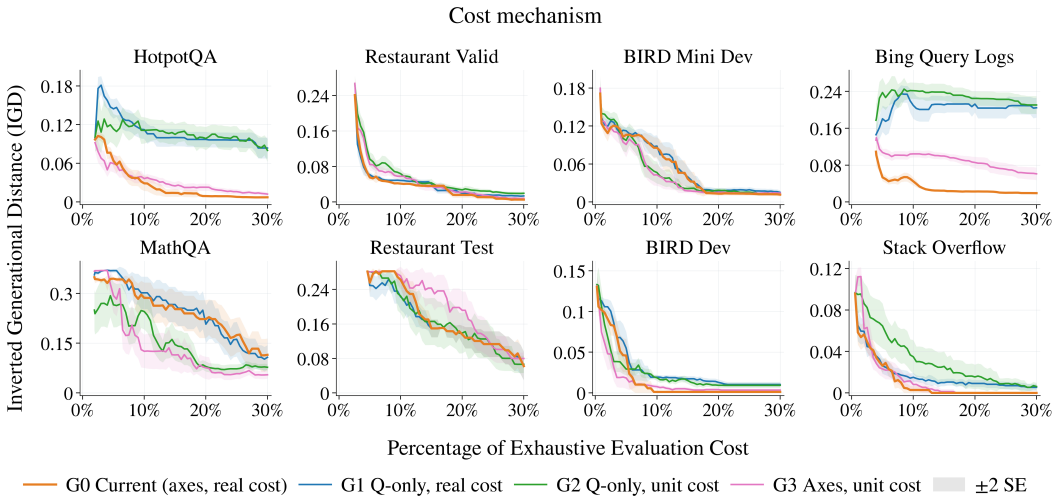

Figure 14: Cost-mechanism ablation: inverted generational distance for the configurations and matched seeds in Fig. 12.

The coverage comparison is more mixed: unit prices increase IGD by 5.5% with both axes, but reduce it by 3.8% under quality-only acquisition. Removing the cost controller while retaining real prices (G1 versus G0) is more damaging: HV regret is 19.2× higher and IGD 2.3× higher. Real evaluation prices and cost-axis visits thus make distinct contributions, without improving every metric in every contrast.

K.2 Directions: frontier coverage and computation

Figures 15–17 retain real acquisition costs and compare quality only (G1), cost only (G4), exact axes (G0), axes plus a midpoint (G5), and five evenly spaced directions (G6). The one-axis controls probe the two objectives' complementary roles; the denser sets test whether additional directions improve frontier approximation.

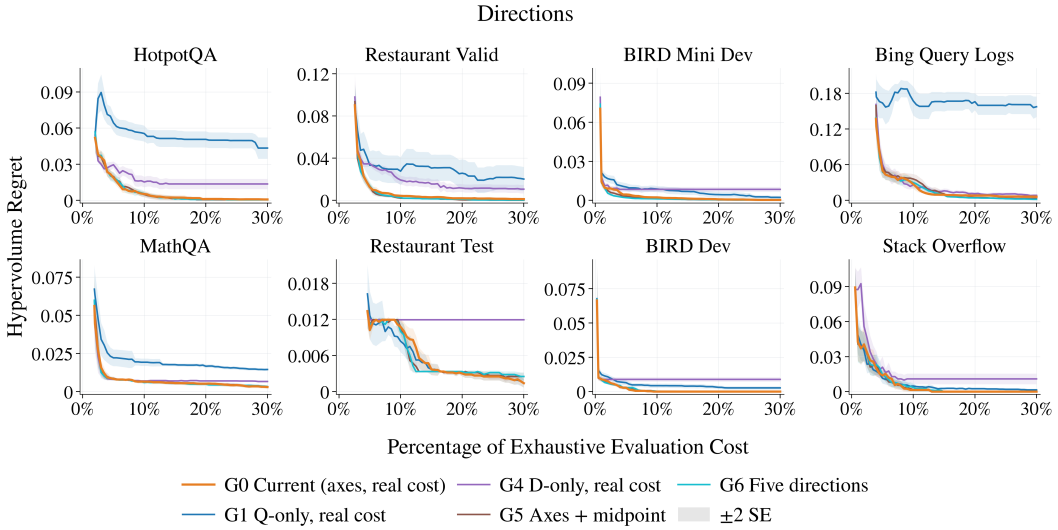

Figure 15: Direction ablation: hypervolume regret for quality-only (G1), deployment-only (G4), exact axes (G0), axes plus midpoint (G5), and five directions (G6), all with real acquisition costs. Bands show ±2 standard errors over 20 matched seeds.

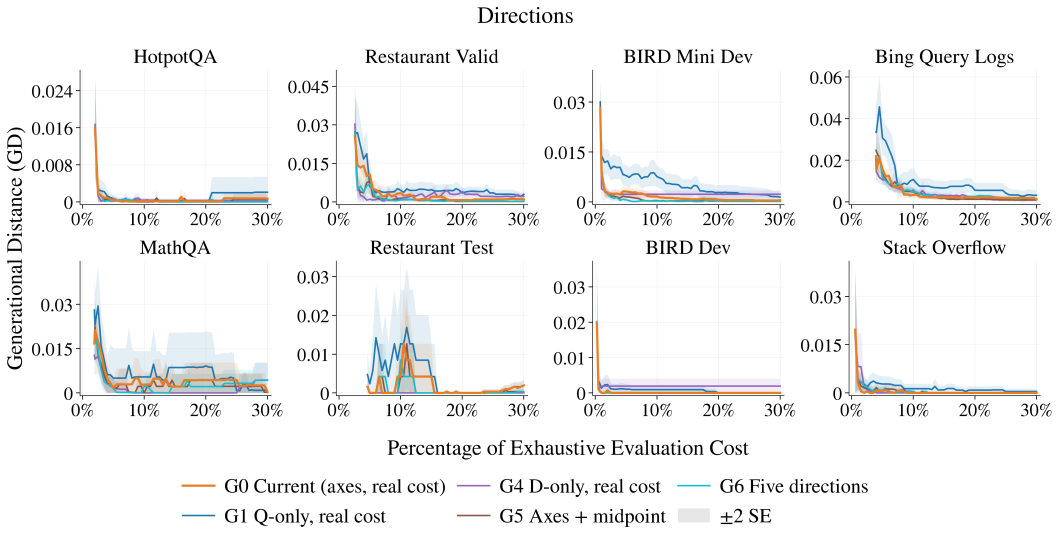

Figure 16: Direction ablation: generational distance for the configurations and matched seeds in Fig. 15.

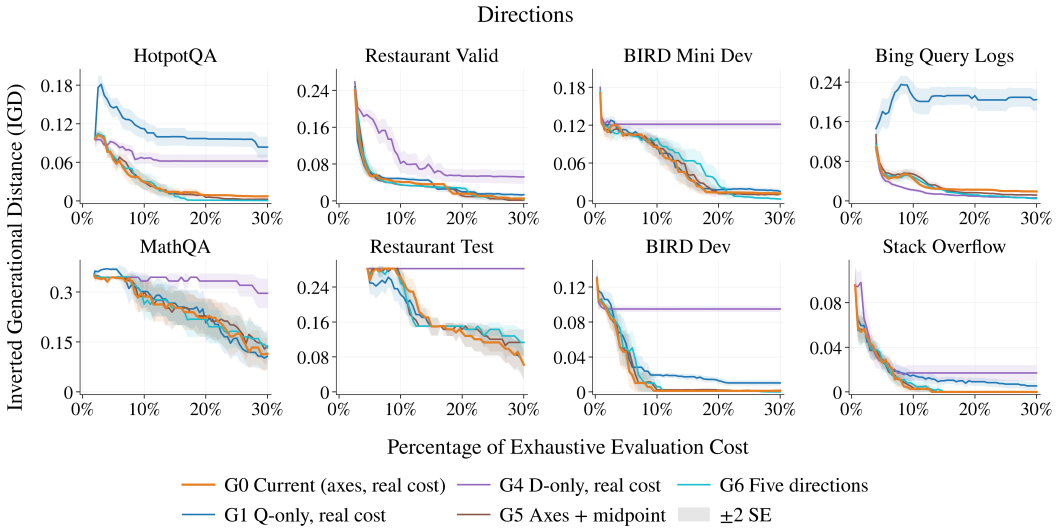

Figure 17: Direction ablation: inverted generational distance for the configurations and matched seeds in Fig. 15.

Extra directions trade improvements among metrics. At 30%, G6 lowers mean HV regret by 36% relative to G0, but increases GD by 14% and IGD by 16%. G5 lowers HV regret by 28% and GD by 58%, while increasing IGD by 26%. G0 retains the lowest mean IGD in this family. The one-axis controls expose the limits of focusing on a single objective: G1 has high HV regret, while G4 has the family's largest mean IGD.

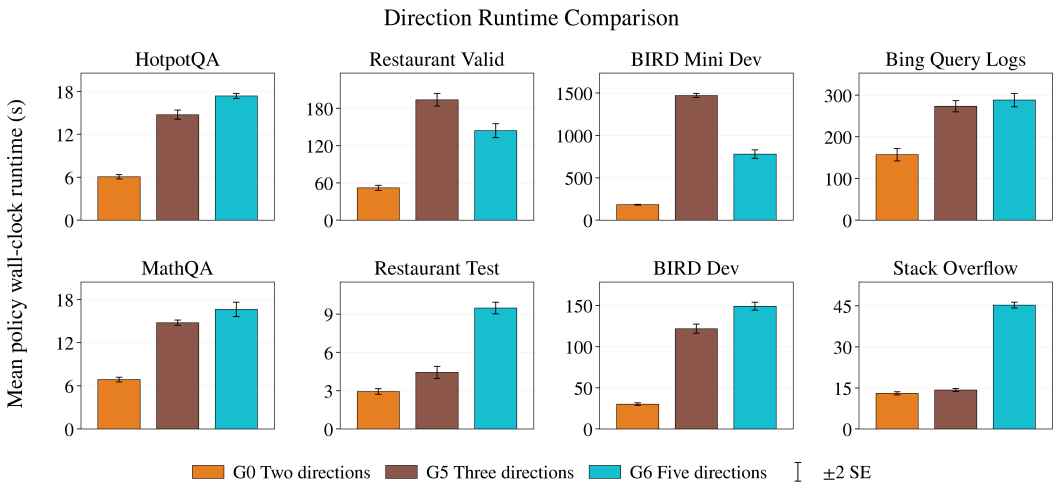

Figure 18: Direction ablation: mean policy wall-clock runtime for exact axes (G0), axes plus the midpoint (G5), and five directions (G6). Bars and error bars show the mean and ±2 standard errors over 20 matched seeds. The policy timer isolates CC-Gittins computation and excludes data loading, result serialization, and Slurm queue time.

Figure 18 adds the computational comparison. G0 has the lowest mean policy runtime on every benchmark. Averaging per-benchmark runtime ratios, G5 and G6 require 3.08× and 3.22× G0's runtime, respectively. The exact axes therefore combine the lowest aggregate IGD with lower computation among these variants; additional directions can still be preferable when HV regret or GD is the priority.

K.3 Continuation: relaxation and stopping

Figures 19–21 compare G0's independent direction-local relaxation and configured stopping rule with G7, which fixes η and continues acquiring evaluations without a Gittins stop. This tests the combined continuation policy: G7 changes both the price schedule and stopping behavior, so the comparison does not isolate the effect of relaxation alone.

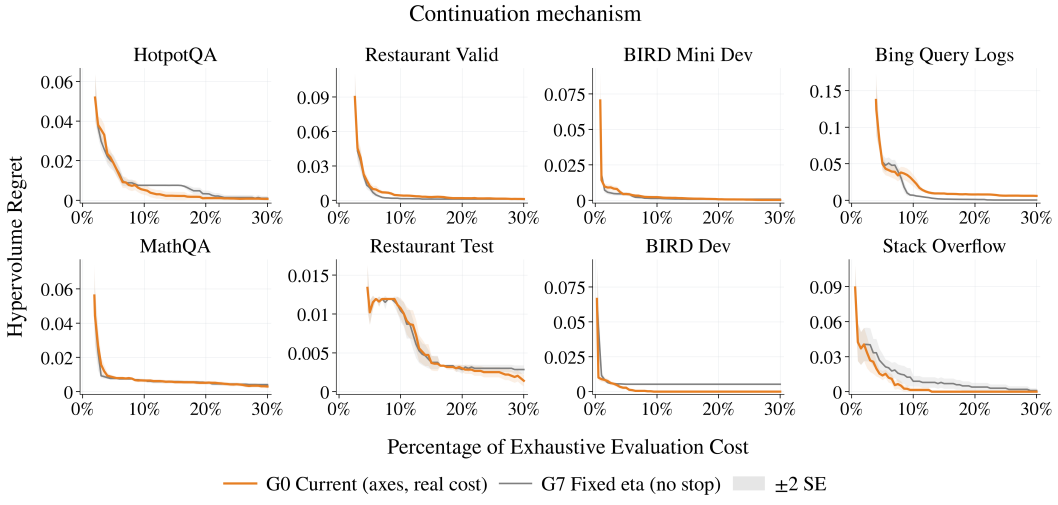

Figure 19: Continuation ablation: hypervolume regret for G0's independent direction-local relaxation and G7's fixed-η continuation without a Gittins stop. Bands show ±2 standard errors over 20 matched seeds.

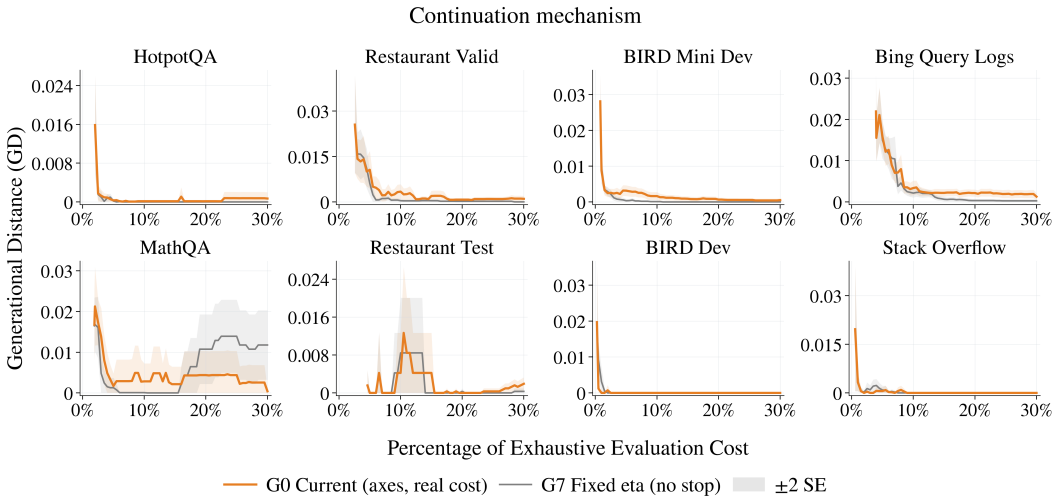

Figure 20: Continuation ablation: generational distance for the configurations and matched seeds in Fig. 19.

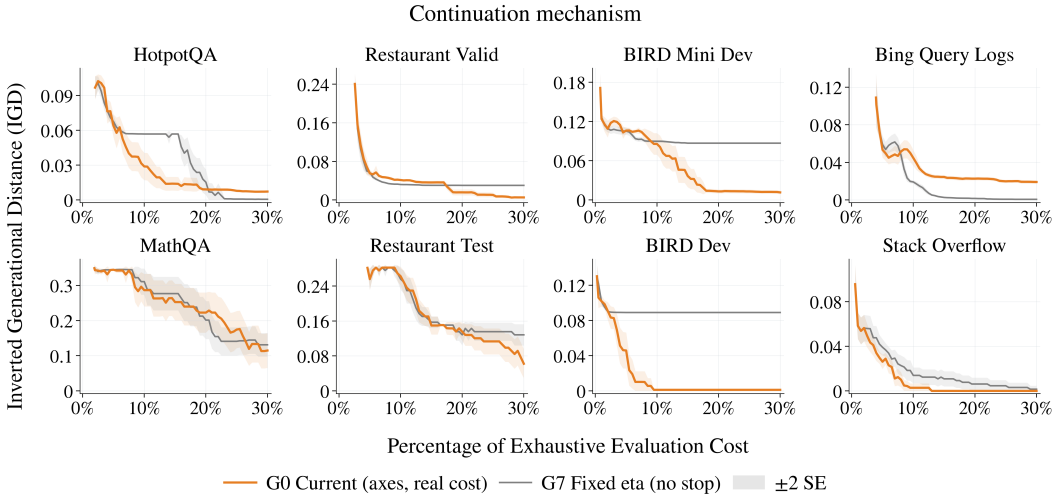

Figure 21: Continuation ablation: inverted generational distance for the configurations and matched seeds in Fig. 19.

Fixed-price continuation can help individual instances, notably Bing Query Logs, but G7's 30% aggregate metrics are worse: HV regret is 31% higher than G0, and GD and IGD are each approximately 2.1× G0. These results favor the tested adaptive relaxation-and-stopping policy overall. Since stopped runs carry their last recommendation to later checkpoints, a higher available budget on these curves need not represent additional evaluations or spending.

K.4 SCHEDULER: ALTERNATING VERSUS SEQUENTIAL VISITS

Figures 22–24 compare G0's alternating visits with quality-then-cost (G8) and cost-then-quality (G9). The sequential variants drain one controller before visiting the other. The comparison tests whether giving both controllers opportunities early in the search matters when the available API budget is limited.

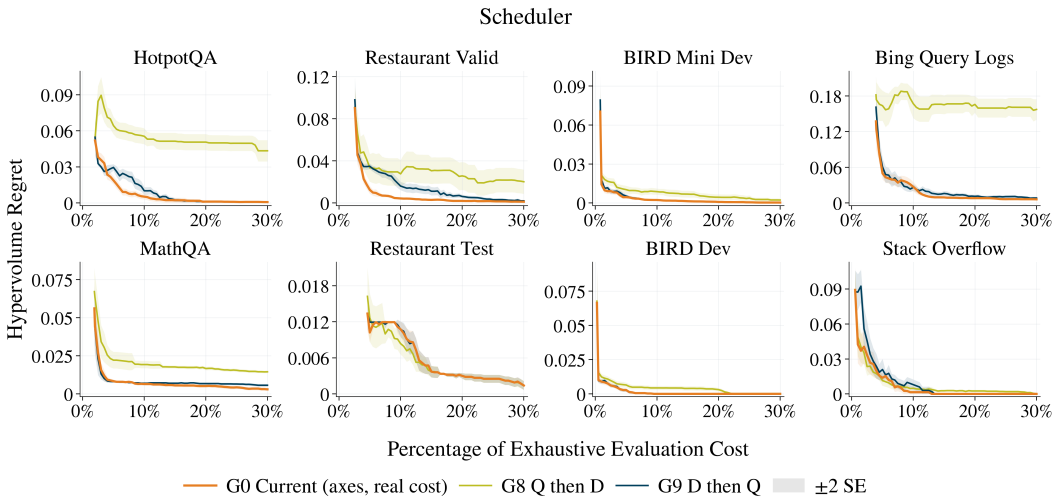

Figure 22: Scheduler ablation: hypervolume regret for round-robin G0, quality-then-deployment G8, and deployment-then-quality G9. Bands show ±2 standard errors over 20 matched seeds.

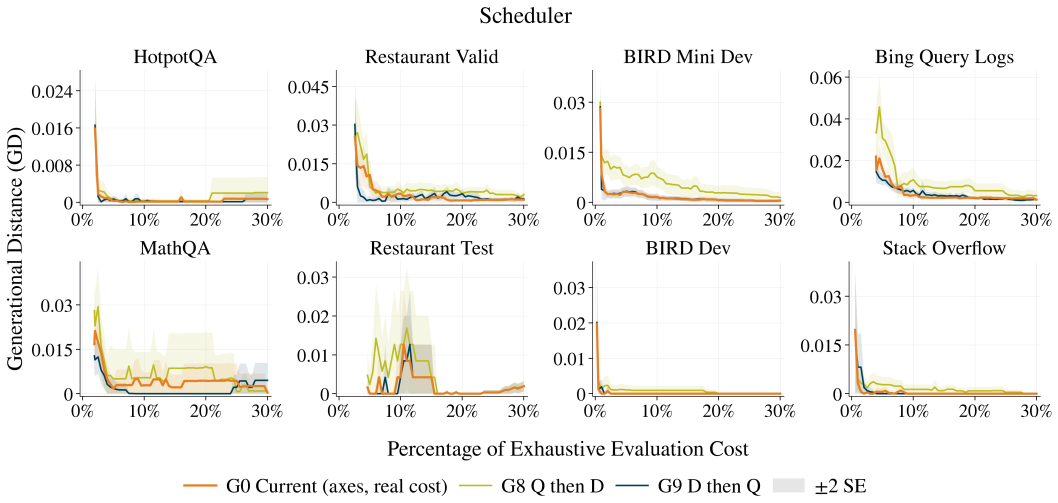

Figure 23: Scheduler ablation: generational distance for the configurations and matched seeds in Fig. 22.

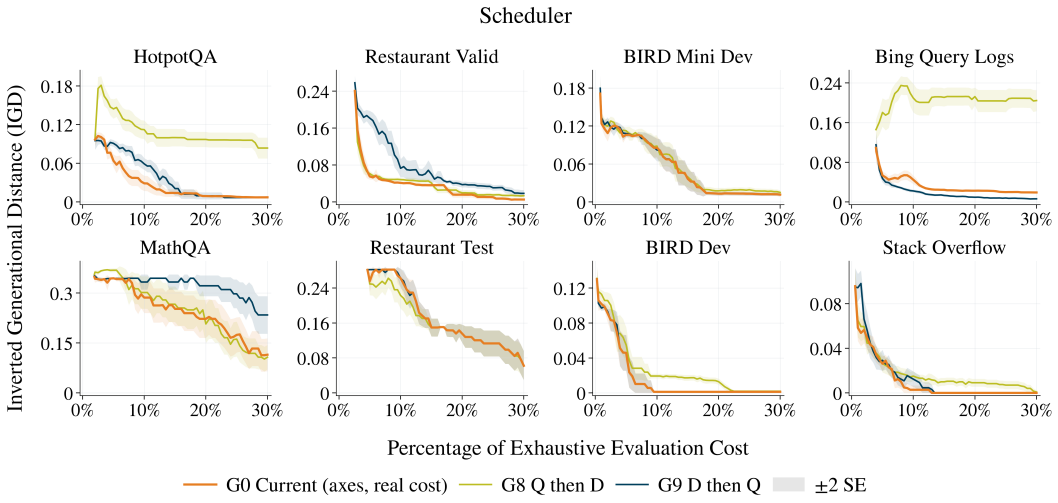

Figure 24: Scheduler ablation: inverted generational distance for the configurations and matched seeds in Fig. 22.

At 30%, quality-first G8 has $18.9\times$ G0's mean HV regret and approximately 2.1–$2.2\times$ its distance metrics. Cost-first G9 performs better than G8, but remains worse than G0 by 47% in HV regret, 79% in GD, and 54% in IGD. This supports interleaving the two controllers within a limited budget. Round-robin balances decision opportunities; it does not imply equal numbers of purchased batches or equal API expenditure by the controllers.

What the study establishes. Across these matched comparisons, real evaluation prices, access to both exact axes, and alternating visits improve G0's aggregate results relative to the corresponding controls. The continuation comparison supports the combined relaxation-and-stopping policy, while the direction comparison reveals a choice between distance, dominated area, and computation. These findings concern the tested matrices and budget range. They motivate the selected configuration without establishing universal superiority of two directions or guarantees for the complete returned recommendation set.