How Can We Recommend Models Cost-Efficiently?

A Bayesian Approach to LLM Evaluation and Agent Workflow Optimization

Qian Xie[1,4]

Based on joint work with

Yueli He[2], Wenyue Hua[3], Sripad Karne[1], Armaan Agrawal[1], Nikos Pagonas[1], Eugene Wu[1], Kostis Kaffes[1], Tianyi Peng[1], Nairen Cao[2]

[1]Columbia University [2]New York University [3]Microsoft [4]Cornell University

Columbia Business School DRO Brown Bag

Motivation: Model recommendation depends on the task and the budget

Single-objective

Resolve GitHub issues

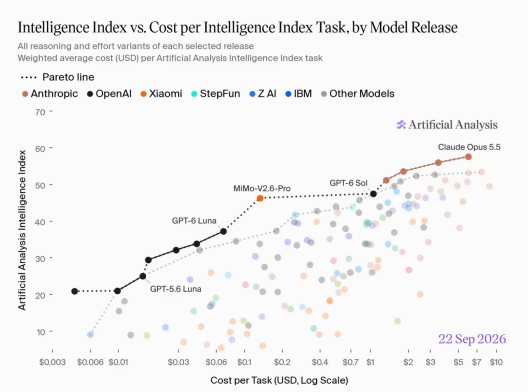
SWE-bench Verified

Model configuration	Resolved
Claude 4.5 Opus (high)	**76.8%**
Gemini 3 Flash (high)	75.8%
MiniMax M2.5 (high)	75.8%
Claude 4.6 Opus	75.6%

Find a high-performing configuration for the task we care about.

Selected entries. mini-SWE-agent v2.0.0.
Runs: Feb 17, 2026. Source: SWE-bench Verified.

Multi-objective



Choose a quality–cost trade-off from the frontier.

Original animation: Artificial Analysis, Sep 23, 2026.

How can we learn these recommendations without evaluating everything?

Part I: Single-objective

Efficient Cost-Aware LLM Evaluation via Bayesian Bandit Gittins Indices

Preliminary version accepted as **Spotlight** at

ICML 2026 Workshop on Decision-Making from Offline Datasets to Online Adaptation: Black-Box Optimization to Reinforcement Learning.

LLM evaluation searches for a good task-specific configuration

Goal: find the best configuration at far lower evaluation cost.

Model	Q1	Q2	Q3	...	avg performance
ChatGPT GPT-4o	✓	?	?	...	?
Claude 3.5 Sonnet	?	✓	?	...	?
Deepseek	✓	? Y_{ij}	?	...	?
Gemini 1.5 Pro	✗	?	✓	...	?
Llama 3.1-70B	?	✗	?	...	?
Mistral Large	?	?	✗	...	?
...	...	...	...	...	...

No need to evaluate every configuration on every question.

Ours: simple regret (↓)

Best mean score − recommended mean score (full benchmark).

Problem setup

Configuration: model + prompt (+ temperature + decoding).

Target: highest mean performance.

Evaluation cost

Token/API price, latency, or grading.

Evaluating the whole matrix is expensive.

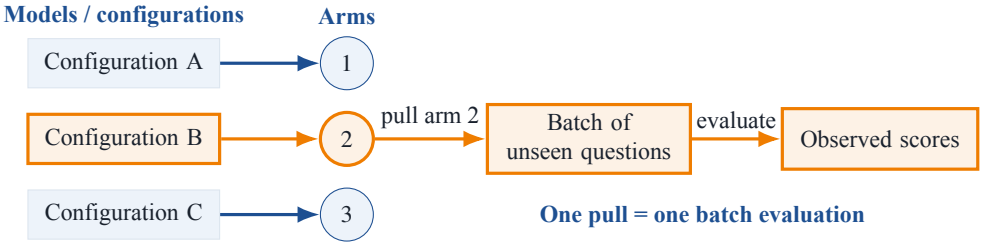
Existing work

BanditEval (UCB-E), **SySRs**, and **PromptEval-BAI**.

They are cost-unaware and frequentist-based.

BanditEval: bandit formulation and UCB-E (Zhou et al., 2025)

Goal: best-arm identification under a fixed evaluation budget.



UCB-E allocation (frequentist, cost-unaware)

Sample mean + confidence bonus

Evaluate the arm with the largest score.

The bonus shrinks with more observations of that arm.

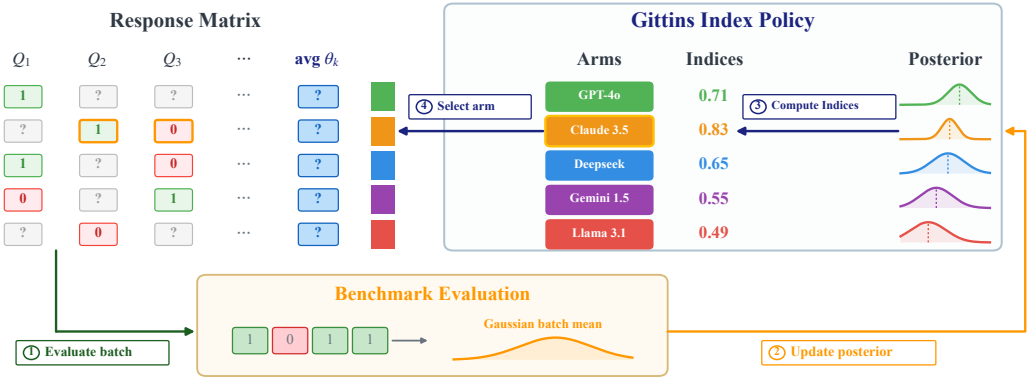
Final recommendation

Highest sample mean

Recommend when the evaluation budget is exhausted.

GittinsEval: our Bayesian evaluation framework

Posterior beliefs and evaluation costs guide the next batch.

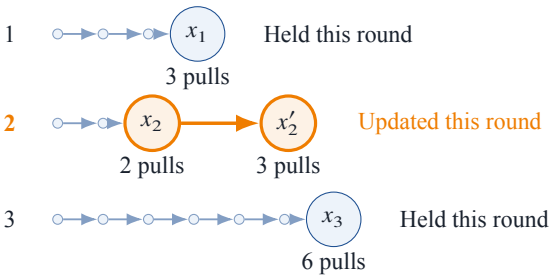


Each Gittins index weighs further evaluation against its cost.

Background: Gittins policy for Markov chain selection (Bayesian bandits)

In our model, posterior means evolve as independent Markov chains (Gaussian random walks).

Only the selected chain (arm) advances this round



State: posterior mean and local pull count

Gittins index

A score for each arm, using posterior belief and evaluation cost.

Advance

Evaluate the unfinished arm with the highest score.

Stop

When a completed arm has the highest score.

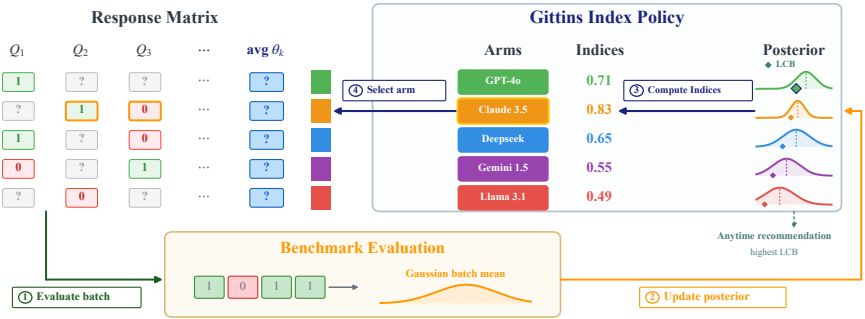
The Gittins policy is optimal under required completion in this model.
Only fully evaluated configurations are eligible for recommendation.

$$\max_{\pi} \ \mathbb{E}_{\pi}[\text{terminal reward} - \text{total evaluation cost}]$$

Our adaptation: anytime recommendation before completion

Partially evaluated configurations remain eligible for recommendation.

A high posterior mean can come with substantial uncertainty when few questions are evaluated.



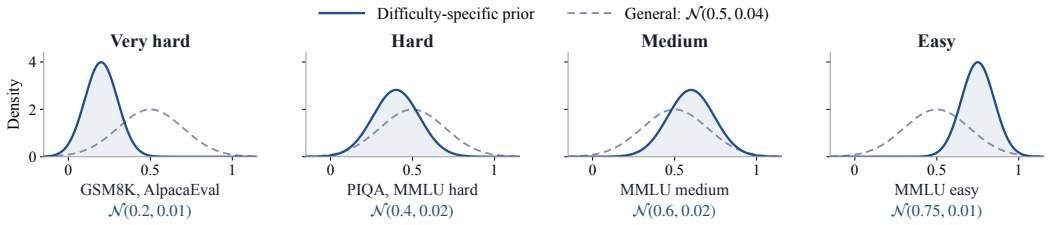
Recommend the configuration with the highest LCB score: $\mathrm{LCB}_{k,t} = M_{k,t} - \sqrt{V_{k,t}}$

$M_{k,t}, V_{k,t}$: posterior mean and variance of the **full-benchmark mean**.

Design choice: prior selection

Each variant shares one prior across all configurations in a task.

Shared Gaussian priors



Prior beliefs about mean performance; $\mathcal{N}(\mu_0, v_0)$ uses variance v_0.

GittinsEval-G

General prior: $\mathcal{N}(0.5, 0.04)$.
The same prior for every configuration.

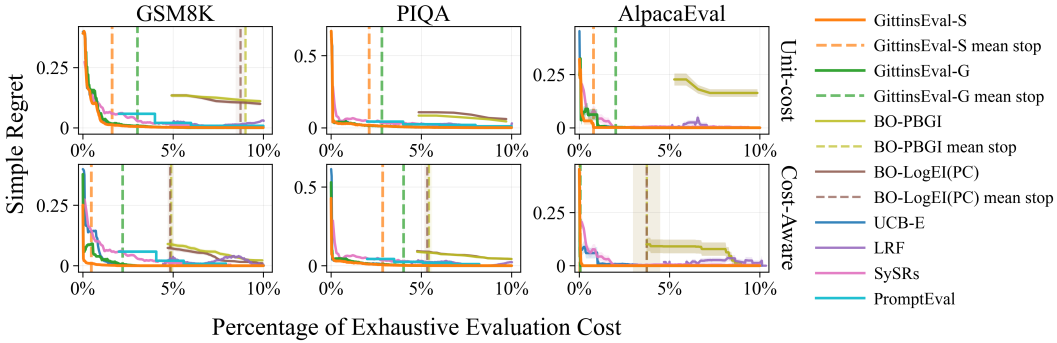
GittinsEval-S

One difficulty bucket shared by all arms.
AlpacaEval also uses the very-hard bucket.

Experiments select difficulty buckets from aggregate offline performance.
Prospective applications can use related tasks or pilot data.

Results on many-example benchmarks: BanditEval

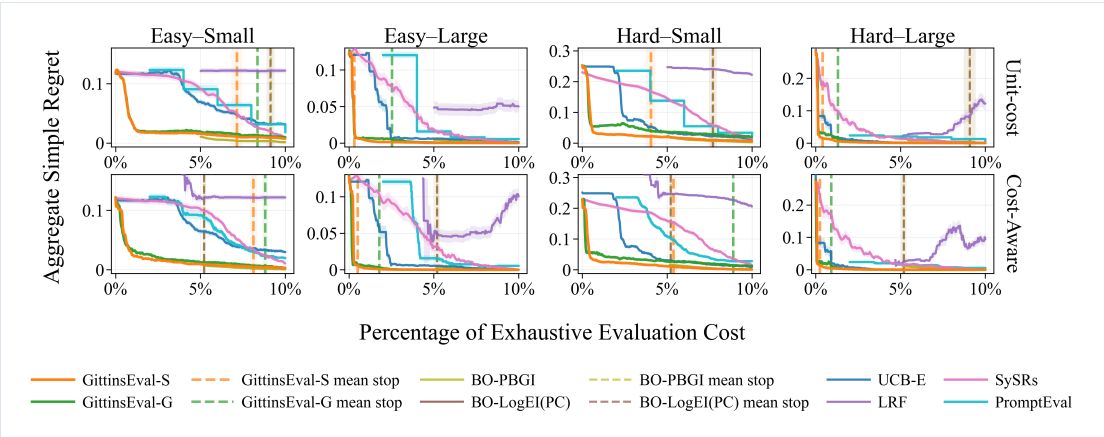
GittinsEval-S reaches low regret much earlier than configuration-level Bayesian optimization.



Scale: 805–1,000 examples per candidate, with 103–152 candidates per benchmark.
Bayesian optimization baselines: BO-PBGI, BO-LogEI(PC).

Results on many-candidate benchmarks: DOVE (57 MMLU datasets)

GittinsEval-S reaches low regret much earlier than frequentist approaches.



Scale: 1,500 candidates per subject (15 models × 100 templates).
Frequentist baselines: UCB-E, LRF, SySRs, and PromptEval-BAI.

Part II: Multi-objective

Which Model for Which Role?
Cost-Efficient Pareto Recommendation for LLM Agentic Workflows

Preliminary version accepted at

NeurIPS 2026 Workshop on Resource-Aware Agentic AI.

Model assignments for agentic workflows

Each candidate assigns one model to every role in a fixed workflow.

Example: HotpotQA workflow



Ten model choices per role give **10 × 10 = 100 candidate assignments**.

Evaluation data

AgentOpt (our evaluation harness)
HotpotQA and MathQA; two-role workflows.

SCOPE (prior work; Huang et al., 2026)
Constrained optimization:
min cost s.t. quality ≥ threshold.
Released data: six instances with 3–5 roles.

Our goal: Pareto optimization

Two objectives: quality ↑, cost ↓
Tested on AgentOpt and SCOPE data.

Three objectives: add latency ↓
Tested on AgentOpt data only.

Deployment cost
Mean API cost per query for a chosen assignment.

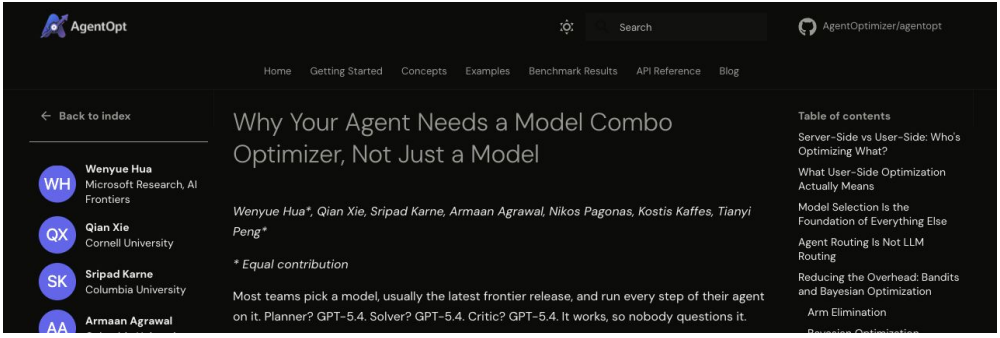
Search cost
Total API spend to evaluate candidate assignments.

Costly workflows also tend to cost more to evaluate (cost coupling).

Workflow Evaluation Harness in AgentOpt

Our open-source package supports client-side optimization and records quality, API cost, and latency.
CC-Gittins can be integrated as a Pareto search algorithm.

GitHub: github.com/AgentOptimizer/agentopt **Project blog**

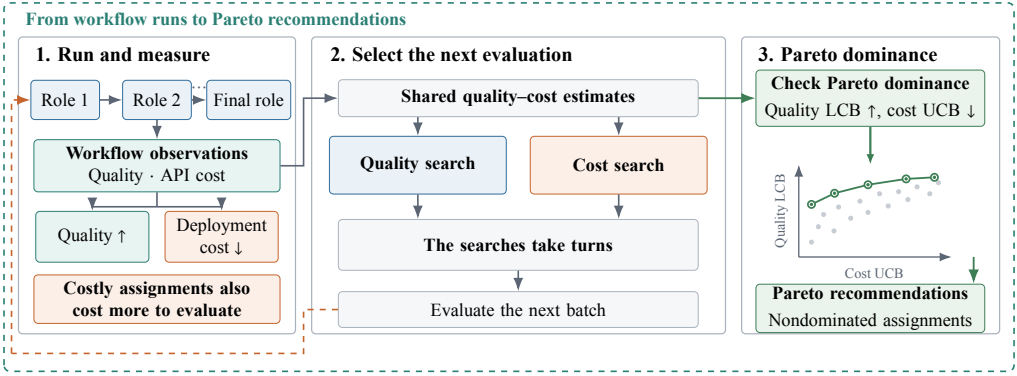


Technical report Hua et al. (2026) arXiv:2604.06296

AgentOpt v0.1 Technical Report: Client-Side Optimization for LLM-Based Agent

Cost-Coupled Gittins: our Pareto recommendation algorithm

Quality and cost searches share every workflow observation.



Direction choice

Cost coupling and penalty relaxation let us explore trade-offs with just two searches.

Batch selection

Controllers alternate, choosing the unfinished assignment with their highest Gittins index.

Independent continuation

Each controller relaxes its own cost penalty after a local stopping signal.

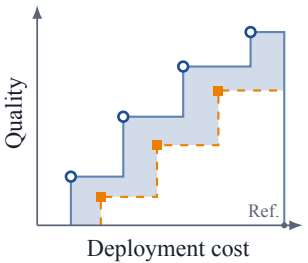
How we evaluate Pareto recommendations

Compare with the exhaustive Pareto frontier. **Lower is better.**



HV regret

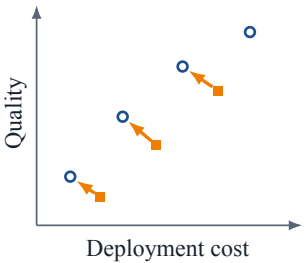
Hypervolume regret



Missed trade-off volume

GD

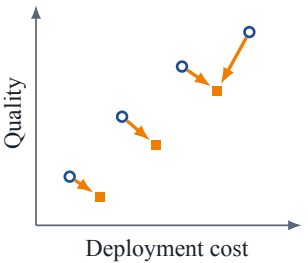
Generational distance



Recommendation proximity

IGD

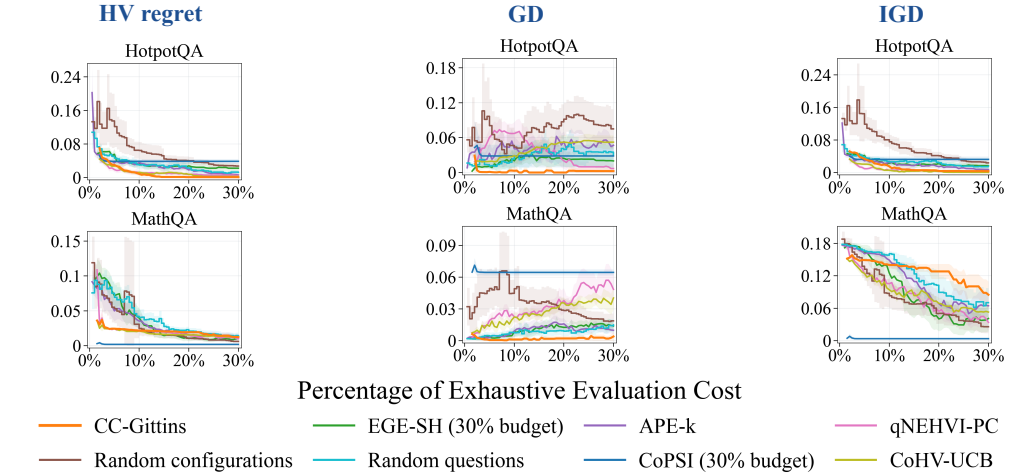
Inverted generational distance



Frontier coverage

Our goal: low HV regret, GD, and IGD.

Quality–cost results: HV regret, GD, and IGD



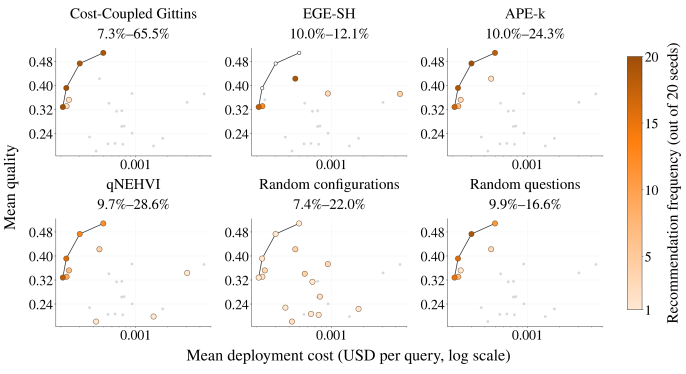
Means with ±2 SE over 20 matched seeds. Lower is better for all metrics.

At 30% cost, CC-Gittins has low HV regret and near-zero GD.
Frontier coverage is weaker on MathQA.

Useful frontier recommendations

Stack Overflow (SCOPE): CC-Gittins vs. five baselines

At 10% API-cost budget



Precision and recall

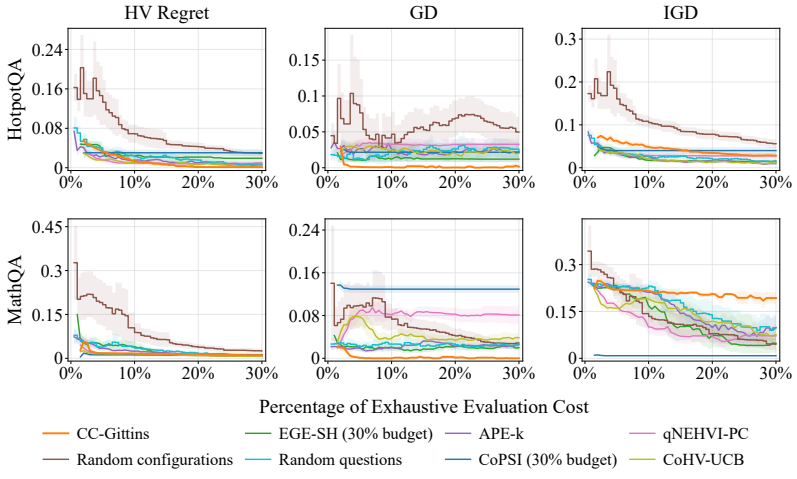
CC-Gittins, 30% API-cost budget

Dataset	Precision	Recall
HotpotQA	0.985	0.743
Restaurant Valid	0.881	0.795
BIRD Mini Dev	0.822	0.795
Bing Query Logs	0.881	0.663
MathQA	0.958	0.658
Restaurant Test	0.900	0.767
BIRD Dev	1.000	0.888
Stack Overflow	1.000	1.000

Precision: avoiding false positives

Recall: avoiding false negatives

Quality, API cost, and latency: three-objective results



Means with ±2 SE over 20 matched seeds. Lower is better for all metrics.

At 30% cost, CC-Gittins generally has lower HV regret and GD than the baselines.
IGD is weaker, especially on MathQA.

Bayesian evaluation for model recommendation

Part I

Single-objective

LLM evaluation

One configuration

Efficient Cost-Aware LLM Evaluation via Bayesian Bandit Gittins Indices



Preprint



Code

Part II

Multi-objective

Agent workflow optimization

Pareto set

Which Model for Which Role? Cost-Efficient Pareto Recommendation for LLM Agentic Workflows



Code



AgentOpt report