IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 3, November 2026
Impact Factor: 8.2

LLM Cost Autopilot: Intelligent AI Model Routing for Cost Optimization – A Survey of Cost-Aware LLM Routing Techniques and Proposed Architecture

Bhavya R. Patel[1], Rupal M. Meshram[2], Isha M. Saxena[3], Chaitali P. Jawale[4], Prof. Shubangi Deshmukh[5]
Department of Artificial Intelligence & Data Science[1-5]
Terna Engineering College, Navi Mumbai, India

Abstract: *Large Language Models (LLMs) provide strong capabilities across reasoning, coding, knowledge retrieval, and generation, but API usage becomes expensive when every request is routed to a high-cost model. The central problem is choosing an adequate model and inference strategy for each request while balancing quality, cost, latency, and operational constraints. This paper surveys twelve studies spanning unified routing infrastructure, benchmark design, cost-aware selection, preference-based routing, resource-aware serving, output-budget control, batching, cascading, quality-aware routing, adaptive test-time compute, and online routing with user feedback. The literature indicates a shift from static model choice toward multi-objective and adaptive routing. A practical gap remains for a lightweight, transparent, configurable application-level cost controller for small multi-provider applications. To address this, the paper proposes LLM Cost Autopilot, combining query analysis, explicit cost/capability rules, response caching, multi-provider model invocation, and cost/performance observability with an evaluation framework. No experimental results for the proposed system are claimed; it is presented as an implementation-oriented baseline for future extensions. Index Terms—LLM routing, cost optimization, cost-aware routing, model selection, response caching, token usage, latency, multi-model serving, adaptive compute, user satisfaction, AI cost monitoring.*

Keywords: LLM routing, cost optimization, cost-aware routing, model selection, response caching, token usage, latency, multi-model serving, adaptive compute, user satisfaction, AI cost monitoring.

I. INTRODUCTION

Hosted LLM APIs provide access to models with different capabilities, token prices, latency, context handling, and reliability. Sending every request to the most capable model is operationally simple but can create avoidable expenditure when a lower-cost model can adequately satisfy the task. LLM routing introduces a decision layer between an application and a pool of candidate models. Recent work evaluates routing under performance and performance–cost settings [1], formalizes it as a sequential decision process [2], and jointly considers routing with resource allocation under latency service-level objectives [3].

Other approaches estimate cost and performance [4], learn from human preferences [5], control output budgets [6], and optimize batching [7].

Cost reduction also includes prompt adaptation, model approximation, and cascading [8]. Quality-aware routing [9], adaptive test-time compute [10], large-scale router evaluation [11], and online optimization using limited user feedback [12] further broaden the field. This paper contributes: (1) a review of twelve studies using common criteria; (2) a concise identification of the application-level implementation gap; and (3) a proposed architecture, workflow, and evaluation

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 3, November 2026
Impact Factor: 8.2

framework for LLM Cost Autopilot. The proposed system is a design baseline; it has not yet been experimentally evaluated.

II. RELATED WORK / LITERATURE REVIEW

A. Unified Benchmarking and Routing Infrastructure

LLMRouterBench [1] provides a large-scale benchmark and unified framework with more than 400K instances from 21 datasets and 33 models, integrating ten routing baselines for performance-oriented and performance–cost evaluation. Its authors report that many methods perform similarly under unified evaluation, some commercial routers do not reliably outperform a simple baseline, and a gap to the oracle remains. This highlights the need to evaluate routing quality, not savings alone. LLMRouter [2] models routing as a sequential decision process using context encoders, model encoders, scoring functions, decision rules, and learning signals. It covers single-turn, multi- turn, and personalized routing, and introduces xRouteBench and open-source infrastructure with more than sixteen representative routers. It offers a path from rule-based to learned routing, though with greater implementation complexity.

B. Resource- and Latency-Aware Routing

RouterWise [3] jointly optimizes routing and resource allocation. Given a GPU cluster, model pool, and latency service-level objective (SLO), it allocates resources and selects a routing policy to maximize output quality under the SLO, using deployment- specific latency profiling. This motivates response-time monitoring in the proposed system. However, LLM Cost Autopilot operates at the application/API layer and does not perform cluster-level GPU scheduling.

C. Cost-Aware and Preference-Based Routing

CARROT [4] estimates candidate-model performance and cost, then selects an operating point using a risk function. Its SPROUT prompt collection supports cost–performance analysis and reinforces that the cheapest model is not necessarily appropriate: expected quality must be considered with cost. RouteLLM [5] trains routers on human preference data to choose dynamically between stronger and weaker LLMs. It reports reduced use of the more expensive model while preserving evaluated response quality. Historical user feedback and quality judgments are a potential future addition to the proposed rule-based system.

D. Output-Budget and Batch-Aware Routing

R2-Router [6] jointly selects an LLM and output-length budget, enforcing the budget through length-constrained instructions. Its R2-Bench captures model behavior across output budgets, and the authors report state-of-the-art performance at 4–5× lower cost than existing routers in their experiments. It motivates explicit output-token budgets in future versions. RoBatch [7] jointly chooses a target model and batch size under a total cost budget. The authors formulate the problem as NP-hard and propose a two-stage approach combining a batch-aware proxy utility model with greedy scheduling. This is relevant to high- throughput workloads, while the present project focuses on interactive query routing and caching.

E. Cascading, Quality-Aware Routing, and Adaptive Compute

FrugalGPT [8] combines prompt adaptation, LLM approximation (including reuse of earlier completions), and LLM cascading. It shows that cost can be reduced before, during, and after model selection. The proposed router is a lightweight pre-inference control layer, while its response cache relates to approximation; cascading could be added when an initial model is uncertain or fails a quality check. Hybrid LLM [9] routes between smaller and larger models using predicted query difficulty and a desired quality level. The quality target can be adjusted at test time; the authors report up to 40% fewer calls to the large model with no reported quality drop in their experiments. BEST-Route [10] jointly selects a model and the number of sampled responses according to query difficulty and quality thresholds. Its experiments report up to 60% cost reduction with less than 1% performance drop in evaluated settings. These studies show that routing can control inference effort as well as model identity.

F. Router Benchmarking and Online Feedback

RouterEval [11] introduces a comprehensive benchmark with more than million performance records, twelve LLM evaluations, and more than 8,500 LLMs across knowledge, commonsense reasoning, semantic understanding,

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 3, November 2026
Impact Factor: 8.2

mathematics, and instruction-following. It emphasizes consistent test sets and multiple metrics rather than inexpensive model selection alone. SLARouter [12] learns online from sparse, one-sided production feedback, balancing inference cost and user satisfaction with theoretical guarantees for cost optimality and satisfaction constraints. The authors report operating-cost reductions of up to 2.2× over evaluated baselines. Such feedback could eventually adapt routing thresholds while maintaining a satisfaction target.

G. Literature Synthesis

Across the twelve studies, the field moves conceptually from cascades and model approximation [8] to explicit quality–cost selection [4], preference-based routing [5], resource- and output- aware decisions [3], [6], batch-aware inference [7], hybrid quality thresholds and adaptive compute[9], [10], common benchmarking [1], [11], and online feedback [12]. This progression motivates a project architecture that begins with transparent rules and instrumentation, then evolves toward adaptive routing as evidence accumulates.

III. PROBLEM STATEMENT AND OBJECTIVES

Problem statement: In a multi-provider application, sending every request to a high-capability LLM can increase inference expenditure even when a less expensive model can satisfy the request. A practical application-level controller is needed to analyze each query, reuse suitable cached responses, select a model using cost and capability information, and record performance evidence without requiring a trained router or GPU- cluster optimizer. Aim: To propose an interpretable LLM Cost Autopilot baseline that reduces avoidable LLM API expenditure while respecting the capability and quality needs of incoming requests. Objectives: (1) characterize query length, task type, complexity, estimated tokens, and required capability; (2) check for an appropriate reusable response before calling a provider; (3) compare candidate models using configured pricing and capability metadata; (4) route requests through provider adapters; (5) monitor input/output tokens, estimated cost, latency, model choice, and cache status; and (6) define an evaluation protocol that considers cost, response quality, and latency together. Scope and assumptions: The design targets application/API-level multi-model serving, not GPU-cluster scheduling. Model prices and capability metadata are configured and must be maintained. Query difficulty can initially be estimated heuristically. The twelve reviewed studies form a curated reference set, not an exhaustive or systematic survey; some are recent arXiv preprints. Reported quantitative results belong to the cited authors and were not independently reproduced here. Research gap and challenges: The reviewed work leaves room for a lightweight, interpretable application-level controller that integrates query-aware routing, response caching, and unified cost/performance monitoring for small multi-provider applications. Key challenges include maintaining changing provider prices and capabilities, imperfect heuristic difficulty estimates, safe cache reuse, API failures and latency, and the absence of a baseline quality evaluator; learned approaches also require training or preference data, while evaluation depends on representative datasets and reliable feedback.

IV. PROPOSED METHODOLOGY / SYSTEM DESIGN

A. Proposed Architecture

The architecture combines query analysis, cost-aware routing, response caching, multi-provider invocation, and cost/performance monitoring. The design separates decision logic from model invocation so that pricing and routing thresholds can change without redesigning the user interface. Monitoring and caching remain reusable if a learned router replaces the rule engine in a later version.

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 3, November 2026
Impact Factor: 8.2

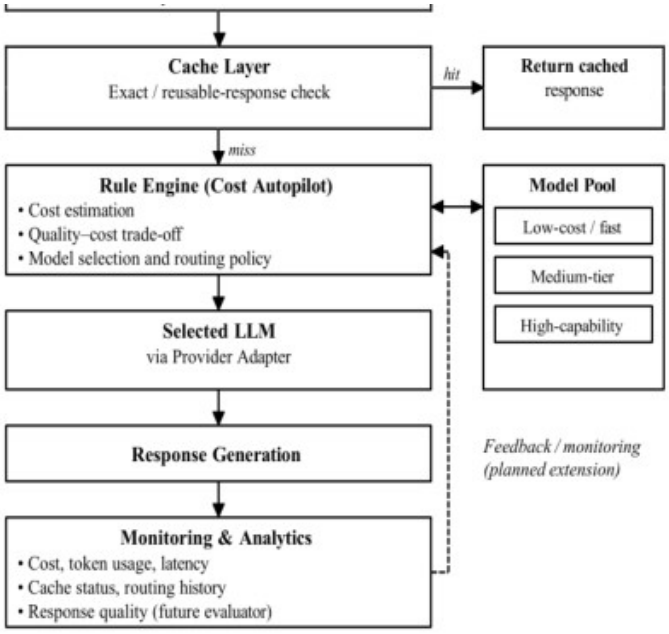


Fig. 1. Proposed architecture of LLM Cost Autopilot. Dashed feedback and response-quality paths are planned extensions; the baseline uses fixed rules and has no quality evaluator.

response-quality paths are planned extensions; the baseline uses fixed rules and has no quality evaluator.

B. Functional Components

Query Analyzer: extracts query length, estimated token requirements, task type, complexity, and required capability. Model Catalogue: stores provider/model capabilities, token- pricing metadata, and operational attributes. Rule Engine: maps query requirements to candidate models using configurable cost/capability thresholds. Cache Layer: checks whether an appropriate reusable response exists before an external call. Provider Adapter: sends the request to the selected LLM and normalizes response metadata. Monitoring Layer: records model selection, tokens, estimated cost, latency, cache status, and usage. Analytics Layer: presents routing history, cost trends, savings estimates, and model utilization.

C. Design Principle and Technology Context

The architecture is an extensible control layer. Model metadata and pricing can be updated without rewriting the application; monitoring supplies data for later learning; and caching can eliminate calls before model selection incurs an API expense. The project technology context specifies Python and FastAPI for the backend/API; React.js or another web front end; PostgreSQL/MySQL for persistence; Redis for caching; Anthropic Claude, OpenAI, and Google Gemini APIs as model providers; Requests, Pydantic, and Pandas as supporting libraries; and VS Code, Git/GitHub, and Postman for development. A GPU is not required at the application layer, but hosted APIs require a stable internet connection.

D. Routing Workflow

The methodology converts cost-aware routing principles into a deterministic application workflow. The cache is consulted first; a hit avoids routing and an external call. On a miss, the system evaluates difficulty and cost/capability before invoking the selected model, then records evidence for evaluation and policy refinement.

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 3, November 2026
Impact Factor: 8.2

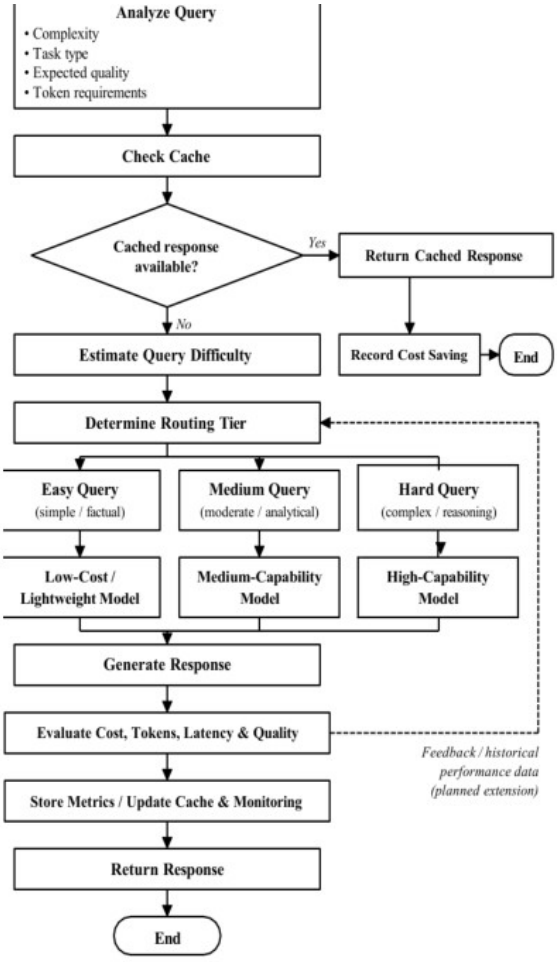


Fig. 2. Proposed methodology and routing workflow. The dashed feedback path is a planned extension; quality evaluation requires an evaluator not included in the baseline.

path is a planned extension; quality evaluation requires an evaluator not included in the baseline. Step 1—User query: the application receives a request through the web interface. Step 2—Query analysis: estimate length/tokens, task type, complexity, and required capability. Step 3—Cache check: return an appropriate stored response and record the saving on a hit; continue on a miss. Step 4—Difficulty and cost/capability evaluation: estimate difficulty and compare candidate models using configured pricing and capability metadata. Step 5—Rule-based routing: select the lowest-cost candidate satisfying the required capability threshold. Step 6—Model invocation: the selected provider adapter sends the request and returns a normalized response and usage metadata. Step 7—Monitoring: record input/output tokens, estimated cost, response time, selected model, and cache status. Step 8—Cache update: store appropriate responses for future reuse. Step 9—Analytics: aggregate routing history and cost/performance data for dashboards and reports.

E. System Modules and Routing Policy

The system analyzes incoming queries, routes requests using predefined cost/capability rules, compares models by API cost, token usage, response time, and capability, reduces unnecessary calls through response caching, and monitors LLM usage and API costs. Query analysis converts a raw request into routing signals—query length, estimated token usage, task category, complexity, and required model capability. Initial heuristics remain deterministic, with an interface for a trained difficulty predictor later. The router maintains model metadata and evaluates candidates against query requirements. A tiered policy assigns simple, low- risk queries to a low-cost/lightweight model; medium-complexity

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 3, November 2026
Impact Factor: 8.2

tasks to a balanced, medium-capability model; and complex or capability-sensitive requests to a high-capability model. Tier 1 typically serves factual questions, simple Q&A, short summaries, and information lookup with low inference cost and latency. Tier 2 supports analysis, explanations, medium-length generation, multi-step tasks, and general reasoning at moderate cost and latency. Tier supports complex reasoning, multi-domain analysis, long-form generation, and advanced problem solving at higher cost and computational requirement. Thresholds are configurable and are not fixed by this paper. The objective is to minimize LLM cost while maintaining required response quality.

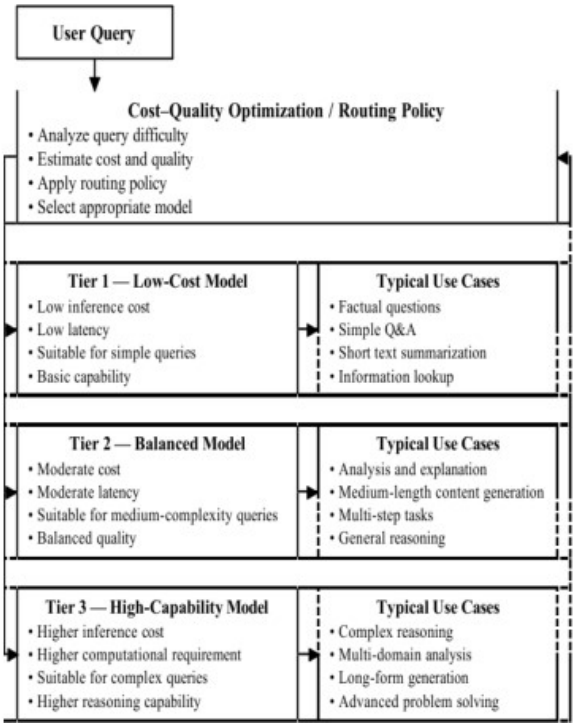


Fig. 3. Illustrative three-tier routing policy. Tier boundaries are configurable; complexity and quality requirements increase from Tier 1 to Tier 3.

complexity and quality requirements increase from Tier 1 to Tier 3. Response caching is treated as cost avoidance, not merely latency optimization. Exact or appropriately reusable responses can be returned without a provider call. Future semantic caching may use embeddings and similarity thresholds, but relevance must be evaluated before production use. The monitoring layer records token consumption, estimated API cost, response time, model usage, cache hits/misses, and routing history. These data support analytics and research evaluation and provide historical observations for later learned routing or feedback-based adaptation.

V. RESULTS AND DISCUSSION

A. Evaluation Framework and Baselines

A research publication should distinguish a proposed architecture from measured performance. The project material defines the architecture and evaluation objectives, but no experimentally validated numerical results are available; therefore, no cost- saving or quality-improvement result is claimed for LLM Cost Autopilot. Evaluation should begin after a representative test set and model-price configuration are fixed. The primary baseline is fixed premium-model routing, where every request is sent to a high-capability model. Additional baselines are fixed low-cost routing and simple complexity-tier routing. These comparisons separate savings caused by routing from savings caused by caching.

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 3, November 2026
Impact Factor: 8.2

B. Evaluation Metrics

The recommended metrics are: total estimated cost (sum of input/output API expenditure across the evaluation set); average cost per request (total cost divided by completed requests); cost savings (relative reduction against a selected baseline, reported together with quality); quality score (consistent task-specific or evaluator-based measurement); average latency (reported separately for cache hits and provider calls); cache-hit rate (fraction served from reusable responses); token consumption (input and output tokens across models); and model-selection distribution (percentage of requests routed to each candidate model).

C. Experimental Protocol and Quality–Cost Trade-off

Construct a representative query set covering simple, medium, and difficult tasks. Run the same set through the baseline and proposed router using the same model catalogue and pricing configuration. Record quality, cost, latency, token usage, cache status, and model selection. Repeat across multiple routing thresholds to examine the cost–quality frontier. Report aggregate and category-level results so that a low-cost policy that fails on difficult tasks is not mistaken for a universally effective one. The key outcome should be a trade-off curve, not a single savings number. Routing effectiveness depends on quality and cost together, consistent with CARROT [4], Hybrid LLM [9], BEST-Route
[10], LLMRouterBench [1], and RouterEval [11]. The contribution
should be judged by the cost avoided at an acceptable quality and latency level.

VI. CONCLUSION AND FUTURE WORK

This paper surveyed twelve studies on cost-aware and intelligent LLM routing. The literature spans cost-reduction mechanisms, model selection, unified routing infrastructure, resource-aware serving, preference learning, output-budget control, batching, quality-aware routing, adaptive test-time compute, benchmarking, and online learning under user-satisfaction constraints. Together, these works show that effective LLM routing is a multi-objective problem involving cost, quality, latency, compute, and operational feedback. LLM Cost Autopilot is proposed as an interpretable application- level baseline. It analyzes incoming queries, checks reusable responses, evaluates model cost and capability, applies configurable routing rules, invokes the selected LLM when required, and records token, cost, latency, model-selection, and cache information. Its evaluation framework assesses savings together with response quality and latency. The system has not yet been experimentally evaluated; validation against the defined baselines and metrics is the immediate next step. Future work includes semantic caching using embeddings and relevance thresholds; learned routing from historical quality, cost, latency, and preference data; explicit quality estimates and targets; adaptive model/output/sample-budget selection; latency- aware policies using workload conditions and response-time constraints; batch-aware routing for high-throughput workloads; dynamic model-price updates and threshold recalculation; online feedback using sparse satisfaction signals; standardized benchmark integration; and operational budget controls such as daily/monthly limits, alerts, and policy enforcement. The architecture is intended to remain a transparent baseline against which progressively more intelligent routing strategies can be evaluated.

REFERENCES

[1] H. Li et al., “LLMRouterBench: A massive benchmark and unified framework for LLM routing,” arXiv:2601.07206, 2026.
[2] T. Feng et al., “LLMRouter: Unified infrastructure for developing, evaluating, and deploying LLM routers,” arXiv:2608.06867, 2026.
[3] H. H. Kasnavieh, C. Leckie, and A. N. Toosi, “RouterWise: Joint resource allocation and routing for latency-aware multi-model LLM serving,” arXiv:2604.10907, 2026.
[4] S. Somerstep et al., “CARROT: A cost aware rate optimal router,” arXiv:2502.03261, 2025.
[5] I. Ong et al., “RouteLLM: Learning to route LLMs with preference data,” arXiv:2406.18665, 2024.
[6] J. Xue, Q. Lou, J. Xing, and H. Huang, “R2-Router: A new paradigm for LLM routing with reasoning,” arXiv:2602.02823, 2026.

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 3, November 2026
Impact Factor: 8.2

[7] H. Xu, K. Zhao, and J. Xie, "Towards cost-effective LLMs routing with batch prompting," arXiv:2605.28268, 2026.
[8] L. Chen, M. Zaharia, and J. Zou, "FrugalGPT: How to use large language models while reducing cost and improving performance," arXiv:2305.05176, 2023.
[9] D. Ding et al., "Hybrid LLM: Cost-efficient and quality-aware query routing," in Proc. 12th Int. Conf. on Learning Representations (ICLR), 2024, arXiv:2404.14618.
[10] D. Ding et al., "BEST-Route: Adaptive LLM routing with test-time optimal compute," in Proc. 42nd Int. Conf. on Machine Learning, PMLR 267, pp. 13870–13884, 2025.
[11] Z. Huang et al., "RouterEval: A comprehensive benchmark for routing LLMs to explore model-level scaling up in LLMs," arXiv:2503.10657, 2025.
[12] H. Woisetschläger, A. Mammadli, R. Zhang, and S. Wang, "Cost-optimal LLM routing with limited user feedback under user satisfaction guarantees," arXiv:2606.19376, 2026.