Efficient Training-Free Online Routing for High-Volume Multi-LLM Serving

Fangzhou Wu[1] Sandeep Silwal[1]

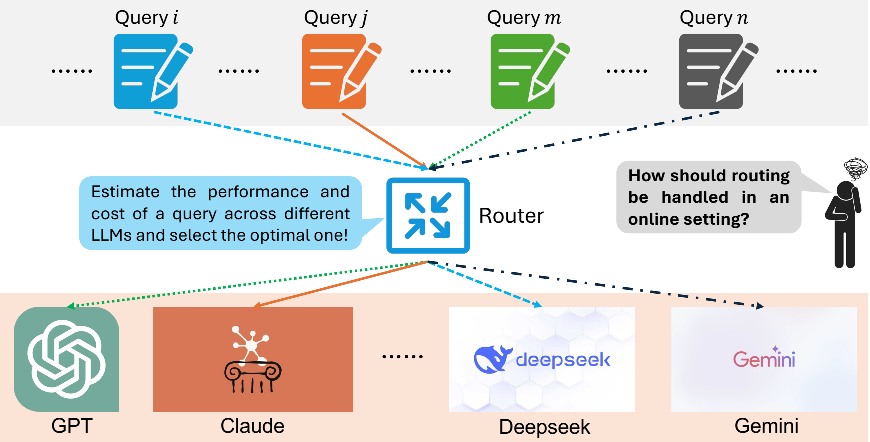
[1]University of Wisconsin–Madison

Background

1. Large Language Models (LLMs) are increasingly embedded in diverse domains.
2. The growing volume of user queries imposes substantial deployment costs on LLM-serving providers.
3. Improving the overall quality of service, particularly under limited token budgets, **has become a critical priority for LLM-serving providers**.

LLM Routing

❶ LLM routing offers a cost-efficient solution by directing queries to the optimal LLM based on model and query features.



LLM Routing

1. Existing works primarily focus on **offline** scenarios and struggle to adapt to **online** settings with high query volume and constrained token budgets.
    - **Computational Scalability**: High computational demands and added latency render them impractical for high-volume, low-latency online applications.
    - **Deployment Scalability**: Difficulty adapting to evolving LLM configurations, often requiring costly and time-consuming retraining for any deployment changes.
    - **Sequential Query Arrival**: Fundamental ineffectiveness in handling the sequential, unpredictable nature of real-world query arrivals with constrained token budgets, due to their reliance on offline assumptions.

Our Solution

1. **Efficient Performance and Cost Estimation**.
   - For each query, we employ Approximate Nearest Neighbor Search to efficiently estimate its features (performance and cost) for each deployed LLM using a historical dataset.
2. **Online Routing from Observed Queries.**
   - We formulate routing as a Mixed-Integer Linear Program.
   - At optimality, the dual objective can be fully parameterized by a single dual variable, which yields the optimal routing rule and inspires treating it as **learnable LLM weights** optimized through the dual objective.
   - We estimate these weights by optimizing over a **small** initial set of observed queries, then use these learned weights to route subsequent queries. A control parameter is also added to the MILP objective to improve generalization to future queries.
3. **Theoretical Guarantees.** We provide theoretical guarantees demonstrating that our algorithm **achieves a competitive ratio of** $1 - o(1)$ **under mild assumptions**.

Main Results

Table 1: The main results on RouterBench, SPROUT, and Open LLM Leaderboard v2 are under a total budget equal to the cost of the cheapest model, split across models based on the cost efficiency. Here, Perf represents the *Performance*, PPC represents *Performance per Cost*, Tput represents *Throughput*, and RP denotes *Relative Performance* compared with offline approximate optimum ($\hat{C}_{opt}$).

Algorithm	**RouterBench**					**SPROUT**					**Open LLM Leaderboard v2**				
	Perf	Cost	PPC	Tput	RP	Perf	Cost	PPC	Tput	RP	Perf	Cost	PPC	Tput	RP
Random	1384.25	0.427	3243.25	3276	43.10%	2827.6	0.72	3927.29	4742	47.61%	953.0	0.741	1284.37	2877	49.89%
Greedy-Perf	1012.1	0.27	3742.379	1687	31.52%	764.9	0.406	1881.742	1083	12.88%	553.0	0.499	1107.91	1189	28.95%
Greedy-Cost	1626.25	0.46	3534.46	4061	50.64%	3934.7	0.849	4630.41	6789	66.25%	1051.0	0.766	1371.30	3164	55.02%
KNN-Perf	1005.1	0.27	3720.58	1677	31.3%	769.6	0.407	1888.46	1084	12.96%	556.0	0.498	1114.29	1194	29.11%
KNN-Cost	1592.05	0.46	3454.04	4027	49.58%	3905.1	0.85	4593.37	6709	65.75%	991.0	0.766	1293.07	3172	51.88%
BatchSplit	1838.05	0.458	4005.93	3903	57.24%	3975.5	0.83	4784.49	6221	66.94%	1059.0	0.76	1392.07	3099	55.44%
Roberta-Perf	154.5	0.077	2019.00	190	4.81%	458.9	0.283	1621.64	536	7.73%	153.0	0.207	738.21	283	8.01%
Roberta-Cost	481.4	0.129	3738.88	1292	14.99%	3996.2	0.848	4709.22	6765	67.29%	1044.0	0.766	1362.53	3173	54.66%
Ours	**2718.6**	0.447	**6075.58**	**5195**	**84.66%**	**4513.05**	0.815	**5536.74**	**7475**	**75.99%**	**1465.0**	0.711	**2060.3**	**3692**	**76.7%**
Offline Oracle (Algorithm Upper Bounds Reference)															
Approx Optimum($\hat{C}_{opt}$)	3211.35	0.46	6975.16	6225	100%	5938.99	0.85	6986.45	8781	100%	1910.0	0.765	2493.66	4319	100%
Optimum (C_{opt})	6376.9	0.46	13865.62	6436	198.57%	11934.4	0.848	14060.34	12336	200.94%	4688.0	0.763	6143.64	4688	245.44%

Robustness Results (Query Volume)

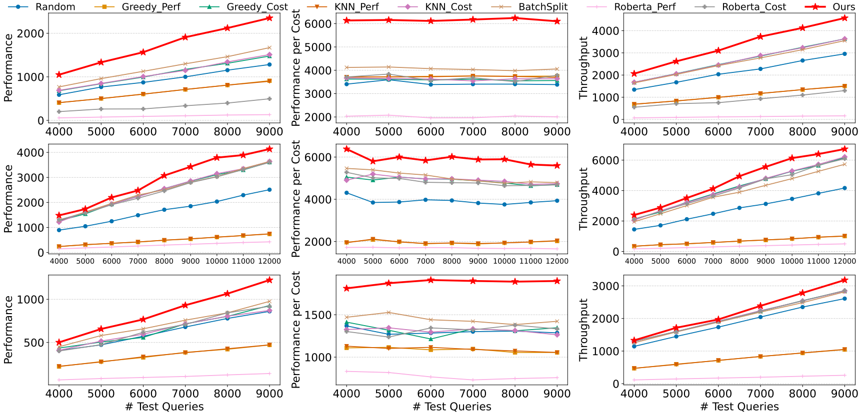


Figure: Results with test query volume varying from 4000 to 9000 (12000). Rows correspond to different datasets: RouterBench (top), SPROUT (middle), and Open LLM Leaderboard v2 (bottom).

Query Arrival Order & Scalability to LLM Deployments

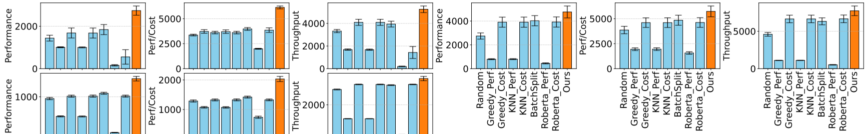


Figure: Results under 100 random query orders. (Left to right): the first three subfigures show results on RouterBench, the next three on SPROUT, and the last three on Open LLM Leaderboard v2.

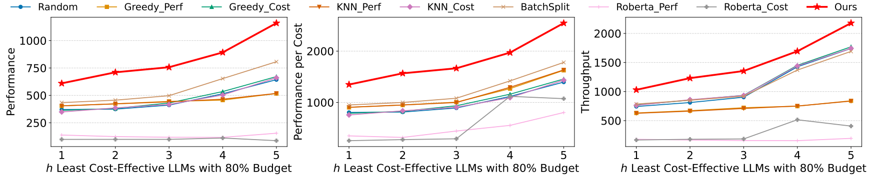


Figure: Results on Open LLM Leaderboard v2 when varying LLM deployment configurations.

Total Budget

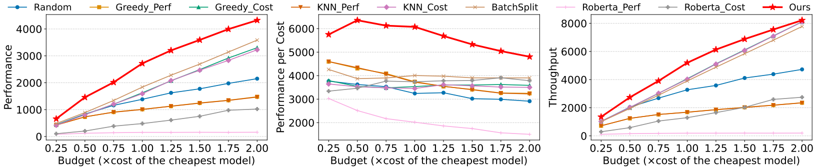


Figure: Results on RouterBench with budget from 0.25 to 2$\times$ the cost of the cheapest model.

Budget Split

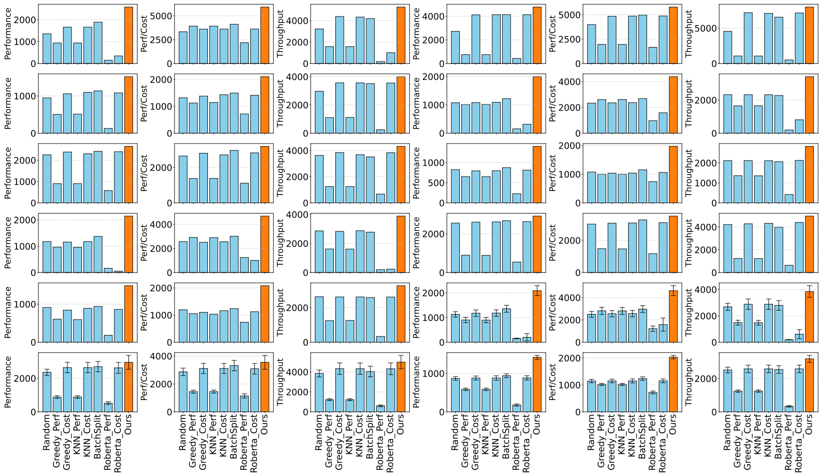


Figure: Results under different budget splitting strategies. Each group of 9 subfigures corresponds to one strategy: (a) cost-based split (subfigures 1–9), (b) performance-based split (10–18), (c) uniform split (19–27), and (d) random split (28–36). Within each group, the first three subfigures correspond to RouterBench, the next three to SPROUT, and the last three to Open LLM Leaderboard v2.

Thanks

Thank you!

References I