

“AlphaRoute: Large Language Models as Semantic Optimizers for Multi-Objective Routing”

IEEE International Conference on LLM-Aided Design, 2026
(LAD 2026), Stanford University, USA

Kabir Murjani^{1*}, Mishri Bhavsar², **Manish I. Patel²**, Jonti Talukdar³

¹Department of Electrical Engineering, Institute of Technology, Nirma University, Ahmedabad, India

²**Department of Electronics and Communication Engineering, Institute of Technology, Nirma University, Ahmedabad, India**

³Center for Semiconductor Microelectronics, Arizona State University, Tempe, Arizona, USA

Outline

- Problem Statement
- Methodology
- Implementation Flow
- Results
 - Ablation Study
 - LLM Comparison
- Conclusion
- References

Problem Statement & Abstract

- **The Problem:** VLSI global routing is an NP-hard combinatorial optimization problem that requires routing signal nets across capacity-constrained 3D grids.
- **The Limitation:** Traditional heuristic routers rely on static penalty schedules that struggle to resolve complex congestion topologies.
- **The Solution (AlphaRoute):** A multi-objective adaptive search framework that reformulates rip-up and reroute (R&R) into a dynamic optimization system.
- **Performance:** Evaluated on ISPD 2025 benchmarks, AlphaRoute reduces overflow by 98.6% on MEMPOOL and achieves a 29.8x overflow reduction over the state of the art on ARIANE.

Major Contributions

- **Targeted Rip-Up:** SHapley Additive exPlanations (SHAP)-inspired overflow decomposition isolates per-net congestion rather than relying on coarse heuristics.
- **Adaptive PathFinder Policy:** A dynamic cost model with SHAP-driven rules that adjusts history penalties, present overflow weights, and via costs iteratively.
- **Dijkstra Maze Rerouting:** Subgraph extraction via 3D Dijkstra routing allows medium-area nets to efficiently detour around congestion hotspots.
- **Safe LLM Integration:** LLMs interpret congestion metrics to propose policy adjustments, which are then strictly validated against a deterministic knowledge graph.

Methodology

Equations	Purpose	Description
$\text{idx}(z, y, x) = z \cdot Y_{\max} \cdot X_{\max} + y \cdot X_{\max} + x$	Node Indexing: Maps a 3D GCell to a single graph node ID.	Flattens the 3D routing grid into RustworkX's node index space so every layer, row and column has a unique node.
$W(e) = (10.0 - 3.0 \cdot \sigma_n) + (11 - c_{z,u}) + (11 - c_{z,v})$	Initial Weight: Sets the base routing cost of an edge before any congestion history exists.	Combines a criticality-based discount—up to 30% for the most timing-critical net—with a capacity-repulsion term that penalizes routing through already-congested cells.

Equation	Purpose	Description
$\sigma_n = \max \left(0, 1 - \frac{s_n}{s_{min}} \right)$	Time Criticality factor: Normalizes each net's timing slack into a [0,1] urgency score.	The net with the most negative slack gets $\sigma_n=1$, earning the largest discount and routing first; less critical nets receive progressively smaller discounts.
$A_n = (x_{max}^n - x_{min}^n)(y_{max}^n - y_{min}^n)$ $\Omega(path) = \sum_{(z,y,x) \in path} \max(0, -c_{z,y,x})$	Initial Routing: Decides which routing algorithm (Steiner / L-route / MST) a net is routed through.	Small nets ($A_n \leq 2,500$) → Steiner tree; Large nets ($A_n > 2,500$) → MST+FlexL with horizontal and vertical first approach. The L-routing having lesser path congestion is chosen.
$\Omega = \sum_{z,y,x} \max(0, -c_{z,y,x})$	Global Overflow: Total unresolved congestion remaining after a routing pass.	Sums negative remaining capacity across the entire grid; $\Omega = 0$ signals completed routing, otherwise the adaptive rip-up-and-reroute loop is triggered.

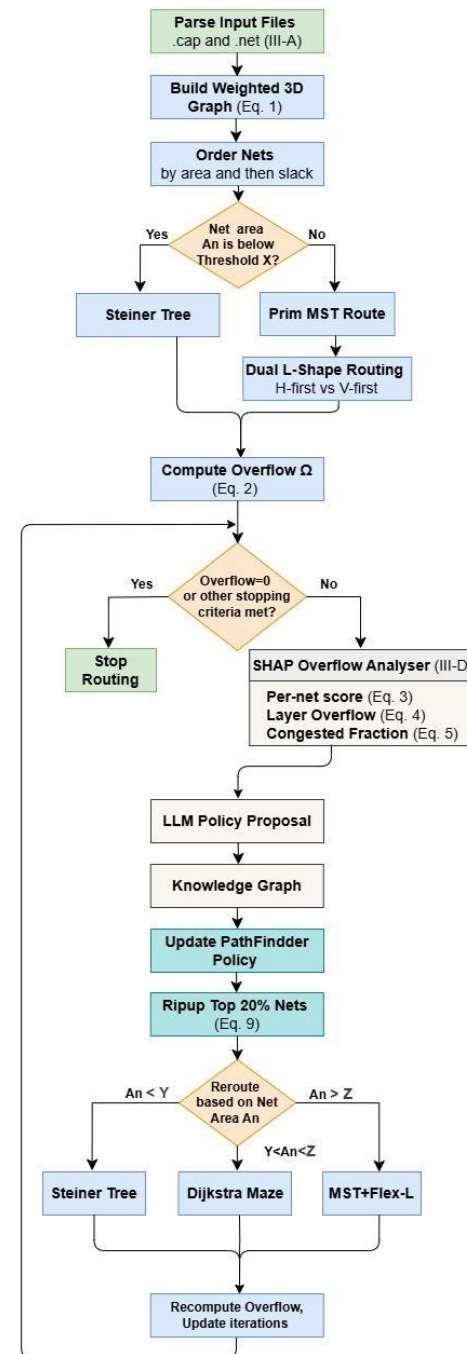
Equation	Purpose	Description
$\psi_n = \sum_{(z,y,x) \in \mathcal{G}_n} \max(0, -c_{z,y,x})$	Per-net overflow score: Attributes congestion to individual nets, enabling targeted rip-up rather than heuristic selection.	Applies the same per-cell overflow measure as overflow (Ω). Nets are ranked by ψ_n , and the top 20% of contributors are ripped up and rerouted (Eq. 3).
$\lambda_z = \frac{\sum_{y,x} \max(0, -c_{z,y,x})}{\sum_{z,y,x} \max(0, -c_{z,y,x})}.$	Layer-wise overflow decomposition: Identifies metal layers having most congestion.	Identifies metal layers having the most congestion. The dominant layers $z^* = \arg \max_z \lambda_z$ are passed to the adaptive policy and LLM analyzer.
$f_c = \frac{ \{(z,y,x) : c_{z,y,x} < 0\} }{N_L \cdot y_{max} \cdot x_{max}}$	Congested cell fraction: Measures how closely cell utilization predicts overflow, used to choose between history-driven and present-driven cost inflation.	f_c is the congested cell fraction, defined as the ratio of GCells with negative remaining capacity to the total grid size.

Equation	Purpose	Description
$u_{z,y,x} = 1 - \frac{c_{z,y,x}}{c_{z,y,x}^0}$	Utilization Definition: Defines cell utilization as the basis for the correlation test in Eq. (6).	$c_{z,y,x}^0$ is the original capacity from the .cap file; utilization rises toward 1 as remaining capacity shrinks toward 0.
$r_{uo} = \frac{Cov(u, \max(0, -c))}{\sigma_u \cdot \sigma_{\max(0, -c)}}$	Utilization-overflow correlation: Pearson correlation between cell utilization and binary overflow, used to choose between history-driven and present-driven cost inflation (per Table I).	A high value ($r_{uo} > 0.5$) indicates cell utilization is a reliable predictor of overflow, forcing the policy to increase the history penalty weight α . A low correlation indicates congestion is structurally driven (e.g., macro blockages) rather than demand-driven, in which case α is not increased.

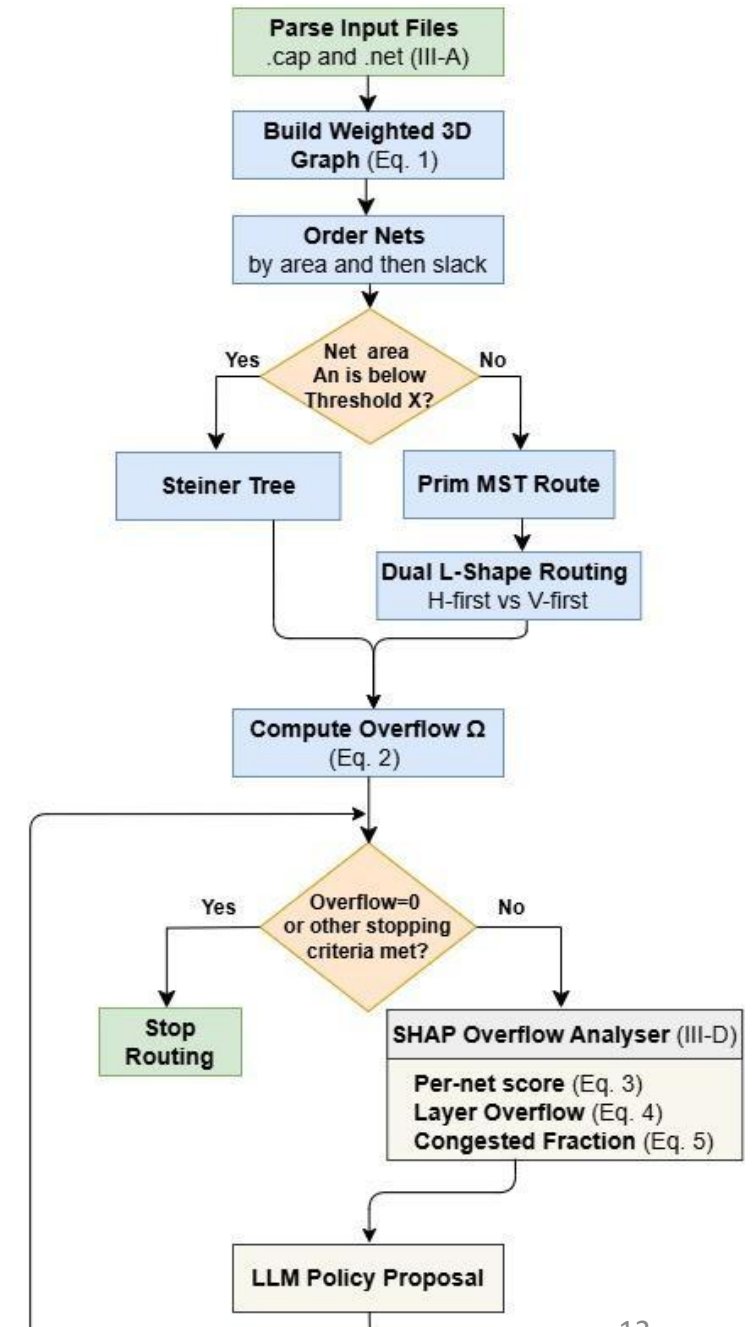
Equation	Purpose	Description
$h^{(k+1)}(e) = h^{(k)}(e) + \alpha^{(k)} \cdot \max(0, -c_e),$	History cost update: Updates the history cost array for every edge at the end of each iteration, building persistent memory of chronic congestion.	ce is minimum remaining capacity of the two cells incident to the edge. The history array is persistent across all R&R iterations; cells that are repeatedly congested accumulate an increasing penalty.
$cost(e) = W(e) + h^{(k)}(e) + p_f^{(k)} \cdot \max(0, -c_e),$	Reroute cost: The combined edge cost the Dijkstra algorithm uses during rerouting.	The three additive terms correspond to: the capacity-weighted base cost, accumulated history penalty from past iterations, and the present-state overflow penalty pf.
$\alpha^{(k+1)} = \alpha^{(k)} \times 1.15,$	Automatic Alpha Growth: Ensures the history penalty increases even without LLM input.	α grows by a factor of 1.15 each iteration.

Equation	Purpose	Description
$\mathcal{R}^{(k)} = \{n \in \mathcal{N} : \psi_n \geq \psi_{\lfloor (1-\rho) \mathcal{N} \rfloor}\}$	Rip-Up Candidate Selection: Defines the rip-up candidate set for iteration k, given the SHAP scores computed by Eq. (4).	p = 0.20 is the rip-up fraction and N is the set of nets with $\psi > 0$; this selects the top 20% of nets ranked by overflow contribution. The capacity consumed by each ripped net is restored before routing begins, and the routing demand of the cell is incremented for every GCell now occupying the ripped net n, ensuring Dijkstra can find the optimum path while considering the capacity of ripped cells. The number of ripped nets decreases monotonically in the full ARIANE design across iterations.

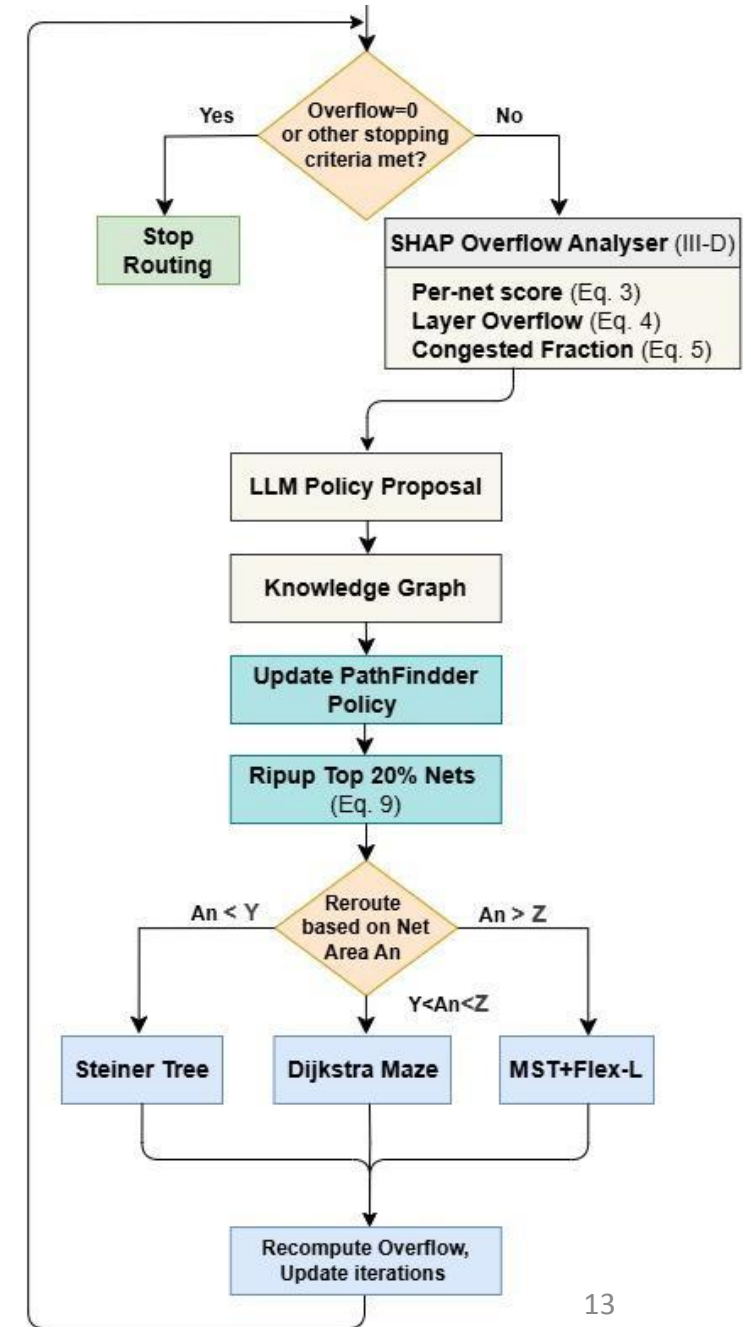
Methodology: Implementation Flow



Methodology: Implementation Flow



Methodology: Implementation Flow



Dataset

Source: The blind dataset from the ISPD 2025 Global Routing Contest [1]

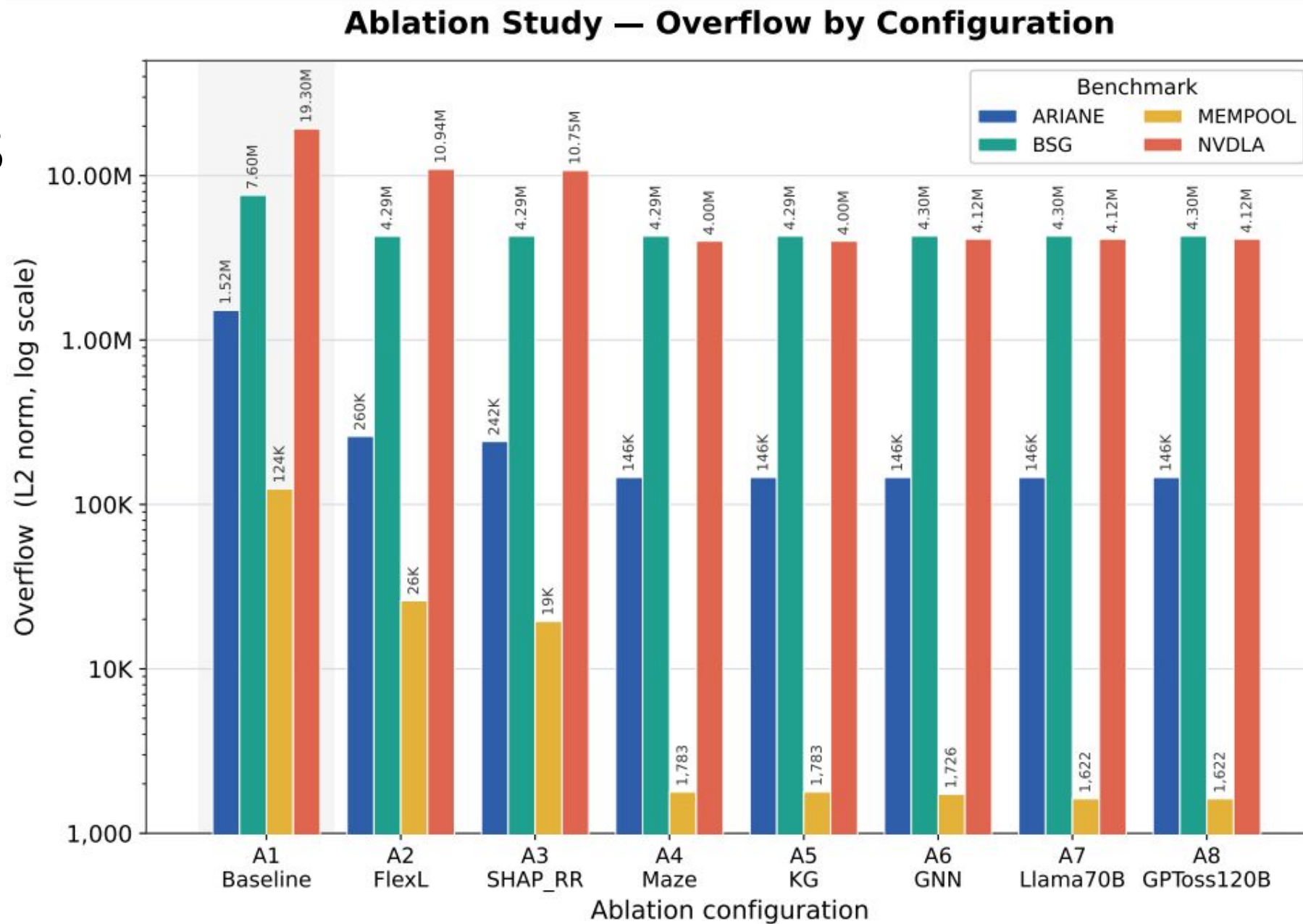
Benchmark	# Layers	Dimension	# Nets
ARIANE	10	646×646 GCells	1,05,924
BSG	10	1384×1384 GCells	7,68,239
MEMPOOL	10	1120×1120 GCells	1,57,744
NVDLA	10	386×386 GCells	1,35,814

Adaptive Policy Rules

TABLE I: Adaptive Policy Rules

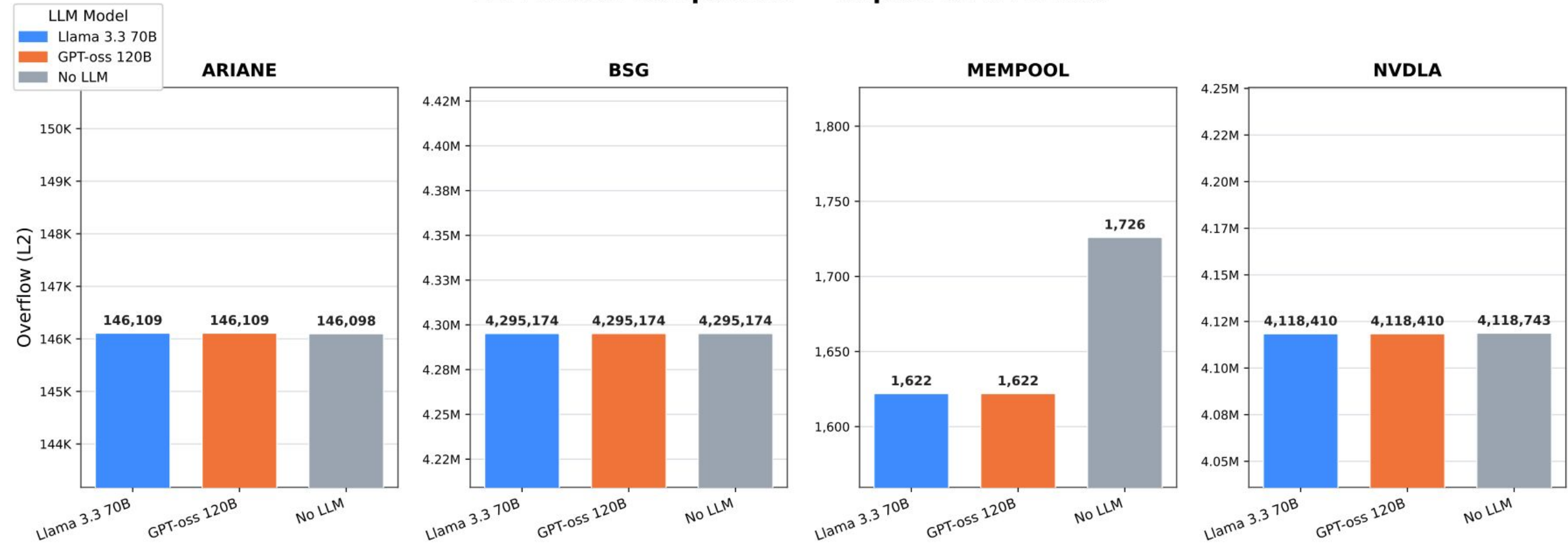
SHAP signal	Condition	Policy action
Congested cell fraction	$f_c < 0.02$	Increase p_f
Util-overflow correlation	$r_{uo} > 0.5$	Increase α
Layer concentration	$\lambda_{z^*} > 0.50$	Decrease via cost

Results: Ablations



Results: LLM Model Comparison

LLM Model Comparison — Impact on Overflow



LLM Log

```
INPUT: Routing iteration 1 metrics:  
- Overflow (L2): 258,229 - Overflow delta from init:  
+0.0% - Congested cells: 1.15% of grid - Util-overflow  
correlation: 0.237 - Top layer overflow: 50.1% -  
Current alpha: 0.4000 - Current present factor: 2.000 -  
Current via cost: 40.0 - Vias ripped this iteration: 0  
  
{ "path_action": "INCREASE", "pf_action": "INCREASE",  
  "via_action": "DECREASE", "alpha_value": 0.8,  
  "pf_value": 4.0, "via_value": 30.0, "reasoning": "High  
overflow, concentrated congestion, and high top layer  
overflow indicate need for stronger penalties and via  
cost reduction to spread routing demand." }
```

Conclusion

- AlphaRoute demonstrates that SHAP-driven overflow decomposition, adaptive PathFinder tuning, Dijkstra maze rerouting, and LLM-guided policy adaptation can be composed into a **single closed-loop system that reduces congestion** by up to two orders of magnitude on ISPD 2025 benchmarks, achieving an original score over 33x lower than the published state of the art on Ariane despite a pure Python implementation.
- Ablations confirm that flexible L-routing, SHAP-targeted rip-up, and maze rerouting account for nearly all overflow reduction, while the knowledge graph and **LLM layers provide interpretability** without degrading routing quality, and the policy loop remains model-agnostic across LLM families.
- Future work includes porting critical loops to compiled code or GPU kernels to close the runtime gap, distilling a compact routing-specific language model from accumulated knowledge-graph logs

References

1. ISPD 2025 contest: Performance-driven global routing,” 2025, contest specification and benchmarks. [Online]. Available: <https://github.com/ISPD25contest>
2. M. R. Garey and D. S. Johnson, “The rectilinear steiner tree problem is NP-complete,” *SIAM Journal on Applied Mathematics*, vol. 32, no. 4, pp. 826–834, 1977.
3. J. Liu, C.-W. Pui, F. Wang, and E. F. Y. Young, “CUGR: Detailedroutability-driven 3d global routing with probabilistic resource model,” in *Proceedings of the 57th ACM/IEEE Design Automation Conference (DAC)*. IEEE, 2020, pp. 1–6.
4. S. Lin, L. Xiao, J. Liu, and E. F. Y. Young, “InstantGR: Scalable GPU parallelization for global routing,” in *Proceedings of the 43rd IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*. ACM, 2024, published in 2025.
5. L. McMurchie and C. Ebeling, “PathFinder: a negotiation-based performance-driven router for FPGAs,” in *Proceedings of the Third International ACM Symposium on Field-Programmable Gate Arrays (FPGA)*. ACM, 1995, pp. 111–117.
6. R. A. Solovyev, I. A. Mkrtchan, D. V. Telpukhov, I. I. Shafeev, A. Y. Romanov, Y. V. Stolbikov, and A. L. Stempkovsky, “Pagr: Accelerating global routing for VLSI design flow,” *IEEE Access*, vol. 13, pp. 6440–6450, 2025.
7. S. M. Lundberg and S.-I. Lee, “A unified approach to interpreting model predictions,” in *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 30, 2017, pp. 4765–4774.
8. C. Chu and Y.-C. Ho, “FLUTE: Fast lookup table based rectilinear steiner minimum tree algorithm for VLSI design,” *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 27, no. 1, pp. 70–83, 2008.
9. P. Velicković, G. Cucurull, A. Casanova, A. Romero, P. Lió, and J. Y. Bengio, “Graph attention networks,” in *International Conference on Learning Representations (ICLR)*, 2018.
10. A. Mirhoseini, A. Goldie, M. Yazgan, J. W. Jiang, E. Songhori, S. Wang, Y.-J. Lee, E. Johnson, O. Pathak, A. Nova, J. Pak *et al.*, “A graph placement methodology for fast chip design,” *Nature*, vol. 594, no. 7862, pp. 207–212, 2021.
11. L. Zheng, W.-L. Chiang, Y. Sheng, S. Zhuang, Z. Wu, Y. Zhuang, Z. Lin, Z. Li, D. Li, E. P. Xing, H. Zhang, J. E. Gonzalez, and I. Stoica, “Judging LLM-as-a-judge with MT-bench and chatbot arena,” in *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 36, 2023.
12. L. Ouyang, J. Wu, X. Jiang, D. Almeida, C. L. Wainwright, P. Mishkin, C. Zhang, S. Agarwal, K. Slama, A. Ray *et al.*, “Training language models to follow instructions with human feedback,” in *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 35, 2022, pp. 27 730–27 744.
13. L. Gao, J. Schulman, and J. Hilton, “Scaling laws for reward model overoptimization,” in *Proceedings of the 40th International Conference on Machine Learning (ICML)*, vol. 202, 2023.
14. U. Khandelwal, O. Levy, D. Jurafsky, L. Zettlemoyer, and M. Lewis, “Generalization through memorization: Nearest neighbor language models,” in *International Conference on Learning Representations (ICLR)*, 2020.

Thank You!!