AdaptEvolve: Improving Efficiency of Evolutionary AI Agents through Adaptive Model Selection

Pretam Ray[†*], **Pratik Prabhanjan Brahma**[◇], **Zicheng Liu**[◇] **and Emad Barsoum**[◇]
[†] IIT Kharagpur, [◇] Advanced Micro Devices, Inc. (AMD)

Abstract

Evolutionary agentic systems intensify the trade-off between computational efficiency and reasoning capability by repeatedly invoking large language models (LLMs) during inference.This setting raises a central question: *how can an agent dynamically select an LLM that is sufficiently capable for the current generation step while remaining computationally efficient?* While model cascades offer a practical mechanism for balancing this trade-off, existing routing strategies typically rely on static heuristics or external controllers and do not explicitly account for model uncertainty. We introduce *AdaptEvolve: Adaptive LLM Selection for Multi-LLM Evolutionary Refinement* within an evolutionary sequential refinement framework that leverages intrinsic generation confidence to estimate real-time solvability. Empirical results show that confidence-driven selection yields a favorable Pareto frontier, reducing total inference cost by an average of **37.9%** across benchmarks while retaining **97.5%** of the upper-bound accuracy of static large-model baselines. Our codebase is available at https://github.com/raypretam/AdaptEvolve.

1 Introduction

The computational cost of state-of-the-art large language models (LLMs) remains a key barrier to scalable deployment. While recent advances in small-model distillation (Grattafiori et al., 2024; Abdin et al., 2024) and self-consistency mechanisms (Wang et al., 2022) have substantially improved the capabilities of efficient models in the 1–4B parameter range, a persistent performance gap relative to frontier models continues to limit their standalone applicability in complex reasoning tasks.

At the same time, agentic reasoning paradigms have evolved from single-turn generation toward *iterative evolutionary refinement* involving multiple LLMs, as exemplified by AlphaEvolve (Novikov et al., 2025), OpenEvolve (Sharma, 2025), and ShinkaEvolve (Lange et al., 2025). These frameworks iteratively generate, mutate, and select candidate solutions using multiple models. Also, GEAK-OptimAgentV2 (Wang et al., 2025) uses OpenEvolve framework for generating Triton kernels. However, model usage within such systems is typically governed by fixed sampling weights or predetermined schedules, without explicit consideration of which model is most appropriate for a given refinement step. Recent efforts have explored accelerating agentic systems through improved planning and search strategies (Yu et al., 2025; Wu et al., 2025; Lange et al., 2025), while parallel lines of work have studied *multi-model inference systems* that explicitly optimize the quality-cost trade-off predominantly fall into three paradigms: *query routing*, where a classifier predicts input difficulty and dispatches requests accordingly (Ong et al., 2024); *speculative decoding*, which employs smaller models to draft candidate tokens for verification by larger models (Kim et al., 2023); and *model cascades*, which progressively escalate queries from lower-cost to higher-capacity models (Chen et al., 2023).

In this work, we draw inspiration from multi-model inference systems to accelerate evolutionary agentic coding frameworks composed of multiple LLMs. We argue that effective model selection in this setting should be driven by *intrinsic uncertainty signals* rather than external routing models. Prior work has shown that entropy and logit-based measures expose systematic confidence signatures that correlate with model failure modes and reasoning breakdowns (Fu et al., 2025; Liu et al., 2024). These signals are readily available at inference time and incur negligible additional computational cost.

Building on these observations, we propose *AdaptEvolve*, a framework that conditions model al-

[*]Work done during Internship at Advanced Micro Devices, Inc. (AMD)

Findings of the Association for Computational Linguistics: ACL 2026, pages 40625–40633

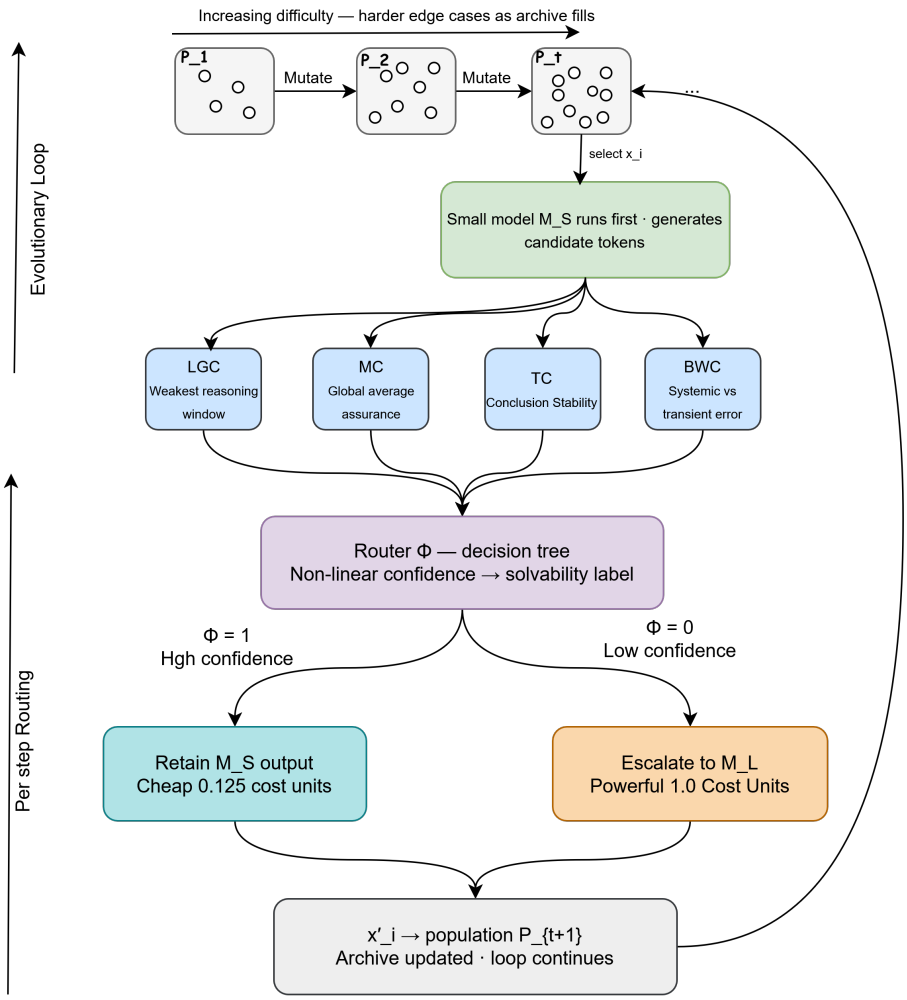

Figure 1: **Adaptive Evolutionary Refinement Framework.** The workflow initiates with a candidate generation using the small model ($\mathcal{M}_S$). We compute intrinsic confidence metrics (LGC, MC, TC, BWC) based on token entropy. A lightweight decision tree router (Φ), bootstrapped via a warm-up phase, dynamically determines whether to retain the efficient generation or escalate to the large model ($\mathcal{M}_L$) for complex reasoning hurdles.

location on intrinsic generation uncertainty within evolutionary agentic settings. This creates a dynamic decision process where efficient models handle routine tasks, and powerful models are selectively invoked for high-entropy steps. To our knowledge, this is the first application of uncertainty-aware adaptive selection in evolutionary agentic refinement.

Our contributions are:

- **Lightweight Calibration:** We introduce a resource-efficient calibration method that bootstraps a shallow decision tree using intrinsic entropy metrics (Fu et al., 2025) from a minimal warm-up set ($N = 50$), eliminating the need for heavyweight routers.
- **Pareto Efficiency:** Empirical evaluation demonstrates a superior Pareto frontier, reducing total inference compute by **37.9%** while retaining **97.5%** of the upper-bound performance of static large-model baselines.

2 Methodology

2.1 Adaptive Evolutionary Refinement

Our framework builds upon OpenEvolve (Sharma, 2025), which iteratively optimizes solution candidates, represented as code generations P_t, through mutation and selection. Standard evolutionary approaches utilize a fixed model for all mutations $\mathcal{M}$, ignoring the dynamic variance in reasoning difficulty. We propose **AdaptEvolve: Adaptive Evolutionary Refinement**, transforming the mutation operator into a conditional routing function.

Let $\mathcal{M}_S$ and $\mathcal{M}_L$ denote a cost-efficient model (e.g., 4B) and a capability-dense model (e.g., 32B),

Configuration	LiveCodeBench V5				MBPP			
	Ratio (S:L)	Cost	Acc	Eff.	Ratio (S:L)	Cost	Acc	Eff.
Baselines								
4B Iterative (Lower Bound)	100:0	0.46	62.3	135.4	100:0	0.37	80.1	216.5
32B Iterative (Upper Bound)	0:100	3.17	75.2	23.7	0:100	1.18	94.0	79.7
Static Routing								
Decision Tree (Gini, Depth=5)	43:57	2.02	71.2	35.2	96:04	0.46	82.2	178.7
Random Sampling	43:57	1.98	67.5	34.1	96:04	0.45	81.3	180.7
Online Adaptation								
Adaptive Hoeffding Tree (Bifet and Gavaldà, 2009)	**42:58**	**2.08**	**73.6**	**35.4**	**85:15**	**0.69**	**91.3**	**132.3**
Random Sampling (Online Ratio)	42:58	2.01	68.2	33.9	85:15	0.68	88.3	129.9

Table 1: **Comparative Analysis of Adaptive Routing across Benchmarks.** We evaluate cost-efficiency trade-offs. *Cost* denotes Compute Cost (one 32B call = 1 units and one 4B call = 0.125 units). *Eff* (**Efficiency** (Eff = Accuracy/Compute Cost)).

respectively. For a candidate $x_i \in P_t$, we compute a confidence vector $C(x_i)$ from the generation statistics of $\mathcal{M}_S$. The mutation for the subsequent generation is defined as:

$$x_i' = \begin{cases} \mathcal{M}_S(x_i) & \text{if } \Phi(C(x_i)) = 1 \\ \mathcal{M}_L(x_i) & \text{otherwise} \end{cases} \quad (1)$$

where $\Phi : \mathbb{R}^d \to \{0, 1\}$ is a lightweight binary classifier predicting solvability. This mechanism creates a dynamic computation graph where $\mathcal{M}_S$ handles routine refinement steps, and $\mathcal{M}_L$ is reserved for high-entropy reasoning hurdles.

A key property distinguishing evolutionary search from generic multi-step generation is *population-level concept drift*: as the MAP-Elites archive fills with high-scoring solutions, subsequent mutations must address increasingly difficult edge cases, causing the difficulty distribution to shift non-stationary over time. Static routers degrade in this setting, as evidenced by the 2.4-point accuracy gap between the static Decision Tree and HAT on LiveCodeBench (Table 1). To our knowledge, AdaptEvolve is the first routing framework to address this non-stationarity via test-time online adaptation, distinguishing it from prior single-step routing approaches such as RouteLLM (Ong et al., 2025).

2.2 Entropy-Based Confidence Metrics

We quantify generation uncertainty via *Token Confidence* (c_t), defined as the negative mean log-probability of the top-k tokens at position i (Fu et al., 2025):

$$c_i = -\frac{1}{k} \sum_{j=1}^{k} \log P_i(j) \quad (2)$$

High confidence corresponds to peaked distributions and greater model certainty, while low confidence indicates uncertainty in token prediction. We also use four scalar metrics (Fu et al., 2025) over the sequence for uncertainty quantification:

- **Mean Confidence (MC):** The global average uncertainty ($\frac{1}{T}\sum c_i$), serving as a proxy for overall model assurance.
- **Lowest Group Confidence (LGC):** The window with the maximum uncertainty score ($\min G_i$), identifying the localized "weakest link" in the reasoning chain.
- **Tail Confidence (TC):** The uncertainty of the final W tokens (G_{T-W+1}), targeting the stability of the solution's conclusion.
- **Bottom-K% Confidence (BWC):** The average of the highest K-th percentile of window scores, distinguishing systemic hallucination from transient token noise.

3 Experiment

3.1 Experimental Setup

We implement our adaptive framework on top of OpenEvolve (Sharma, 2025), an open-source evolutionary framework inspired by AlphaEvolve (Novikov et al., 2025), an AI-coding Agent, which performs iterative population-based refinement of candidate solutions using large language models as mutation and evaluation operators. The experiments were executed on a single node with 8× AMD Instinct™MI250 GPUs. The code generations are dynamically updated using a map-elites (Mouret and Clune, 2015) style archive, where previous high-scoring generations serve as few-shot

exemplars for the next step. We define hyperparameters of the system in subsection A.4.

3.2 Models

We evaluate the following multi-LLM system from the Qwen3 family (Yang et al., 2025) known for strong reasoning capabilities: Qwen3-4B (Small/Router Target) and Qwen3-32B (Large/Escalation Target).

3.3 Dataset and Metrics

We select coding benchmarks that provide deterministic ground-truth signals also due to the fact that Openevolve generates code as solution representations.

- LiveCodeBench v5 (Jain et al., 2024): A challenging code generation benchmark requiring syntax correctness and functional logic, consisting of 880 samples.
- MBPP (Austin et al., 2021): The benchmark consists of around 1,000 crowd-sourced Python programming problems, consisting of 974 samples.

Metrics: We evaluate performance using accuracy, assigning a binary score of 1 to generated solutions that pass all test cases and 0 otherwise.

Cost Metric Validity. We normalize inference cost as: Cost $\propto$ Parameters $\times$ Tokens. Since our setup enforces a strict 20,000-token maximum per call (Table 6) and empirically yields matched mean generation lengths ($\sim$15K tokens) across all configurations with no systematic token inflation from smaller models, total token counts are effectively controlled. Under these constraints, model size becomes the dominant scaling factor, making the call-count normalization (32B $= 1$, 4B $= 0.125$) a faithful proxy for actual FLOPs.

Configuration	Ratio (S:L)	Cost (Norm.)	Accuracy	Eff.
Cascading Baseline (Chen et al., 2023)	36:64	2.81	73.8	26.3
Adaptive Hoeffding Tree (Ours)	**42:58**	**2.08**	**73.6**	**35.4**

Table 2: **Comparison with Cascading.** The Adaptive Hoeffding Tree achieves a superior Pareto trade-off. While the raw accuracy is comparable, our method demonstrates significantly higher Efficiency (**35.4** vs 26.3) due to intelligent routing that lowers compute cost by 0.73.

3.4 Baselines

We compare our Adaptive Confidence Router against both static and heuristic baselines:

- **Pure Small / Pure Large:** The lower and upper bounds of performance and cost.
- **Random Routing:** Randomly selecting models to establish a chance baseline.
- **Cascade:** We start with the smaller model always and as soon as the smaller model fails we defer to the bigger model.

3.5 Methodolody

The mapping from confidence signatures to solvability is non-linear and non-stationary. A simple threshold on Mean Confidence often fails to detect confident hallucinations (low MC, high LGC) subsection A.2

Static Decision Tree To bootstrap Φ, we execute a minimal warm-up phase ($N = 50$) using $\mathcal{M}_S$ and $\mathcal{M}_L$ to generate a labeled dataset $\{(C(x_i), y_i)\}$, where y_i denotes whether that sample will be solved by the model or not Table 7. We train an initial Decision Tree (Buitinck et al., 2013) (Gini impurity, max_depth=5) on this subset, enabling inference that captures non-linear interactions between metrics (e.g., low MC but high LGC).

Online Adaptation To handle the non-stationary nature of evolutionary search, we employ a **Hoeffding Adaptive Tree (HAT)** (Bifet and Gavaldà, 2009) using (Montiel et al., 2021) package. HAT incrementally updates split criteria and monitors leaf error rates. Upon detecting concept drift, it automatically prunes and regrows decision branches, enabling the router to dynamically recalibrate escalation thresholds as reasoning difficulty evolves.

3.6 Results and Analysis

We quantify agent acceleration using **Efficiency** (Eff $=$ Accuracy$/\mathcal{C}_{\text{total}}$), where $\mathcal{C}_{\text{total}}$ represents the cumulative inference cost. We standardize computational units such that a single invocation of the 32B model incurs a cost of 1.0, while the efficient 4B model incurs a cost of 0.125. As shown in Table 1, **AdaptEvolve** establishes a superior Pareto frontier across both difficulty regimes. Figure 2

Benchmark Performance On *LiveCodeBench*, our method retains **97.9%** of the 32B upper-bound accuracy (73.6% vs 75.2%) while reducing compute cost by **34.4%**. This yields an efficiency score of **35.4**, significantly surpassing the static 32B model (23.7) and the Cascading baseline (26.3; see Table 2). The acceleration is magnified on *MBPP*, where the router identifies that 85% of queries are solvable by the small model. This reduces cost by **41.5%** while maintaining **97.1%** of peak accuracy, resulting in an efficiency score of **132.3**, nearly double that of the pure large model (79.7).

Generalization Across Model Families. To validate that AdaptEvolve is not restricted to the Qwen3 family, we evaluate on LLaMA 3.1 models (Grattafiori et al., 2024) using the HumanEval benchmark (Chen et al., 2021). As shown in Table 3, the Adaptive Hoeffding Tree consistently outperforms random sampling under identical routing ratios (38:62), achieving 74.3% accuracy versus 69.4% for random selection, while attaining a 1.56× speedup over the 70B iterative baseline. These results confirm that the confidence-driven routing strategy generalizes across architectures without modification.

Configuration	Ratio (S:L)	Cost	Acc.	Speedup
8B Iterative	100:0	0.54	64.3	3.33×
70B Iterative	0:100	4.16	77.1	1×
HAT (Ours)	38:62	3.61	74.3	1.56×
Random Sampling	38:62	3.57	69.4	1.15×

Table 3: Generalization to LLaMA 3.1 models on HumanEval. HAT outperforms random sampling at the same routing ratio, confirming model-agnostic applicability of AdaptEvolve. Speedup is relative to the 70B iterative baseline.

Routing Dynamics We analyze the impact of adaptation mechanisms:

- **Adaptation to Drift:** The Adaptive Hoeffding Tree (Bifet and Gavaldà, 2009) strictly outperforms static decision trees. On LiveCodeBench, it improves accuracy by 2.4 points (73.6% vs 71.2%) over the static baseline by dynamically adjusting decision boundaries as the population evolves toward complex edge cases.
- **Overall Efficiency:** Across benchmarks, uncertainty-aware routing reduces inference compute by an average of **37.9%** while recovering **97.5%** of the upper-bound performance, confirming that intrinsic confidence provides a robust signal for safe agentic acceleration.

4 Conclusion

We introduced *AdaptEvolve*, a confidence-driven routing framework that accelerates evolutionary agentic coding systems by optimizing the cost-capability trade-off at each refinement step. By conditioning model selection on intrinsic generation uncertainty, our approach eliminates the need for heavy external controllers. Empirical evaluation demonstrates that this strategy is highly effective: it reduces total inference compute by **37.9%** while retaining **97.5%** of the upper-bound accuracy. These results confirm that intelligent resource allocation is a viable pathway for scalable agentic reasoning, enabling significant acceleration without compromising solution quality.

5 Limitations

While our adaptive framework demonstrates significant efficiency gains, we acknowledge several limitations inherent to our experimental design and methodology:

Our confidence metrics (LGC, MC, TC, BWC) rely on access to token-level log-probabilities. This restricts the applicability of our method to open-weight models or APIs that expose logprobs. It cannot be directly applied to closed systems that return only generated text, limiting its deployment with certain proprietary frontier models.

References

Marah Abdin, Jyoti Aneja, Harkirat Behl, Sébastien Bubeck, Ronen Eldan, Suriya Gunasekar, Michael Harrison, Russell J Hewett, Mojan Javaheripi, Piero Kauffmann, and 1 others. 2024. Phi-4 technical report. *arXiv preprint arXiv:2412.08905*.

Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie Cai, Michael Terry, Quoc Le, and 1 others. 2021. Program synthesis with large language models. *arXiv preprint arXiv:2108.07732*.

Albert Bifet and Ricard Gavaldà. 2009. Adaptive learning from evolving data streams. pages 249–260.

Lars Buitinck, Gilles Louppe, Mathieu Blondel, Fabian Pedregosa, Andreas Mueller, Olivier Grisel, Vlad Niculae, Peter Prettenhofer, Alexandre Gramfort, Jaques Grobler, Robert Layton, Jake VanderPlas, Arnaud Joly, Brian Holt, and Gaël Varoquaux. 2013. API design for machine learning software: experiences from the scikit-learn project. In *ECML PKDD*

Workshop: Languages for Data Mining and Machine Learning, pages 108–122.

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

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, and 39 others. 2021. Evaluating large language models trained on code.

Yichao Fu, Xuewei Wang, Yuandong Tian, and Jiawei Zhao. 2025. Deep think with confidence. *Preprint*, arXiv:2508.15260.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, and 1 others. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

Naman Jain, King Han, Alex Gu, Wen-Ding Li, Fanjia Yan, Tianjun Zhang, Sida Wang, Armando Solar-Lezama, Koushik Sen, and Ion Stoica. 2024. Livecodebench: Holistic and contamination free evaluation of large language models for code. *arXiv preprint arXiv:2403.07974*.

Sehoon Kim, Karttikeya Mangalam, Suhong Moon, Jitendra Malik, Michael W Mahoney, Amir Gholami, and Kurt Keutzer. 2023. Speculative decoding with big little decoder. *Advances in Neural Information Processing Systems*, 36:39236–39256.

Robert Tjarko Lange, Yuki Imajuku, and Edoardo Cetin. 2025. Shinkaevolve: Towards open-ended and sample-efficient program evolution. *arXiv preprint arXiv:2509.19349*.

Shudong Liu, Zhaocong Li, Xuebo Liu, Runzhe Zhan, Derek F. Wong, Lidia S. Chao, and Min Zhang. 2024. Can LLMs learn uncertainty on their own? expressing uncertainty effectively in a self-training manner. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 21635–21645, Miami, Florida, USA. Association for Computational Linguistics.

Jacob Montiel, Max Halford, Saulo Martiello Mastelini, Geoffrey Bolmier, Raphael Sourty, Robin Vaysse, Adil Zouitine, Heitor Murilo Gomes, Jesse Read, Talel Abdessalem, and 1 others. 2021. River: machine learning for streaming data in python.

Jean-Baptiste Mouret and Jeff Clune. 2015. Illuminating search spaces by mapping elites. *Preprint*, arXiv:1504.04909.

Alexander Novikov, Ngân Vũ, Marvin Eisenberger, Emilien Dupont, Po-Sen Huang, Adam Zsolt Wagner, Sergey Shirobokov, Borislav Kozlovskii, Francisco J. R. Ruiz, Abbas Mehrabian, M. Pawan Kumar, Abigail See, Swarat Chaudhuri, George Holland, Alex Davies, Sebastian Nowozin, Pushmeet Kohli, and Matej Balog. 2025. Alphaevolve: A coding agent for scientific and algorithmic discovery. *Preprint*, arXiv:2506.13131.

Isaac Ong, Amjad Almahairi, Vincent Wu, Wei-Lin Chiang, Tianhao Wu, Joseph E. Gonzalez, M Waleed Kadous, and Ion Stoica. 2024. Routellm: Learning to route llms with preference data. *Preprint*, arXiv:2406.18665.

Isaac Ong, Amjad Almahairi, Vincent Wu, Wei-Lin Chiang, Tianhao Wu, Joseph E. Gonzalez, M. Waleed Kadous, and Ion Stoica. 2025. RouteLLM: Learning to Route LLMs with Preference Data. *arXiv preprint*. ArXiv:2406.18665 [cs].

Asankhaya Sharma. 2025. Openevolve: an open-source evolutionary coding agent.

Jianghui Wang, Vinay Joshi, Saptarshi Majumder, Xu Chao, Bin Ding, Ziqiong Liu, Pratik Prabhanjan Brahma, Dong Li, Zicheng Liu, and Emad Barsoum. 2025. Geak: Introducing triton kernel ai agent & evaluation benchmarks. *Preprint*, arXiv:2507.23194.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. 2022. Self-consistency improves chain of thought reasoning in language models. *arXiv preprint arXiv:2203.11171*.

Jiaqi Wu, Qinlao Zhao, Zefeng Chen, Kai Qin, Yifei Zhao, Xueqian Wang, and Yuhang Yao. 2025. Gap: Graph-based agent planning with parallel tool use and reinforcement learning. *Preprint*, arXiv:2510.25320.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, and 41 others. 2025. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*.

Zhaojian Yu, Kaiyue Feng, Yilun Zhao, Shilin He, Xiao-Ping Zhang, and Arman Cohan. 2025. Alphasearch: Accelerating new algorithm discovery with language models. *Preprint*, arXiv:2511.08522.

A Appendix

A.1 Details of Confidence Metrics

To construct the feature vector $C(x_i)$, we adopt an uncertainty quantification approach based on Fu et al. (2025). Rather than relying on raw probability margins, we utilize negative token entropy as the fundamental unit of confidence, defined as

	LGC		MC		TC		BWC	
Statistic	**Unsolved (0)**	**Solved (1)**	**Unsolved (0)**	**Solved (1)**	**Unsolved (0)**	**Solved (1)**	**Unsolved (0)**	**Solved (1)**
Mean	5.858	7.982	7.649	9.761	8.509	10.285	6.373	8.732
Std Dev	1.535	2.570	1.756	1.981	1.839	1.858	1.791	2.522
Skew	2.044	0.935	0.790	0.698	0.533	0.205	1.708	0.720

Table 4: **Statistical Separability of Confidence Metrics.** A comparison of generation statistics (Mean, Standard Deviation, and Skewness) between Unsolved (0) and Solved (1) instances on the calibration set. A distinct shift in the Mean values across all metrics (LGC, MC, TC, BWC) indicates a clear decision boundary, supporting the hypothesis that confidence is a strong predictor of correctness (on a stratified subset of 50 samples).

Token Confidence metric c_t, defined as the negative average log-probability of the top-k at position i:

$$c_i = -\frac{1}{k}\sum_{j=1}^{k}\log P_i(j) \tag{3}$$

where $P_i(j)$ is the probability of the i-th token. Under this formulation, values close to 0 indicate model uncertainty (peaked distributions), while larger values indicate high certainty.

Lowest Group Confidence (LGC) LGC detects localized reasoning collapses by identifying the "weakest link" in the generation. We compute the average confidence over sliding windows of size W and select the minimum:

$$\text{LGC} = \min_i \left(\frac{1}{W} \sum_{j=i}^{i+W-1} c_j \right) \tag{4}$$

Mean Confidence (MC) MC serves as a proxy for global model self-assurance, averaging entropy across the full sequence length T:

$$\text{MC} = \frac{1}{T}\sum_{t=1}^{T} c_t \tag{5}$$

Tail Confidence (TC) Acknowledging that reasoning errors often compound (or resolve) towards the end of a chain, TC isolates the confidence of the final W tokens:

$$\text{TC} = \frac{1}{W} \sum_{t=T-W+1}^{T} c_t \tag{6}$$

Bottom-K% Group Confidence (BWC) BWC aggregates the lowest $K\%$ of window scores to distinguish between isolated token noise and systemic hallucination. Let S_K be the subset of window scores $\{G_i\}$ comprising the bottom K-th percentile:

$$\text{BWC} = \frac{1}{|S_K|} \sum_{G \in S_K} G \tag{7}$$

A.2 Statistical Analysis of Confidence Metrics

To validate the fundamental hypothesis that intrinsic model uncertainty correlates with solution correctness, we performed a statistical analysis of the confidence metrics generated during the warm-up phase. Table 4 details the distributional properties (Mean, Standard Deviation, and Skewness) of the four proposed metrics: Lowest Group Confidence (LGC), Mean Confidence (MC), Tail Confidence (TC), and Bottom-Weighted Confidence (BWC) - segregated by ground truth outcomes.

The data reveals a significant distributional shift between Unsolved (0) and Solved (1) instances. Most notably, the **Mean** values for successful generations are consistently higher across all metrics, with gaps ranging from approximately 1.7 to 2.4 units of negative entropy. For instance, LGC shifts from a mean of 5.86 in unsolved cases to 7.98 in solved instances. Additionally, the **Skewness** is markedly higher in unsolved instances for metrics like LGC (2.04 vs. 0.93) and BWC (1.71 vs. 0.72), suggesting that heavy tails of extreme uncertainty characterize incorrect generations.

This statistical separation signifies that the "confidence signature" of the small model is not random noise but a reliable predictor. The distinct margins between the class means provide the necessary signal for our Decision Tree router to establish effective decision boundaries, thereby justifying the use of these specific metrics for adaptive model escalation.

A.3 Justification for Non-Linear Routing (Pilot Study)

To validate our architectural choice of a Decision Tree over simpler heuristics, we conducted a preliminary ablation study on a stratified subset of the dataset ($N = 50$). We compared our proposed Decision Tree router against a *Threshold-based Switch*, which aggregates confidence metrics into a scalar z-score and applies a linear cutoff.

Setting	Cost	Accuracy	Eff.
4B Iterative (Lower Bound)	0.46	62.15	135.1
32B Iterative (Upper Bound)	**3.07**	**76.20**	**24.8**
Threshold Switch (z-score)	2.08	70.07	33.7
Random Sampling (35:65)	2.03	72.23	35.6
Decision Tree (Ours)	**2.02**	**74.12**	**36.7**

Table 5: **Pilot Study on Router Selection ($N = 50$).** A comparison of routing mechanisms on a stratified subset of LiveCodeBench. The **Decision Tree** achieves the highest efficiency among the mixed-model strategies, validating the effectiveness of non-linear confidence routing.

As detailed in Table 5, the linear Threshold baseline achieved an accuracy of 0.70 with a compute cost of 2.08 compute cost. Conversely, the Decision Tree router achieved a superior accuracy of 0.74 while maintaining a lower computational cost (2.02 compute cost). This performance gap indicates that the "confidence signature" of hallucinations is non-linear; the Decision Tree successfully captures all-to-one interactions between metrics (e.g., high mean confidence but low tail confidence) that a simple linear threshold fails to distinguish. Consequently, the Decision Tree was selected as the routing mechanism for the full evolutionary framework.

A.4 Hyperparameters

Category	Parameter	Value
General	Max Iterations	8
	Checkpoint Interval	1
	Random Seed	42
LLM Ensemble	Temperature	0.6
	Top-p	0.95
	Max Tokens	20000
	Timeout	3600s
	Model	Qwen3-32B, Qwen3-4B
Database / Evolution	Population Size	8
	Archive Size	3
	Number of Islands	2
	Elite Selection Ratio	0.3
	Exploration Ratio	0.2
	Exploitation Ratio	0.5
	Feature Dimensions	Pass Rate, Complexity
	Feature Bins	5
	Migration Interval	10
	Migration Rate	0.15
Evaluator	Evaluation Timeout	1200s
	Parallel Evaluations	32
	Max Retries	4
	Cascade Evaluation	False
Confidence Calculation Window	Tail Confidence	2048
	Bottom Grouped Confidence Window	2048
	Tail Grouped Confidence Window	2048

Table 6: **Evolutionary Framework Configuration.** Hyperparameters used for the updated evolutionary search and inference environment.

Current Model	Current Pass Rate (P)	Ground Truth Other Model	Target Label	Reasoning
4B	$P \geq 1.0$	(Irrelevant)	**0** (Use 4B)	4B solved efficiently; maintain status quo.
4B	$P < 1.0$	32B Solvable	**1** (Use 32B)	4B failed, but 32B is known to succeed.
4B	$P < 1.0$	32B Not Solvable	**0** (Use 4B)	Hard problem; larger model implies waste.
32B	$P \geq 1.0$	4B Solvable	**0** (Use 4B)	Both solve it; switch down to save cost.
32B	$P \geq 1.0$	4B Not Solvable	**1** (Use 32B)	Only 32B solves it; maintain larger model.
32B	$P < 1.0$	(Irrelevant)	**0** (Use 4B)	32B failed; revert to base model.

Table 7: Labeling Logic for Adaptive Model Switching ($P = 1.0$)

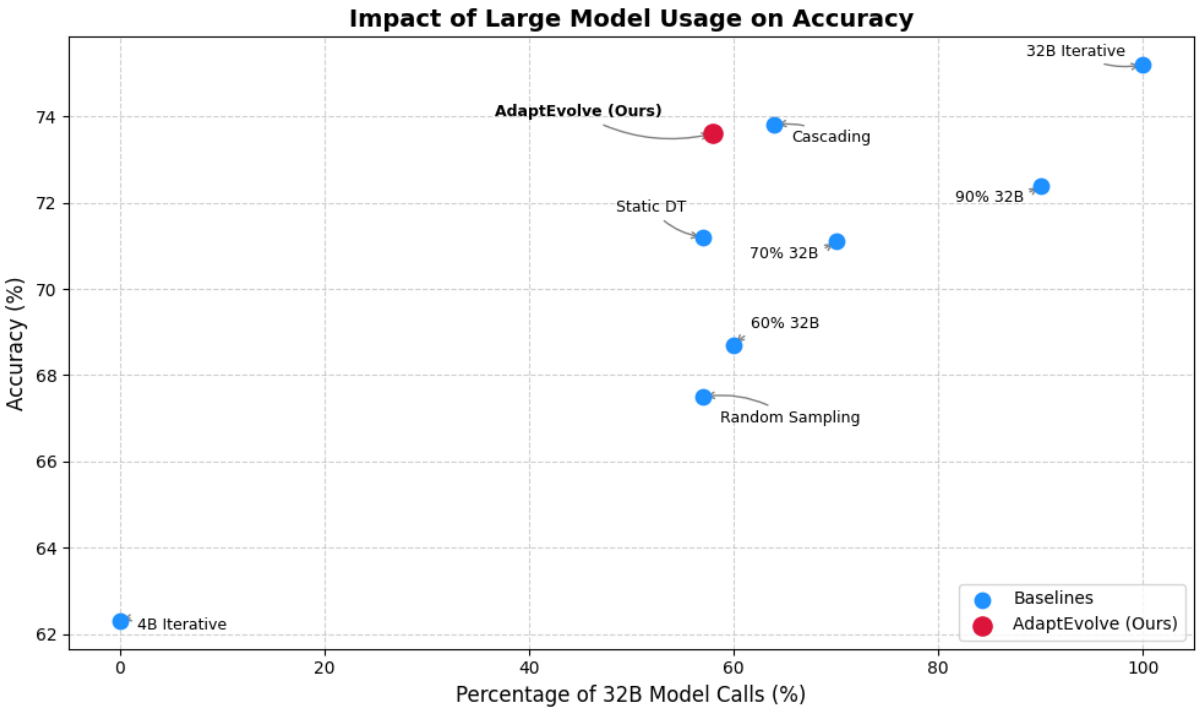

Figure 2: **Impact of Large Model Usage on Accuracy (LiveCodeBench).** We plot the percentage of calls routed to the 32B model against the final accuracy. *AdaptEvolve* (Red) achieves a significantly better trade-off than the Cascading baseline and Random Sampling, attaining near-peak accuracy with only 58% large-model usage.