Proceedings of the 4th International Conference on Software Engineering and Machine Learning
DOI: 10.54254/2755-2721/2026.34075

Adaptive Instance-Wise Budget Allocation for Sampling-Based Motion Planning

Xianying Liu

SHIEN-MING WU School of Intelligent Engineering, South China University of Technology, Guangzhou, China
202364820632@mail.scut.edu.cn

Abstract. Robotic motion planning often requires a balance between computational efficiency and solution quality. In sampling-based motion planning, different instances may have different levels of difficulty and optimization potential. A uniform small budget may lead to planning failures or insufficient path quality, whereas a uniform large budget may waste computation on simple instances. To address this issue, this paper proposes a two-stage adaptive budget allocation framework with AORRTC as the underlying planner, without modifying its internal search mechanism. The method first performs a low-budget Stage 1 probe and then decides whether to trigger large-budget Stage 2 re-planning according to Stage 1 failure, tortuosity ratio, and total curvature. Experiments are conducted on seven VAMP/Panda planning environments. The results show that the budget escalation mechanism achieves a 1.00 solved rate in all tested environments. Compared with the Fail-Trigger Method, the Three-Trigger Method reduces the average simplified path cost by 0.39 and 0.50 in cage and table_under_pick, respectively. In cage, its average total time is about 0.33 times that of the High-Budget Baseline. The proposed method provides a simple and interpretable outer-layer decision strategy for instance-wise computational budget allocation in sampling-based motion planning.

Keywords: sampling-based motion planning, adaptive budget allocation, AORRTC, path quality, total curvature

1. Introduction

Sampling-based motion planning is an important class of methods for robotic path planning, especially for single-query planning problems in high-dimensional configuration spaces and constrained environments. Classical RRT-based methods explore the configuration space through random sampling, while RRT-Connect further improves the efficiency of finding feasible paths in single-query planning by using bidirectional tree expansion and a greedy connect strategy [1, 2]. However, in practical planning tasks, obtaining a feasible path does not necessarily mean that the planning process is complete: a smaller budget can reduce planning time but may cause failures or poor path quality in difficult instances; a larger budget can usually improve feasibility or path quality, but applying it uniformly to all instances wastes computation on simple ones.

© 2026 The Authors. This is an open access article distributed under the terms of the Creative Commons Attribution License 4.0 (https://creativecommons.org/licenses/by/4.0/).

DOI: 10.54254/2755-2721/2026.34075

In practical robotic applications, this trade-off between computational efficiency and solution quality is directly meaningful. In scenarios such as real-time interaction, online manipulation planning, and task-level planning, the planner needs to return executable paths within a limited time so that the system can respond promptly to environmental changes or task requirements. At the same time, path quality can affect motion stability, execution cost, and the transition to subsequent tasks. Therefore, adaptively allocating computational budgets according to the early planning behavior of different instances is important for improving the practicality of robotic motion planning systems.

Driven by RRT*, Anytime RRT*, and AORRTC, sampling-based planning research has gradually expanded from quickly finding feasible solutions to improving path quality with additional computation time [3-6]. This means that large-budget re-planning may be valuable for difficult instances or instances with poor initial path quality. However, for many simple instances, assigning a large budget by default may not bring proportional benefits. A natural question is therefore whether, given a fixed underlying planner, early planning results can be used to determine which instances deserve a larger computational budget.

This paper studies instance-wise budget allocation for sampling-based motion planning. Unlike methods that directly modify the sampling, expansion, or rewiring mechanism of the underlying planner, this paper fixes AORRTC as the underlying planner and builds an external two-stage budget allocation framework. The method first probes each instance with a small Stage 1 budget. If Stage 1 fails, or if the solved path shows a high tortuosity ratio or high total curvature, Stage 2 large-budget re-planning is triggered; otherwise, the Stage 1 result is kept directly.

Experiments are conducted on Panda robot planning tasks in the VAMP framework, covering seven environments and comparing the Low-Budget Baseline, High-Budget Baseline, Fail-Trigger Method, and Three-Trigger Method. The results show that the budget escalation mechanism achieves a 1.00 solved rate in all tested environments. In addition, the Three-Trigger Method reduces the average simplified path cost compared with the Fail-Trigger Method in representative environments such as cage and table_under_pick. Trigger source statistics and ablation studies further show that different trigger signals are active to different degrees across environments, and that Stage 1 budget and total curvature threshold affect the trade-off among escalation rate, path quality, and computation time.

The main contributions of this paper are threefold. First, it proposes a two-stage budget allocation framework under a fixed underlying planner, transforming computational budget allocation into an instance-wise decision problem. Second, it designs an interpretable budget escalation mechanism consisting of Stage 1 failure, tortuosity ratio, and total curvature, jointly considering feasibility and the quality of already solved paths. Third, it analyzes the budget allocation behavior and parameter effects of the framework through multi-environment main experiments, trigger source statistics, and ablation studies.

The rest of the paper is organized as follows. Section 2 reviews related work on sampling-based motion planning, computational resource allocation and scheduling, and path quality evaluation. Section 3 introduces the proposed two-stage budget allocation framework and its escalation triggers. Section 4 presents the experimental setup, main results, and ablation studies. Section 5 concludes the paper and discusses future work.

2. Related work

This section reviews related work from three aspects. First, sampling-based motion planning and anytime planning are introduced to clarify the origin of AORRTC as the underlying planner. Then, computational resource allocation and planner-level scheduling in motion planning are discussed.

DOI: 10.54254/2755-2721/2026.34075

Finally, path quality evaluation and geometric indicators are reviewed to explain how this paper uses path geometry for budget escalation decisions.

2.1. Sampling-based motion planning and anytime planning

Sampling-based motion planning constructs search structures by randomly sampling in the configuration space, and it is an important approach for high-dimensional robotic path planning. RRT was originally proposed for rapid exploration in high-dimensional spaces and has been widely used in single-query path planning [1]. Based on this idea, RRT-Connect improves the efficiency of finding feasible paths in single-query planning through bidirectional tree expansion and a greedy connect strategy [2]. The AORRTC planner used in this paper is directly related to this line of work: it inherits the advantage of RRT-Connect in quickly finding an initial feasible solution and further provides anytime optimization ability for path quality improvement.

Beyond improving sampling strategies and search structures, some studies accelerate sampling-based motion planning through low-level computational subroutines. VAMP accelerates key components such as forward kinematics, collision checking, and motion validation using vectorized representations and CPU SIMD instructions, enabling microsecond-level sampling-based planning [3]. Unlike such low-level acceleration work, this paper does not focus on further accelerating a single planner call; instead, it studies instance-wise computational budget allocation based on the fast planning capability provided by VAMP/AORRTC.

While classical RRT-based methods mainly focus on quickly obtaining feasible paths, RRT* introduces asymptotic optimality into incremental sampling-based planning, showing that path cost can gradually converge to the optimum as the number of samples increases [4]. Anytime RRT* further shows that a planner can first return an executable path and then continuously improve solution quality with additional time [5]. The AO-x framework demonstrates that an existing feasible planner can be transformed into an asymptotically optimal planner through an outer construction [6]. AORRTC follows this idea by extending RRT-Connect into an almost-surely asymptotically optimal planner [7]. This paper uses AORRTC as the underlying planner, but its focus is not to modify the internal search mechanism. Instead, it uses the anytime property to decide externally which instances should enter Stage 2 large-budget re-planning.

2.2. Computational resource allocation and planner-level scheduling

In addition to the design of underlying planners, some studies improve motion planning performance from the perspective of planner-level scheduling and computational resource allocation. Learning When to Quit formulates the stopping time in motion planning as a meta-reasoning problem and learns when to stop planning and start execution under limited computational resources [8]. HyperPlan formulates motion planning as a planner selection and parameter optimization problem, showing that algorithm selection and parameter tuning at the planner level can significantly affect planning performance [9]. List Prediction Applied to Motion Planning further predicts a planner list using problem features so that a suitable planner can be found faster under time constraints [10].

These studies show that motion planning performance depends not only on the underlying search mechanism but also on outer-layer decisions and the way computational resources are used. Unlike planner selection, parameter optimization, or stopping policy methods, this paper neither selects among multiple planners nor learns a termination policy. Instead, it adds a two-stage budget

DOI: 10.54254/2755-2721/2026.34075

allocation framework outside the same underlying AORRTC planner and decides whether to perform Stage 2 large-budget re-planning according to Stage 1 failure and path geometry.

2.3. Path quality evaluation and geometric indicators

In addition to solved rate and computation time, path quality is also an important evaluation dimension in motion planning. Existing surveys point out that path quality may involve not only path length but also clearance, smoothness, energy consumption, and other criteria [11]. To address redundancy, detours, or lack of smoothness in paths generated by sampling-based planners, Geraerts and Overmars studied high-quality path generation and improved path length and clearance through shortcutting [12]. Hauser and Ng-Thow-Hing further proposed bounded-acceleration shortcutting for multi-degree-of-freedom manipulator trajectories to improve trajectory smoothness and execution quality [13].

On the other hand, geometric indicators such as tortuosity, straightness, and turning-angle-based measures are also commonly used to describe path shape complexity. Benhamou systematically discussed straightness, sinuosity, and tortuosity, providing a geometric reference for characterizing global detours using the relationship between path length and endpoint distance [14]. Unlike works that mainly use path geometry for result evaluation or path post-processing, this paper further uses the geometric features of the simplified path for computational resource allocation: after Stage 1 obtains a feasible path, the tortuosity ratio and total curvature are used to determine whether the instance should be upgraded to Stage 2 large-budget re-planning.

In summary, existing studies improve the efficiency and path quality of motion planning from perspectives such as sampling strategies, anytime planning, planner-level scheduling, and path quality evaluation. In contrast, this paper focuses on instance-wise budget allocation under a fixed underlying AORRTC planner, using early results from the low-budget stage.

3. Method

This section introduces the proposed two-stage budget allocation framework. The core idea is to first probe each planning instance with a small budget in Stage 1 and then make a budget escalation decision based on the Stage 1 result and path geometric features, namely whether the instance should enter Stage 2 for large-budget re-planning. Since the focus of this method is not to modify the internal search mechanism of AORRTC but to use early planning results for instance-wise computational budget allocation, this section mainly introduces the overall framework and the definitions and roles of the tortuosity trigger and total curvature trigger.

3.1. Two-stage budget allocation framework

Different planning instances often require different computational budgets. For simple instances, a small budget may already be sufficient to obtain a feasible path. For complex instances, however, a low budget may lead to planning failure or to paths with obvious detours and complex local turns. Using a uniformly small budget may sacrifice feasibility and path quality for difficult instances, while using a uniformly large budget may introduce unnecessary computation for simple instances. Therefore, this paper treats computational budget allocation as an instance-wise decision problem: based on the early behavior of each instance in the low-budget stage, the method decides whether the instance deserves to enter a large-budget planning stage.

DOI: 10.54254/2755-2721/2026.34075

Specifically, for each planning instance, AORRTC is first run in Stage 1 with a small Stage 1 budget. If Stage 1 fails to find a solution, the failure trigger is activated and the instance enters Stage 2 for large-budget re-planning. If Stage 1 is solved, the tortuosity ratio and total curvature of the simplified path are further computed. When either geometric indicator exceeds its corresponding threshold, Stage 2 re-planning is triggered; otherwise, the Stage 1 result is kept directly. In this way, the method can restore feasibility for instances that fail under the low budget and further identify instances that have been solved but may still have insufficient path quality.

It should be noted that this paper adopts restart-based Stage 2 re-planning. When an instance satisfies an escalation condition, Stage 2 is implemented as a new large-budget planning call rather than a continuation of the Stage 1 search tree or internal state. This design keeps the method external and minimally intrusive, allowing the budget allocation decision to be implemented without modifying the internal search mechanism of AORRTC.

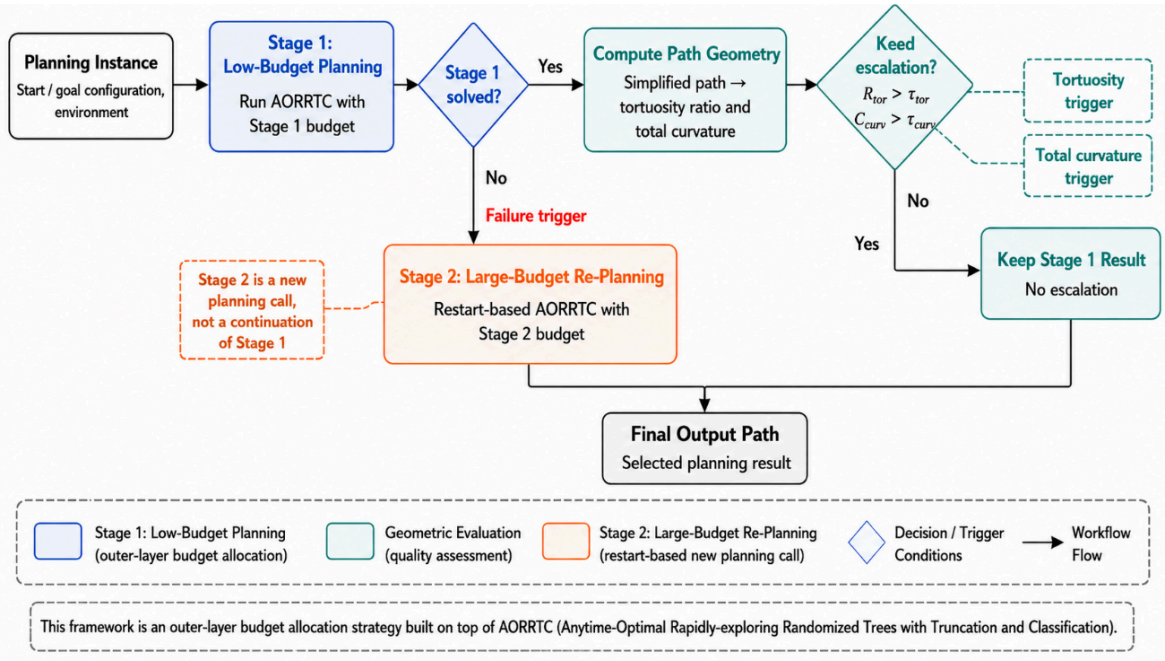

Figure 1. Overview of the proposed two-stage budget allocation framework

3.2. Tortuosity trigger

In the two-stage budget allocation framework, the failure trigger can identify difficult instances that are not solved under the low budget, but it cannot further distinguish the quality of already solved paths. Even if an instance obtains a feasible path in Stage 1, the path may still contain a significant detour, resulting in a high simplified path cost. Therefore, this paper introduces the tortuosity trigger to identify instances that are solved but have a high degree of global detour.

This paper uses the tortuosity ratio to measure the overall detour of the Stage 1 path relative to a geometric lower bound. Specifically, the tortuosity ratio is defined as the ratio between the Stage 1 simplified path cost and the lower-bound cost:

$$R_{\mathrm{tor}} = \frac{C_{\mathrm{simp}}}{C_{\mathrm{lb}}} \tag{1}$$

DOI: 10.54254/2755-2721/2026.34075

where R_{tor} denotes the tortuosity ratio, C_{simp} denotes the simplified path cost obtained after Stage 1 planning and path simplification, and C_{lb} denotes the lower-bound cost, namely the configuration-space straight-line distance from the start configuration to the nearest goal configuration. Since C_{simp} and C_{lb} are computed using the same configuration-space distance metric, R_{tor} reflects the overall detour of the current path relative to the geometric lower bound.

If Stage 1 is solved but R_{tor} exceeds the preset threshold τ_{tor}, i.e.,

$$R_{\text{tor}} > \tau_{tor} \tag{2}$$

the path is considered feasible but globally detoured, and the instance is upgraded to Stage 2 for large-budget re-planning. Conversely, if R_{tor} does not exceed τ_{tor}, the Stage 1 result is kept directly. The tortuosity trigger focuses on global path detour rather than local bending. Therefore, this paper further introduces the total curvature trigger to identify path instances with complex local geometry.

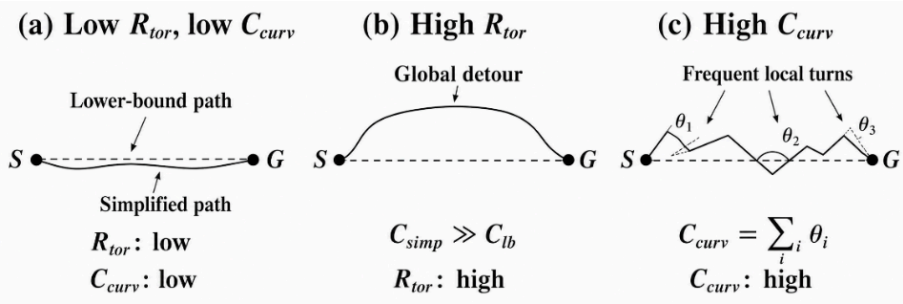

Figure 2. Illustration of tortuosity and total curvature on simplified paths

3.3. Total curvature trigger

Although the tortuosity trigger measures the global detour of a path relative to the geometric lower bound, it does not fully characterize local geometric shape. For some paths, the tortuosity ratio may not exceed the threshold, but local sharp turns, backtracking, or dense turning may still appear. Therefore, this paper further introduces the total curvature trigger to measure the local turning complexity of the simplified path. Let the simplified path obtained in Stage 1 consist of a sequence of configuration points:

$$P = \{q_0, q_1, \ldots, q_n\} \tag{3}$$

where q_i denotes the i-th configuration point along the path. For adjacent configuration points, the path segment vector is defined as:

$$v_i = q_i - q_{i-1}, i = 1,2, \ldots, n \tag{4}$$

The turning angle θ_i between two adjacent path segments is defined as:

DOI: 10.54254/2755-2721/2026.34075

$$\theta_i = arccos\left(\frac{v_i^\top v_{i+1}}{\|v_i\|\|v_{i+1}\|}\right), i = 1,2,\ldots,n-1 \tag{5}$$

Based on this, the total curvature is defined as the accumulated turning angle of all adjacent path segments:

$$C_{\text{curv}} = \sum_{i=1}^{n-1} \theta_i \tag{6}$$

where C_{curv} denotes the total curvature of the simplified path. A larger C_{curv} indicates that the path may contain more or stronger turning changes and thus has a more complex local geometry, suggesting that the path may still have further optimization potential.

Based on this definition, if Stage 1 is solved but the C_{curv} of the simplified path exceeds the preset threshold τ_{tor}, i.e.,the total curvature trigger is activated, and the instance is upgraded to Stage 2 for large-budget re-planning. Compared with the tortuosity trigger, the total curvature trigger focuses more on the accumulated local turns along the path. It should be noted that the total curvature used in this paper is a discrete turning-accumulation measure computed from the configuration points of the simplified path, rather than a strict curvature integral in the continuous trajectory sense. Its main role is to support relative judgment in the budget escalation decision.

4. Experiments

This section validates the effectiveness of the proposed method through main experiments and ablation studies. It first introduces the experimental environments, comparison methods, evaluation metrics, and parameter settings. It then analyzes the main results of four methods across multiple environments and explains the behavior of the Three-Trigger Method using trigger source statistics. Finally, two ablation studies are used to discuss the rationality of the default parameter settings.

4.1. Experimental setups

The experiments are implemented based on the VAMP framework, with AORRTC fixed as the underlying planner [3, 7]. The experiments are conducted on seven Panda robot planning environments: box, cage, table_pick, table_under_pick, bookshelf_small, bookshelf_tall, and bookshelf_thin. These environments differ in obstacle distribution, spatial constraints, and instance difficulty, including both relatively easy scenarios and more challenging constrained manipulation environments. The table_pick environment contains one invalid instance, so its effective sample size is 99, while the other environments nominally contain 100 instances each.

Four comparison methods are considered. The Low-Budget Baseline uses a small budget for all instances. The High-Budget Baseline uses a large budget for all instances. The Fail-Trigger Method upgrades to Stage 2 only when Stage 1 failure occurs. The Three-Trigger Method further combines the tortuosity trigger and total curvature trigger to determine whether an already solved path should still receive a larger budget. Comparing these four methods makes it possible to observe the differences among uniform budgeting, failure-only triggering, and the addition of geometric triggers.

The evaluation metrics include solved rate, average simplified path cost, and average total time, which measure feasibility, simplified path quality, and computational cost, respectively. In addition,

DOI: 10.54254/2755-2721/2026.34075

this paper reports the trigger source distribution in each environment, including Stage 1 failure, high tortuosity, high total curvature, and none, to explain the source of budget escalation in different scenarios. The default configuration in the main experiments is Stage 1 budget = 1000 iterations, Stage 2 budget = 10^6 iterations, and total curvature threshold = 1.4. The effects of Stage 1 budget and total curvature threshold are further analyzed in the ablation studies.

4.2. Main results and analysis

Overall, the Low-Budget Baseline suffers from reduced feasibility in some environments, while the budget escalation mechanism allows both the Fail-Trigger Method and the Three-Trigger Method to restore the solved rate to 1.00. Therefore, the additional value of the Three-Trigger Method over the Fail-Trigger Method is not mainly reflected in solved rate, but in further identifying instances that are solved yet may still have insufficient path quality.

The solved rate results show that the Low-Budget Baseline fails in several environments. In particular, the solved rate is 0.75 in cage and 0.81, 0.91, and 0.89 in bookshelf_small, bookshelf_tall, and bookshelf_thin, respectively. In contrast, the High-Budget Baseline, Fail-Trigger Method, and Three-Trigger Method all achieve a 1.00 solved rate in every environment, indicating that escalation after low-budget failure can restore feasibility for these instances.

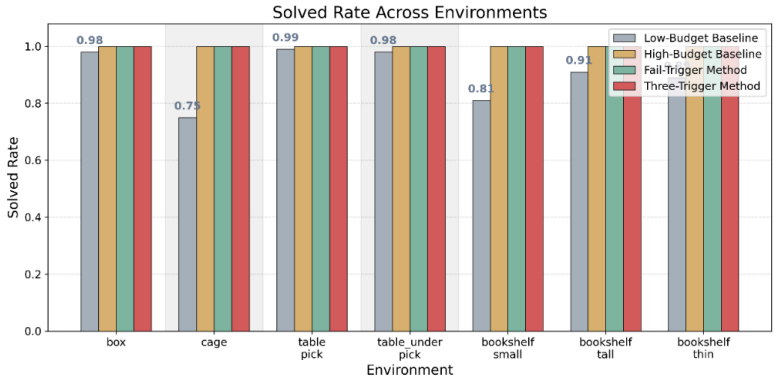

Figure 3. Solved rate across environments

The simplified path cost results show that the effect of the Three-Trigger Method is environment-selective. In cage, the Three-Trigger Method reduces the average simplified path cost by 0.39 compared with the Fail-Trigger Method; in table_under_pick, the reduction is 0.50. This indicates that the geometric triggers can identify solved paths with remaining optimization potential in some complex or constrained environments and improve path quality through large-budget re-planning. In contrast, the differences between the two methods are small in box, table_pick, and the bookshelf environments, suggesting that the additional benefit of geometric triggering is not equally significant in all environments.

DOI: 10.54254/2755-2721/2026.34075

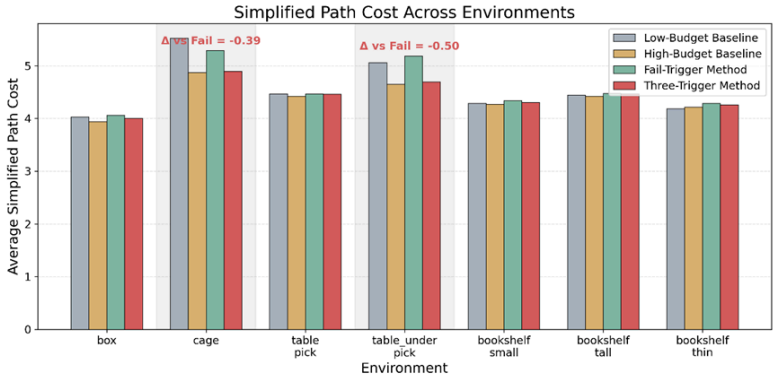

Figure 4. Simplified path cost across environments

The average total time results show that the High-Budget Baseline usually has the highest computational cost because it always uses a large budget for all instances. The Fail-Trigger Method and Three-Trigger Method perform Stage 2 large-budget re-planning only when a trigger condition is satisfied, and therefore reduce the time cost relative to the High-Budget Baseline in most environments. In cage, the average time of the Three-Trigger Method is about 0.33 times that of the High-Budget Baseline, demonstrating the efficiency advantage of selective budget allocation. However, in table_under_pick, its average time is close to that of the High-Budget Baseline, indicating that the computational cost also increases when geometric triggers are highly active.

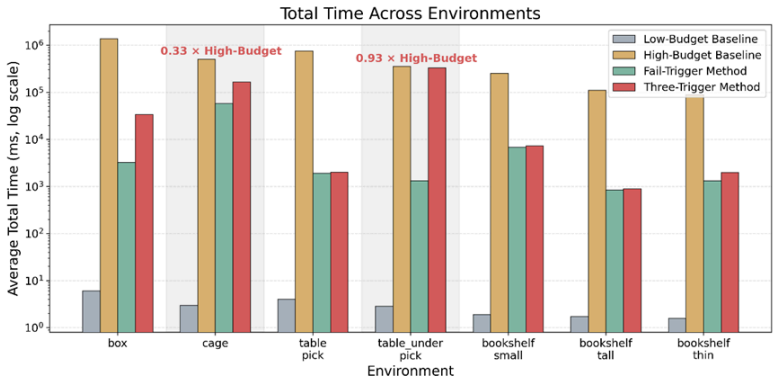

Figure 5. Total time across environments

Trigger source statistics further explain the behavior difference of the Three-Trigger Method across environments. In cage, Stage 1 failure, high total curvature, high tortuosity, and none account for 25%, 30%, 8%, and 37%, respectively, showing that this environment contains both instances that fail under the low budget and solved instances with remaining geometric optimization potential. In contrast, escalation in table_under_pick mainly comes from geometric conditions, with high tortuosity accounting for 54%, high total curvature for 21%, and Stage 1 failure for only about 2%. This indicates that geometric triggers are not intended to improve solved rate, but to identify already solved samples that still deserve optimization. By comparison, box, table_pick, and the bookshelf environments have higher none ratios, suggesting that the three-trigger mechanism does not increase computational cost indiscriminately in all environments.

DOI: 10.54254/2755-2721/2026.34075

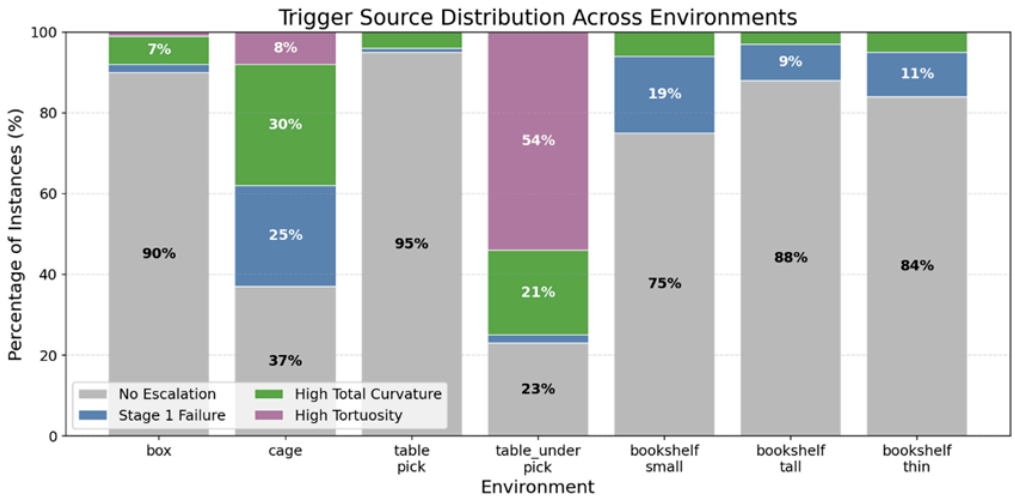

Figure 6. Trigger source distribution across environments

In summary, the core value of the Three-Trigger Method lies in more selective budget allocation. Stage 1 failure mainly restores feasibility for low-budget failures, while the tortuosity trigger and total curvature trigger further identify solved instances whose path shapes still have optimization potential. Compared with the High-Budget Baseline, the method avoids using a large budget for all instances. Compared with the Fail-Trigger Method, it further reduces simplified path cost in representative environments such as cage and table_under_pick.

4.3. Ablation study

To further analyze the influence of key parameters on budget allocation behavior and final planning results, two ablation studies are conducted. The first examines the effects of Stage 1 budget on escalation rate, average simplified path cost, and average total time. The second examines the effects of total curvature threshold on total-curvature trigger rate, path quality, and computation time. The purpose of the ablation studies is not to compare different methods again, but to explain how the default parameters influence whether escalation occurs and the resulting time-quality trade-off.

4.3.1. Stage 1 budget ablation

The Stage 1 budget is varied in three representative environments: box, cage, and table_under_pick. The box environment represents an overall easier setting with low escalation demand; cage represents a difficult environment with high budget sensitivity; and table_under_pick represents a constrained manipulation environment where geometric triggers are more active. Escalation rate is defined as the proportion of instances entering Stage 2 large-budget re-planning, equivalently the sum of instances triggered by failure, high tortuosity, and high total curvature divided by the total number of samples.

DOI: 10.54254/2755-2721/2026.34075

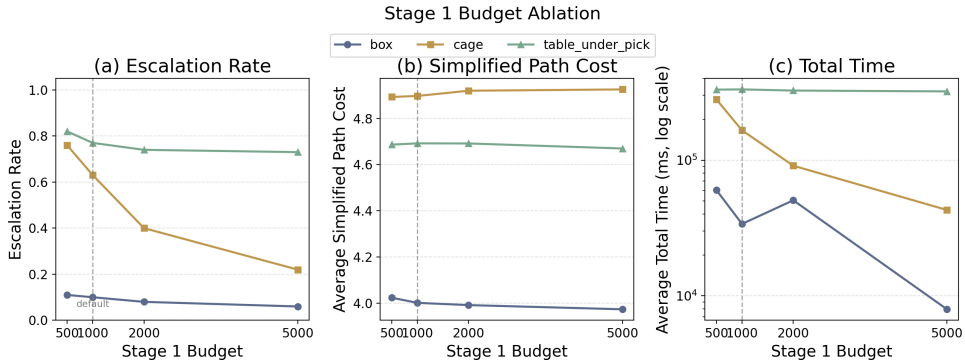

Figure 7. Stage 1 budget ablation

Figure 7 shows that as the Stage 1 budget increases, the escalation rate generally decreases in all three environments, but the magnitude of decrease differs. The box environment consistently maintains a low escalation rate. The decrease is most obvious in cage, indicating that a larger first-stage budget can significantly reduce Stage 2 calls in this difficult environment. In table_under_pick, the escalation rate remains high even with a larger Stage 1 budget, suggesting that escalation is mainly driven by geometric triggers. Meanwhile, the Stage 1 budget has a relatively limited effect on average simplified path cost, but it affects average total time by changing the escalation rate. Based on these results, the default Stage 1 budget is set to 1000 iterations as a compromise between low-cost rapid probing and subsequent selective escalation.

4.3.2. Total curvature threshold ablation

To analyze the effect of the total curvature threshold on geometric triggering, this threshold is varied in box, cage, and table_under_pick, and the total-curvature trigger rate, average simplified path cost, and average total time are observed. The total-curvature trigger rate is defined as the proportion of instances escalated due to the high total curvature condition.

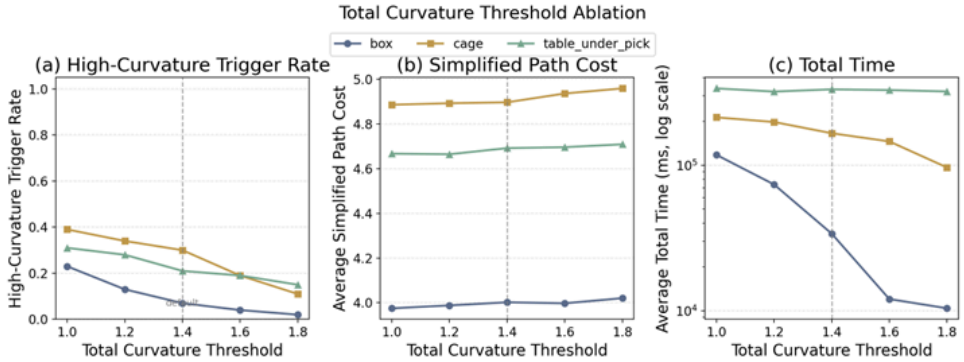

Figure 8. Total curvature threshold ablation

Figure 8 shows that as the total curvature threshold increases from 1.0 to 1.8, the total-curvature trigger rate decreases in all three environments, which is consistent with the design of this trigger: a lower threshold makes paths with moderately high local turning complexity easier to escalate, whereas a higher threshold makes the trigger more conservative. At the same time, the average simplified path cost shows a slight increasing trend, while the average total time generally decreases. This indicates that reducing trigger sensitivity can reduce the number of large-budget re-planning calls, but may also reduce some opportunities for path optimization. Based on this trend, the default

DOI: 10.54254/2755-2721/2026.34075

total curvature threshold is set to 1.4 as a compromise between path quality improvement and computational cost.

5. Conclusion and future work

This paper addresses instance-wise computational budget allocation in sampling-based motion planning and proposes a two-stage budget allocation framework based on AORRTC. Unlike methods that directly modify the internal search mechanism of the underlying planner, this paper fixes AORRTC as the underlying planner and makes a budget escalation decision externally according to early Stage 1 results. Specifically, Stage 1 failure, tortuosity ratio, and total curvature are used as interpretable trigger signals to determine whether a planning instance should enter Stage 2 for large-budget re-planning, thereby allocating more computational resources to instances that are difficult to solve under a low budget or whose path quality may still be insufficient.

The experimental results show that a uniform small-budget strategy can reduce solved rate in some environments, whereas after introducing the budget escalation mechanism, both the Fail-Trigger Method and the Three-Trigger Method achieve a 1.00 solved rate in all tested environments. In addition, the Three-Trigger Method reduces the average simplified path cost compared with the Fail-Trigger Method in representative environments such as cage and table_under_pick, showing that tortuosity ratio and total curvature provide additional information for judging the quality of already solved paths. Trigger source statistics and ablation studies also show that the three trigger signals are active to different degrees across environments, and that Stage 1 budget and total curvature threshold affect the trade-off among escalation rate, path quality, and computation time.

The value of the proposed method does not lie in unconditionally improving all metrics in every environment, but in providing a simple, minimally intrusive, and interpretable outer-layer budget allocation mechanism. Future work can proceed in three directions. First, the framework can be extended to more sampling-based planners and robot platforms to validate its generality. Second, richer online features such as environmental clutter, clearance, and planning-tree structure can be introduced, or data-driven methods can be used to learn the relationship between early signals and later optimization benefit. Third, continuation-based budget extension and multi-level budget allocation can be further studied to compare restart-based planning, state-reusing planning, and multi-stage budget scheduling in terms of computational efficiency and path quality.

References

[1] S. LaValle, "Rapidly-exploring random trees: a new tool for path planning, " Annual Research Report, 1998. [Online]. Available: https: //www.semanticscholar.org/paper/Rapidly-exploring-random-trees-%3A-a-new-tool-for-LaValle/d967d9550f831a8b3f5cb00f8835a4c866da60ad

[2] J. J. Kuffner and S. M. LaValle, "RRT-connect: An efficient approach to single-query path planning, " in Proceedings 2000 IEEE International Conference on Robotics and Automation, 2000, pp. 995–1001. doi: 10.1109/ROBOT.2000.844730.

[3] W. Thomason, Z. Kingston, and L. E. Kavraki, "Motions in Microseconds via Vectorized Sampling-Based Planning, " in 2024 IEEE International Conference on Robotics and Automation (ICRA), 2024, pp. 8749–8756. doi: 10.1109/ICRA57147.2024.10611190.

[4] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning, " The International Journal of Robotics Research, vol. 30, no. 7, pp. 846–894, 2011. doi: 10.1177/0278364911406761.

[5] S. Karaman, M. R. Walter, A. Perez, E. Frazzoli, and S. Teller, "Anytime Motion Planning using the RRT*, " in 2011 IEEE International Conference on Robotics and Automation, 2011, pp. 1478–1483. doi: 10.1109/ICRA.2011.5980479.

[6] K. Hauser and Y. Zhou, "Asymptotically Optimal Planning by Feasible Kinodynamic Planning in a State–Cost Space, " IEEE Transactions on Robotics, vol. 32, no. 6, pp. 1431–1443, 2016. doi: 10.1109/TRO.2016.2602363.

DOI: 10.54254/2755-2721/2026.34075

[7] T. Wilson, W. Thomason, Z. Kingston, and J. Gammell, "AORRTC: Almost-Surely Asymptotically Optimal Planning with RRT-Connect, " IEEE Robotics and Automation Letters, vol. 10, no. 12, pp. 13375–13382, 2025. doi: 10.1109/LRA.2025.3615522.
[8] Y. Sung, L. P. Kaelbling, and T. Lozano-Pérez, "Learning When to Quit: Meta-Reasoning for Motion Planning, " in 2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2021, pp. 4692–4699. doi: 10.1109/IROS51168.2021.9636864.
[9] M. Moll, C. Chamzas, Z. Kingston, and L. E. Kavraki, "HyperPlan: A Framework for Motion Planning Algorithm Selection and Parameter Optimization, " in 2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2021, pp. 2511–2518. doi: 10.1109/IROS51168.2021.9636651.
[10] A. Tallavajhula, S. Choudhury, S. Scherer, and A. Kelly, "List prediction applied to motion planning, " in 2016 IEEE International Conference on Robotics and Automation (ICRA), 2016, pp. 213–220. doi: 10.1109/ICRA.2016.7487136.
[11] O. Salzman, "Sampling-based robot motion planning, " Communications of the ACM, vol. 62, no. 10, pp. 54–63, 2019. doi: 10.1145/3318164.
[12] R. Geraerts and M. H. Overmars, "Creating High-quality Paths for Motion Planning, " The International Journal of Robotics Research, vol. 26, no. 8, pp. 845–863, 2007. doi: 10.1177/0278364907079280.
[13] K. Hauser and V. Ng-Thow-Hing, "Fast smoothing of manipulator trajectories using optimal bounded-acceleration shortcuts, " in 2010 IEEE International Conference on Robotics and Automation, 2010, pp. 2493–2498. doi: 10.1109/ROBOT.2010.5509683.
[14] S. Benhamou, "How to reliably estimate the tortuosity of an animal's path: straightness, sinuosity, or fractal dimension?" Journal of Theoretical Biology, vol. 229, no. 2, pp. 209–220, 2004. doi: 10.1016/j.jtbi.2004.03.016.