Crucible: Quantifying the Potential of Control Algorithms through LLM Agents

Lianchen Jia[1], **Chaoyang Li**[1], **Houde Qian**[1], **Tianchi Huang**[1], **Jiangchuan Liu**[2], **Lifeng Sun**[1,3*]
[1]Department of Computer Science and Technology, Tsinghua University,
[2]Simon Fraser University, [3]BNRist
jlc21@mails.tsinghua.edu.cn, sunlf@tsinghua.edu.cn

Abstract

Control algorithms in production environments typically require domain experts to tune their parameters and logic for specific scenarios. However, existing research predominantly focuses on algorithmic performance under ideal or default configurations, overlooking the critical aspect of Tuning Potential. To bridge this gap, we introduce `Crucible`, an agent that employs an LLM-driven, multi-level expert simulation to turn algorithms and defines a formalized metric to quantitatively evaluate their Tuning Potential. We demonstrate `Crucible`'s effectiveness across a wide spectrum of case studies, from classic control tasks to complex computer systems, and validate its findings in a real-world deployment. Our experimental results reveal that `Crucible` systematically quantifies the tunable space across different algorithms. Furthermore, `Crucible` provides a new dimension for algorithm analysis and design, which ultimately leads to performance improvements. Our code is available at https://github.com/thu-media/Crucible.

1 Introduction

Control algorithms are the core mechanisms that dynamically regulate system behavior to achieve specific objectives, with widespread applications from industrial automation to complex computer systems [1, 2]. These algorithms play pivotal roles in various contexts, including gait control in robotic systems [3], motion control for autonomous vehicles [4], adaptive bitrate (ABR) control [5–8] and congestion control [7, 9] in network applications, and scheduling control within data centers [10]. Such control algorithms are essential for ensuring system stability while simultaneously optimizing overall performance.

However, current research predominantly evaluates algorithms based on their performance under ideal conditions or with default parameters [11, 12]. This paradigm overlooks a critical reality of production environments: algorithms are always tuned by domain experts to adapt to specific scenarios. The performance of an algorithm in the wild is therefore not just a function of its default design, but also of its inherent adaptability—a property we define as its **Tuning Potential**. The lack of a systematic way to measure Tuning Potential makes it difficult to compare the practical adaptability of different algorithms and hinders its establishment as an explicit design goal.

Evaluating Tuning Potential presents a significant challenge that extends beyond conventional parameter sensitivity analysis [13]. The assessment must encompass deeper, logic-level modifications, including the incorporation of additional control branches or the integration of novel components—interventions commonly employed by domain experts. The efficacy of such structural adjustments is linked to expert subjective understanding of the underlying algorithmic logic [14].

*Corresponding author.

39th Conference on Neural Information Processing Systems (NeurIPS 2025).

This complex interplay between objective performance metrics and subjective understanding factors impedes the establishment of a Tuning Potential evaluation framework.

To address this gap, we introduce `Crucible`, the first framework designed to quantitatively evaluate the Tuning Potential of control algorithms. `Crucible` operates on two key principles. First, it employs a Large Language Model (LLM)-driven, multi-level expert simulation agent that is defined with varying capabilities, such as the ability to utilize optimization tools or perform multi-step reflection, thereby emulating how developers with different expertise levels approach algorithm tuning. This approach circumvents the prohibitive costs associated with large-scale subjective studies [15]. Second, it establishes a formalized metric to normalize potential scores across diverse environments. This metric characterizes each task environment by analyzing the performance profile that emerges when a set of probe algorithms is applied to it, thereby enabling consistent comparisons across different domains.

To validate the effectiveness and generalizability of the `Crucible` framework, our evaluation encompasses a wide spectrum of case studies, from classic control tasks (Cart-Pole [16]) to complex computer systems (ABR and scheduling control), and we validate our findings in a real-world deployment. Our experiments demonstrate that `Crucible` consistently identifies a larger optimization space than traditional Bayesian methods. By analyzing the results, we identify that an algorithm's representational capacity and comprehensibility are two primary factors influencing its potential. Finally, we show that these insights can guide targeted algorithm redesign, leading to significant performance improvements.

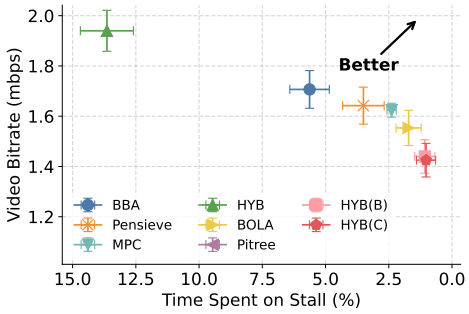
Our key contributions are summarized as follows:

- We identify and formalize Tuning Potential as a critical, yet overlooked, dimension in algorithm evaluation. We argue that neglecting this dimension limits the practical impact of algorithms and hinders its adoption as a core design objective (Section 2).
- We propose `Crucible`, the first system designed to quantify the Tuning Potential of control algorithms. Through an LLM-based multi-level expert simulation agent and a formalized environmental metric, it provides a systematic, quantifiable standard for measuring algorithmic adaptability (Section 3).
- We validate `Crucible`'s effectiveness and generalizability across a diverse range of scenarios, from classic control tasks to complex computer systems, including real-world validation. Our results show that `Crucible`'s quantitative analysis offers clear guidance for algorithm design, ultimately enhancing both performance limits and practical value (Section 4).

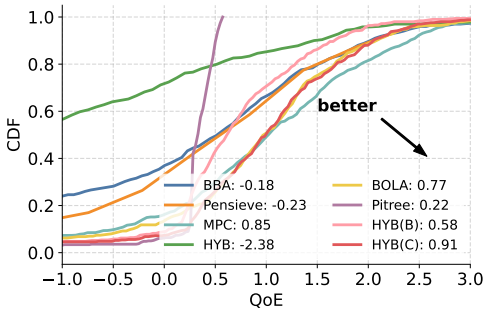
2 Motivation

2.1 LLM-Based Human Behavior Simulation

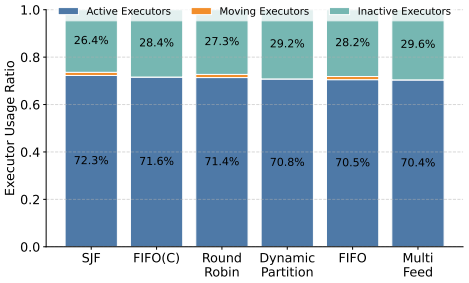
LLMs, trained on internet-scale corpora, have demonstrated exceptional performance not only in traditional natural language processing tasks such as translation, summarization, and question answering [17, 18], but also exhibited emergent general knowledge and analytical reasoning capabilities in accordance with scaling laws [19, 20]. LLM advancements have prompted researchers to explore the application of LLMs in human behavior simulation. In social sciences, numerous studies [21–23] have investigated LLM-based user behavior simulation and subjective perception assessment, successfully generating credible individual behavioral patterns and their resulting emergent social behaviors within groups. In the field of recommendation systems, researchers [24–26] have effectively utilized LLMs to simulate diverse user behaviors, including browsing, searching, and content consumption activities. In our research, we apply LLMs to simulate developers' understanding and adjustment processes of algorithms, circumventing the high costs and time expenditures associated with personnel training in traditional large-scale subjective studies, thereby providing an efficient and scalable new method for algorithm evaluation.



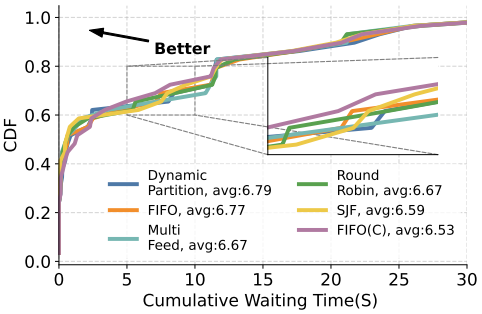
(a) ABR: Bitrate and Stall in Puffer



(b) ABR: CDF of QoE in Puffer



(c) Scheduling Control: Executor utilization Ratio in 2G Input Sizes



(d) Scheduling Control: Cumulative Waiting Time in 2G Input Sizes

Figure 1: The Importance of Algorithm Potential in the Real World

2.2 Control Algorithms in the Real World

We selected two representative examples from the field of computer systems—ABR control in network transmission and scheduling control in distributed systems—to demonstrate the effectiveness of `Crucible` in real-world control algorithms.

Adaptive Bitrate Control Adaptive bitrate algorithms enhance the quality of experience (QoE) by dynamically selecting the bitrate for the next playback chunk, aiming to improve playback quality while preventing stalling events [5]. Common control approaches include buffer-based algorithms (BBA [27], BOLA [28]), hybrid methods combining bandwidth and buffer information (MPC [29], HYB [30]), decision tree-based approaches (Pitree [31]), and reinforcement learning (RL)-based solutions such as Pensieve [32].

Scheduling Control Distributed directed acyclic graph (DAG) task scheduling algorithms are responsible for efficiently allocating and executing interdependent tasks in distributed computing environments, where dependencies are represented through DAGs [10]. These scheduling algorithms optimize task allocation mechanisms while considering resource utilization and task dependencies to enhance overall computational efficiency and reduce completion time. Common scheduling strategies include Shortest Job First (SJF), Shortest Remaining Time First (SRTF), Fair Scheduling, First-In-First-Out (FIFO), Round Robin, Dynamic Partitioning [33], Multi-feed methods [34], and Tetris [35].

2.3 From Algorithm Design to Production Deployment

During the algorithm design phase, researchers typically focus on theoretical performance [11] and cross-scenario robustness [12], pursuing universal performance across a wide range of application scenarios. However, when algorithms are actually deployed in production environments, we can obtain stable feature information specific to the application scenario, which enables targeted optimization. For instance, user bandwidth demands in network content provider services [36] and system loads in task scheduling scenarios [37] often exhibit stable and predictable patterns of change. This stability and predictability of scenario characteristics allow developers to customize optimization strategies

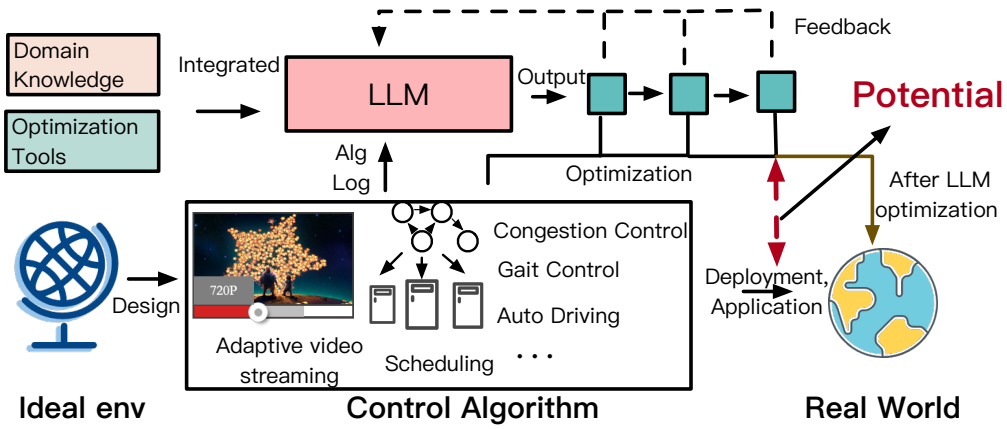


Figure 2: `Crucible` System

according to specific application environments, thereby achieving performance that surpasses general-purpose algorithms in practical deployments.

2.4 Experimental Validation of the Importance of Potential

Figure 1 demonstrates our experimental results in the domains of ABR and scheduling (detailed experimental setup in Appendix A), validating the importance of tunability. Our findings reveal: 1) Algorithms based on simple logic, after appropriate tuning, can significantly outperform complex designs. In the ABR task, as shown in Figure 1(a), HYB(B) optimized through Bayesian optimization [38] and HYB(C) adjusted via `Crucible` reduced video playback stalling time by 92%. Figure 1(b) further indicates that these optimizations elevated the overall QoE ranking from the lowest position to fourth and first place, respectively. Similarly, in the scheduling problem, the simple FIFO algorithm tuned by `Crucible` achieves sub-optimal executor utilization as Figure 1(c) and the optimal cumulative waiting time as Figure 1(d). 2) Effective tuning encompasses not only parameter optimization but also improvements in logical structure. In the ABR task, while parameter-optimized HYB(B) still could not surpass the QoE performance of BOLA and RobustMPC, HYB(C) with logical adjustments through `Crucible` ultimately achieves the best QoE performance, thoroughly demonstrating the importance of algorithm logic adjustment.

3 Design

The overall design of `Crucible` can be seen in Figure 2. We introduce the design of `Crucible` from three aspects: Section 3.1 describes the workflow of the `Crucible` agent, Section 3.2 provides a formalized definition of potential, and Section 3.3 explains the interaction between `Crucible` and the control algorithms being evaluated.

3.1 Agent Workflow

Domain Knowledge Acquisition We inject domain knowledge into LLMs through system prompts across three dimensions: task description, optimization objectives, and environment overview. The task description defines the basic information of the control task, including the input states and the scope of optional output behaviors; the optimization objectives specify the improvement direction and evaluation criteria for the control algorithm; the environment overview provides key characteristics and constraints of the testing scenarios. These three dimensions construct a comprehensive knowledge framework, enabling LLMs to thoroughly understand the task context, optimization direction, and operational environmental characteristics.

Tool Utilization The potential of algorithms is primarily manifested in two key dimensions: first, the representational capacity of the algorithm, referring to the breadth and granularity of its control space—algorithms with higher control dimensions and greater precision exhibit richer performance variability in the hyperparameter space, thus possessing greater optimization potential [39, 40]; second, the comprehensibility [41, 42] of the algorithm—algorithms with higher structural transparency

enable developers to implement targeted functional enhancements through logical reconstruction. The former can be systematically improved through mature automatic optimization techniques, while the latter relies on the abstract understanding capabilities of LLMs. Therefore, we encapsulate optimization tools (such as Bayesian optimization)as standardized function interfaces to quantitatively evaluate the performance improvement of the current algorithmic logic within its hyperparameter space.

Action and Feedback Loop Drawing inspiration from optimization iteration patterns in industrial practice, we utilize historical adjustment records as an experiential foundation for subsequent optimization. Each algorithm modification is structurally preserved as a triplet including modification rationale, specific action, and observed results. Before a new round of optimization, these historical experiences are comprehensively presented to the LLM, enabling it to learn from previous attempts, avoid repeating mistakes, and simultaneously identify and replicate successful [43].

Differential Developer Capability Simulation To model developers with varying expertise and available resources, we simulate different capability levels by adjusting the computational budget of the agent, rather than by crafting different prompts. We primarily restrict agent capabilities along two dimensions: first, limiting the number of Bayesian optimization calls available for fine-grained parameter tuning; second, constraining the number of reflection iterations for breaking through algorithmic logical boundaries through systematic trial and error. This differential simulation, grounded in resource consumption, enables us to more realistically evaluate the practical value of algorithms under various tuning conditions.

3.2 Formalization of Potential

We formalize an algorithm's potential, $\mathcal{P}$, as its performance gain, weighted by a unified environmental distance metric that is derived from the performance profiles of a set of probe algorithms.

Performance Characteristic Vector. To quantify an environment's characteristics, we first define an evaluation set of environments $\mathcal{T}$. We then select a small set of representative probe algorithms. For any environment $E_k \in \mathcal{T}$, we run the n probe algorithms to obtain a raw performance score vector $[s_1(E_k), \ldots, s_n(E_k)]$. To eliminate dimensional effects, we normalize each score component across all environments in $\mathcal{T}$:

$$\text{norm}(s_j(E_k)) = \frac{s_j(E_k) - s_{j,\min}}{s_{j,\max} - s_{j,\min}}, \tag{1}$$

where $s_{j,\max} = \max_{E_k \in \mathcal{T}}(s_j(E_k))$ and $s_{j,\min} = \min_{E_k \in \mathcal{T}}(s_j(E_k))$. This process yields an n-dimensional normalized performance characteristic vector $V(E_k)$ for each environment E_k, which serves as its quantitative fingerprint. The vector is explicitly defined as:

$$V(E_k) = [\text{norm}(s_1(E_k)), \text{norm}(s_2(E_k)), \ldots, \text{norm}(s_n(E_k))]\,. \tag{2}$$

Unified Environment Distance and Similarity. Using these characteristic vectors, the distance between two environments E_i and E_t is defined as the root mean square error (RMSE):

$$\text{dis}(E_i, E_t) = \sqrt{\frac{1}{n}\sum_{j=1}^{n}(V(E_i)_j - V(E_t)_j)^2}. \tag{3}$$

Here, $V(E_i)_j$ denotes the j-th component of the vector $V(E_i)$.

The corresponding environment similarity is then defined as:

$$\text{sim}(E_i, E_t) = \max(0, 1 - \text{dis}(E_i, E_t)). \tag{4}$$

Tuning Potential Definition. With the environmental similarity metric now formally defined, we can present the complete definition of an algorithm's potential, $\mathcal{P}$. It is the similarity-weighted average performance gain across all test environments. For a given algorithm, let $S_{t,o}$ and $S_{t,c}$ be its original and Crucible-tuned performance in a test environment E_t, and let E_i be its ideal environment. The potential is calculated as:

$$\mathcal{P} = \frac{1}{|\mathcal{T}|}\sum_{E_t \in \mathcal{T}}\left[(S_{t,c} - S_{t,o}) \times \text{sim}(E_i, E_t)\right]. \tag{5}$$

This formulation ensures that performance gains in environments highly dissimilar to the ideal one are down-weighted, yielding a more robust and fair measure of an algorithm's intrinsic tunability.

3.3 Interaction Between `Crucible` and Control Algorithms

The `Crucible` framework supports multiple control tasks, thus designing a standardized interaction interface. In this interface, control algorithms provide two types of information to the LLM: the algorithm code itself and execution logs. The execution logs consist of a series of triplets containing states, actions, and results. Based on this information, the LLM exclusively modifies the control algorithm and obtains new test results by invoking the original execution file. This unified interface effectively standardizes the interaction logic, enabling `Crucible` to flexibly support any type of control algorithm.

The overall interaction logic of the system is as follows: First, the system traverses all preset test environments and executes the LLM optimization cycle in each environment. In this cycle, the system compares the performance of the current algorithm with reference algorithms (which can be overfit learning-based algorithms or theoretically optimal algorithms) and collects cases with significant performance differences. Subsequently, the LLM provides algorithm optimization suggestions based on these difference information. The system implements these suggestions and optionally applies Bayesian optimization methods to further adjust algorithm parameters. When all environments have been traversed, the system enters the evaluation phase. For each algorithm under test, it first determines its ideal environment (e.g., the one yielding the best performance). Then, using the unified performance-characteristic-based metric defined in Section 3.2, it calculates the similarity between each test environment and the ideal environment. Finally, it computes the algorithm's potential by aggregating the similarity-weighted performance gains across all test environments. Appendix B provides a detailed pseudocode of the complete Crucible workflow.

4 Evaluation

We begin by outlining our experimental setup in Section 4.1, with complete configuration details deferred to Appendix A. Our evaluation then unfolds in three progressive stages to systematically validate the `Crucible` framework.

First, in Section 4.2, we establish the framework's **effectiveness and generalizability**. We demonstrate its superiority over traditional tuning, validate its performance in a real-world deployment, and confirm its robustness across different LLMs. Building on this, Section 4.3 shows how `Crucible` moves beyond mere performance enhancement to provide **quantitative insights** into the abstract concept of algorithmic potential, revealing the key factors that govern it. Finally, in Section 4.4, we illustrate the **practical impact** of these insights, showing how they guide targeted algorithm design and establish potential as a valuable optimization target.

4.1 Experimental Setup

4.1.1 `Crucible`

This research mainly employs API calls to the Claude 3.7 Sonnet [44], with Bayesian optimization serving as a hyperparameter tuning tool. For simulating developers with varying optimization capabilities, we configure the Bayesian iteration count at three distinct levels (0, 10, and 20 iterations) while setting the reflection iteration steps to 1, 2, and 3, respectively. Our evaluation spans a diverse range of control tasks to demonstrate Crucible's generalizability.

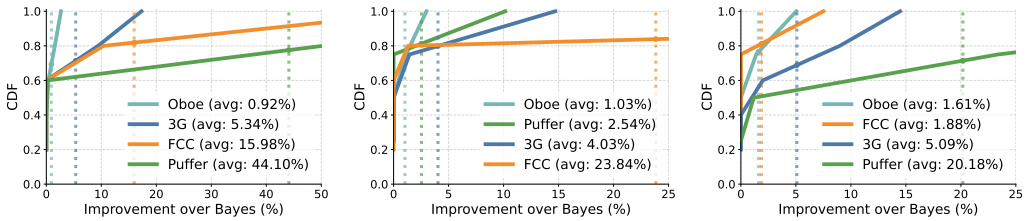
4.1.2 Case Studies

We select testbeds from three distinct domains to demonstrate Crucible's generalizability.

Classic Control. We use the "CartPole-v1" environment from Gym [45], a standard benchmark in control task. Our evaluation focuses on classic controllers such as Proportional-Integral-Derivative (PID) [46] and Linear-Quadratic Regulator (LQR) [47], comparing their optimized performance against RL-based methods DQN [48].

Table 1: Performance of classic control algorithms on Cart-Pole, tuned by `Crucible`.

Algorithm	Initial	After 1 Bayes	After 1 LLM	After 2 Bayes	After 2 LLM
Bang_bang	34	56	500	-	-
PID	34	77	110	271	500
LQR	161	500	-	-	-
DQN (Reference)	500	-	-	-	-



(a) CDF When Bayes=0 (b) CDF When Bayes=10 (c) CDF When Bayes=20

Figure 3: Optimization Space between Crucible and Bayes

Computer Systems (ABR). We utilize a widely adopted adaptive bitrate (ABR) simulator [32, 29] with four public network datasets: Oboe [30], FCC [49], 3G [50], and Puffer [51]. The experiments evaluate prominent algorithms, including BBA, MPC, HYB, BOLA, and Pitree, using the Envivio video trace [32] as standardized content. In addition to simulated environments, we conduct validation in a real-world setting using Dash.js [52].

Computer Systems (Scheduling). We employ a Spark simulator [53] with TPC-H query tasks [54] to evaluate classical and modern scheduling algorithms, including Shortest Job First (SJF), First-In-First-Out (FIFO), Multi-level feedback (MF), and Tetris [35].

4.2 Crucible's Effectiveness and Generalizability

4.2.1 Effectiveness: Expanding the Optimization Space

Unlike traditional optimization tools confined to hyperparameter tuning, `Crucible` implements modifications at the algorithmic logic level, expanding the optimization space. An illustration of this is found in the classic Cart-Pole control problem. As shown in Table 1, simple heuristic algorithms like Bang-bang [16] and PID, which initially perform limited, can achieve the optimal score of 500, matching the performance of a complex black-box algorithm like DQN. This improvement, particularly the jump from a score of 56 to 500 for the Bang-bang controller after a single LLM-driven logic modification, is a direct result of altering the core control logic—a performance leap unattainable through mere parameter tuning.

This principle of logic-level enhancement is not an isolated phenomenon but a general advantage that holds true in more complex domains. Across our computer system benchmarks, Figure 3 illustrates the relative improvement ratio achieved by `Crucible`'s enhanced results S_c compared to Bayesian optimization results S_b, expressed as $(S_c - S_b)/S_b$. The results demonstrate that `Crucible`'s logic-level improvements consistently yield performance enhancements, achieving gains of up to 44.1% on the Puffer dataset.

However, these logic-level modifications also introduce variability. Our experiments indicate that without the fine-tuning provided by Bayesian optimization, pure LLM-driven adjustments can be unreliable; approximately 60% of test scenarios show no significant performance gains, primarily attributable to the LLM's limitations in fine-grained numerical operations (Figure 3(a)). This highlights the critical synergy within our framework. As Bayesian optimization is incorporated, it not only improves the baseline but also effectively explores the new solution space opened up by the LLM's logic changes. Consequently, the proportion of ineffective test scenarios decreases from 60% to 20% (Figure 3(c)), confirming the powerful combination of logic-level exploration and parameter-level exploitation.

4.2.2 Real World Evaluation

We validate `Crucible` in a real-world ABR scenario using Dash.js over a public WiFi network. We test five heuristic algorithms and the RL-based Pensieve as a reference. The results, presented in Table 2, demonstrate that `Crucible` successfully enhances the performance of heuristic algorithms in this noisy, unpredictable environment. For example, the tuned HYB and BBA algorithms achieve a QoE score of 1.72, outperforming their original versions and even surpassing the RL-based Pensieve (1.66). Conversely, the complex and less interpretable Pitree algorithm exhibits no improvement, reinforcing that `Crucible`'s effectiveness correlates with an algorithm's comprehensibility. These findings provide evidence that the benefits of `Crucible`-guided tuning transfer directly to practical, real-world deployments.

Table 2: QoE of ABR in a real-world Dash.js deployment before and after `Crucible` tuning.

QoE State	**HYB**	**BBA**	**BOLA**	**Pitree**	**MPC**	**Pensieve (RL)**
Original	1.40	1.56	1.20	1.73	1.72	1.66
`Crucible`-Tuned	1.72	1.72	1.54	1.73	1.79	-

4.2.3 Robustness Across Different LLMs

We evaluate the framework using three different models: Claude 3.7 Sonnet (our primary model), the previous generation Claude 3.5 Sonnet, and GPT-4o-mini. As detailed in Table 3, while the absolute final performance varies slightly across models, the overall conclusions remain consistent. All LLMs effectively tune the ABR algorithms, and the relative performance ranking among them remains largely stable. Claude 3.7 Sonnet achieves a higher final score on the HYB algorithm, suggesting that more powerful models can unlock greater potential in certain cases, but other models also deliver significant improvements.

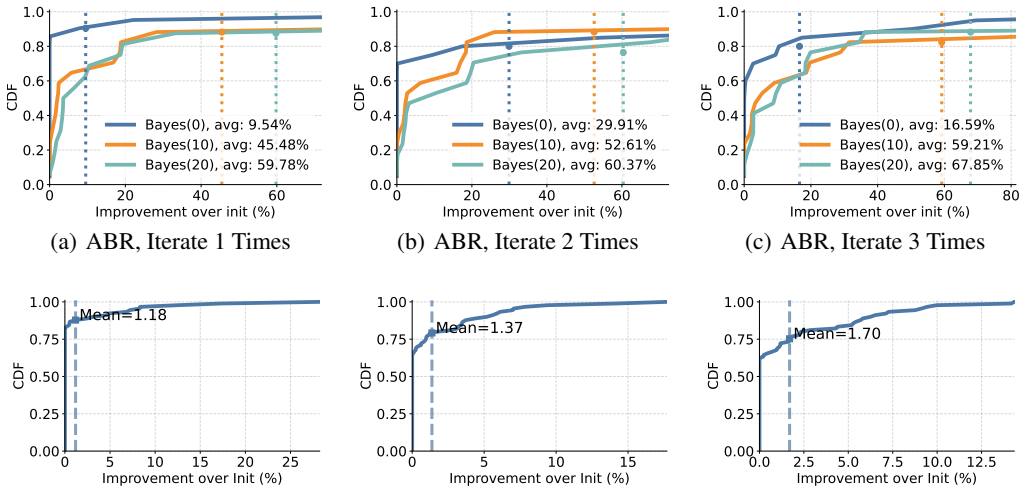
Table 3: Final ABR performance after tuning with different LLMs. All runs were configured with 3 iterations and 20 Bayesian iterations.

Algorithm	**Initial**	**Claude 3.5 Sonnet**	**GPT-4o-mini**	**Claude 3.7 Sonnet**
BBA	0.75	1.13	1.10	1.11
BOLA	1.02	1.07	1.08	1.06
HYB	0.92	1.03	1.04	1.12
Pitree	0.31	0.35	0.37	0.36
MPC	1.07	1.09	1.10	1.09

4.3 Potential Analysis

In Figure 4, we compare the percentage performance improvements of ABR and scheduling algorithms under different `Crucible` capability settings against their original scores. Regarding optimization tool usage constraints, Figures 4(a) through 4(c) demonstrate a positive correlation between increased Bayesian optimization iterations and significant algorithm performance enhancements. Specifically, when the number of iterations increased from one to two, a notable performance leap occurred—in ABR, the improvement rate with zero Bayesian iterations rose from 9.54% to 29.91%, while in scheduling algorithms, the proportion of scenarios failing to achieve performance improvements decreased from 80% to 60%. Between the two dimensions examined, the ability of optimization tools to unlock algorithmic logic potential has a more pronounced impact on performance enhancement. Furthermore, comparing ABR and scheduling algorithms reveals that due to differences in input state complexity (ABR only involves buffer size and bandwidth, whereas scheduling encompasses complex DAG graph information and node states), LLMs face varying degrees of comprehension challenges, resulting in significantly lower improvement magnitudes for scheduling algorithms compared to ABR algorithms.

We illustrate the detailed potential analysis results with ABR algorithms as an example in Table 4. Through our analysis, we derive two key findings: First, even among algorithms representing simple logic, the HYB algorithm utilizing a broader state space demonstrates significantly greater potential



(a) ABR, Iterate 1 Times (b) ABR, Iterate 2 Times (c) ABR, Iterate 3 Times

(d) Scheduling, Iterate 1 Times (e) Scheduling, Iterate 2 Times (f) Scheduling, Iterate 3 Times

Figure 4: Improvement in Different Ability

than the BBA algorithm using only a single state space, indicating that algorithmic representational capacity substantially influences potential enhancement; Second, as a decision tree algorithm distilled from deep reinforcement learning, Pitree exhibits lower optimization potential despite its initial limited performance, suggesting that the complex logical structure of decision trees may negatively impact potential enhancement. Therefore, researchers aspire to design ABR algorithms that improve comprehensibility, such as ComTree [14].

Table 4: Potential of ABR Algorithms

Alg	init	1 Iter	2 Iter	3 Iter	Impro	Potential	Ideal
HYB	0.92 ± 0.64	0.99 ± 0.61	1.03 ± 0.57	1.02 ± 0.60	0.10 ± 0.18	0.068 ± 0.117	FCC
Pitree	0.31 ± 0.10	0.33 ± 0.10	0.44 ± 0.27	0.38 ± 0.09	0.07 ± 0.12	0.033 ± 0.022	Puffer
BOLA	1.01 ± 0.45	1.05 ± 0.46	1.04 ± 0.45	1.05 ± 0.45	0.03 ± 0.06	0.025 ± 0.032	3G
BBA	0.75 ± 0.75	1.04 ± 0.59	0.98 ± 0.56	1.01 ± 0.57	0.26 ± 0.38	0.018 ± 0.008	Oboe
MPC	1.07 ± 0.52	1.10 ± 0.59	1.08 ± 0.53	1.09 ± 0.54	0.02 ± 0.04	0.017 ± 0.014	3G

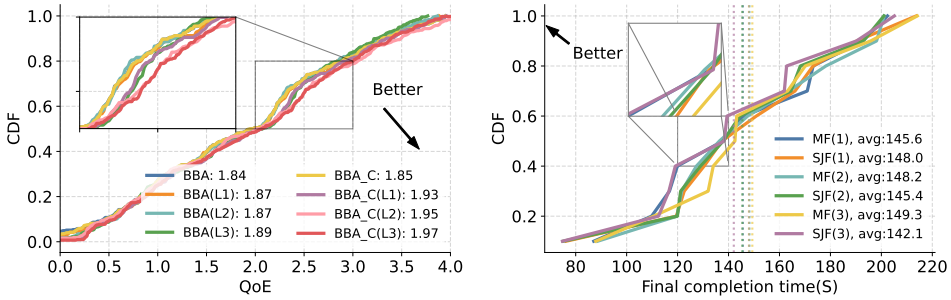
4.4 From Potential Assessment to Algorithm Optimization

This section explores how to transform `Crucible`'s potential assessments into effective algorithm optimization strategies to enhance ultimate performance.

4.4.1 Enhancing Algorithmic Representational Capacity

In the case of ABR algorithms, the results show that the BBA algorithm exhibits both lower optimization potential and inferior final performance compared to the HYB algorithm. This indicates inherent representational limitations in BBA's buffer-only control approach. Based on this insight, we improve BBA to create the BBA_C algorithm (detailed implementation in Appendix C). Specific enhancements include: incorporating current bandwidth as an additional control input and introducing a bandwidth-based control branch to select the highest bitrate that would not cause video stalling.

As shown in Figure 5(a), the enhanced BBA_C's initial performance closely resembles the original BBA algorithm, differing by only 0.5%. From a traditional evaluation perspective, these algorithms appear nearly equivalent. However, after optimization adjustments via `Crucible`, BBA_C's enhanced representational capacity advantage becomes evident. During each optimization iteration, BBA_C consistently outperforms BBA, ultimately achieving a 4% performance improvement. This case demonstrates how enhancing algorithmic representational capacity can increase optimization potential and ultimately improve performance.



(a) CDF of ABR QoE, Oboe, L1 means 1 iters (b) CDF of Scheduling FCT, 10*2G input Sizes

Figure 5: CDF After Optimization

4.4.2 Simplifying Algorithm Logic

Another important dimension involves improving developers' comprehension capabilities and optimization space by simplifying algorithmic logic. Figure 5(b) presents a typical case in task scheduling: initially, the Multilevel Feedback algorithm outperforms the SJF algorithm; however, after optimization adjustments through `Crucible`, the SJF algorithm achieves shorter task completion times than the supposedly more advanced Multilevel Feedback algorithm.

The fundamental cause of this phenomenon lies in the high-dimensional complexity of input states in DAG scheduling problems. This complexity makes it difficult for LLMs to comprehensively understand the internal logic of complex algorithms, thereby limiting their ability to add appropriate optimization logic to complex algorithms. In contrast, LLMs can more effectively understand and optimize foundational algorithms with simple, clear logic. This finding emphasizes the important value of "simplicity" in algorithm design, particularly in scenarios leveraging AI-assisted optimization.

5 Limitations and Broader Impacts

Limitations As the first work exploring the potential of control algorithms, this research has two main limitations. First, the stability issue: since we use an LLM as our foundation, different capabilities and versions of LLMs may influence the results. However, we believe this does not diminish the value of assessing algorithmic potential, as different LLMs essentially simulate developers with varying skill levels, making the evaluation, selection, and design of algorithms still practically meaningful. Second, we currently cannot directly modify the internal logic of black-box algorithms; therefore, in this paper, we discuss and analyze decision trees distilled from black-box algorithms. Effectively understanding and adjusting the internal logic of black-box algorithms remains an open challenge, providing direction for future research.

Broader Impacts We hope our work can inspire a rethinking of algorithm design, positioning potential as a new optimization direction or even as an optimization metric.

6 Conclusion

We introduced `Crucible`, a framework that addresses the gap between algorithm design and practical deployment by quantitatively evaluating Tuning Potential. `Crucible` leverages an LLM agent to simulate developer behavior and introduces a formalized potential metric to quantify this untapped optimization space. Our extensive evaluations—spanning classic control tasks, complex computer systems, and a real-world deployment—demonstrate that `Crucible` not only enhances algorithm performance beyond traditional methods but also establishes tuning potential as a valuable, quantifiable metric. This work advocates for a shift in algorithm design, treating Tuning Potential as a property to be evaluated from the outset, rather than as a post-deployment afterthought. This empowers designers to build adaptable algorithms that maintain long-term value in evolving real-world environments.

Acknowledgments We thank the anonymous NeurIPS reviewers for their constructive feedback, which has significantly improved this work exploring new optimization directions. This research was supported by the Beijing National Research Center for Information Science and Technology under Grant BNR2023TD03005-2, the NSERC Discovery Grant, and the Beijing Key Laboratory of Networked Multimedia.

References

[1] J. C. Doyle, B. A. Francis, and A. R. Tannenbaum, *Feedback control theory*. Courier Corporation, 2013.

[2] E. Yurtsever, J. Lambert, A. Carballo, and K. Takeda, "A survey of autonomous driving: Common practices and emerging technologies," *IEEE access*, vol. 8, pp. 58 443–58 469, 2020.

[3] S. G. Tzafestas, "Mobile robot control and navigation: A global overview," *Journal of Intelligent & Robotic Systems*, vol. 91, no. 1, pp. 35–58, 2018.

[4] B. Paden, M. Čáp, S. Z. Yong, D. Yershov, and E. Frazzoli, "A survey of motion planning and control techniques for self-driving urban vehicles," *IEEE Transactions on intelligent vehicles*, vol. 1, no. 1, pp. 33–55, 2016.

[5] Y. Sani, A. Mauthe, and C. Edwards, "Adaptive bitrate selection: A survey," *IEEE Communications Surveys & Tutorials*, vol. 19, no. 4, pp. 2985–3014, 2017.

[6] L. Jia, C. Zhou, T. Huang, C. Li, and L. Sun, "Rdladder: Resolution-duration ladder for vbr-encoded videos via imitation learning," in *IEEE INFOCOM 2023-IEEE Conference on Computer Communications*. IEEE, 2023, pp. 1–10.

[7] L. Jia, T. Huang, and L. Sun, "Zixia: A reinforcement learning approach via adjusted ranking reward for internet congestion control," in *ICC 2022-IEEE International Conference on Communications*. IEEE, 2022, pp. 365–370.

[8] L. Jia, C. Zhou, C. Li, J. Liu, and L. Sun, "Towards user-level qoe: Large-scale practice in personalized optimization of adaptive video streaming," in *Proceedings of the ACM SIGCOMM 2025 Conference*, 2025, pp. 1154–1166.

[9] L. Jia, C. Zhou, T. Huang, C. Li, and L. Sun, "Meet challenges of rtt jitter, a hybrid internet congestion control algorithm," in *Proceedings of the ACM Web Conference 2024*, 2024, pp. 2768–2776.

[10] M. Kumar, S. C. Sharma, A. Goel, and S. P. Singh, "A comprehensive survey for scheduling techniques in cloud computing," *Journal of Network and Computer Applications*, vol. 143, pp. 1–33, 2019.

[11] D. E. Kirk, *Optimal control theory: an introduction*. Courier Corporation, 2004.

[12] U. Mackenroth, *Robust control systems: theory and case studies*. Springer Science & Business Media, 2013.

[13] D. M. Hamby, "A review of techniques for parameter sensitivity analysis of environmental models," *Environmental monitoring and assessment*, vol. 32, pp. 135–154, 1994.

[14] L. Jia, C. Li, Z. Yuan, J. Chen, T. Huang, J. Liu, and L. Sun, "Beyond interpretability: Exploring the comprehensibility of adaptive video streaming through large language models," *arXiv preprint arXiv:2508.16448*, 2025.

[15] A. Berenzweig, B. Logan, D. P. Ellis, and B. Whitman, "A large-scale evaluation of acoustic and subjective music-similarity measures," *Computer Music Journal*, pp. 63–76, 2004.

[16] R. S. Sutton, A. G. Barto *et al.*, *Reinforcement learning: An introduction*. MIT press Cambridge, 1998, vol. 1, no. 1.

[17] R. Nakano, J. Hilton, S. Balaji, J. Wu, L. Ouyang, C. Kim, C. Hesse, S. Jain, V. Kosaraju, W. Saunders *et al.*, "Webgpt: Browser-assisted question-answering with human feedback," *arXiv preprint arXiv:2112.09332*, 2021.

[18] J. Achiam, S. Adler, S. Agarwal, L. Ahmad, I. Akkaya, F. L. Aleman, D. Almeida, J. Altenschmidt, S. Altman, S. Anadkat *et al.*, "Gpt-4 technical report," *arXiv preprint arXiv:2303.08774*, 2023.

[19] J. Kaplan, S. McCandlish, T. Henighan, T. B. Brown, B. Chess, R. Child, S. Gray, A. Radford, J. Wu, and D. Amodei, "Scaling laws for neural language models," *arXiv preprint arXiv:2001.08361*, 2020.

[20] W. X. Zhao, K. Zhou, J. Li, T. Tang, X. Wang, Y. Hou, Y. Min, B. Zhang, J. Zhang, Z. Dong *et al.*, "A survey of large language models," *arXiv preprint arXiv:2303.18223*, 2023.

[21] J. S. Park, J. O'Brien, C. J. Cai, M. R. Morris, P. Liang, and M. S. Bernstein, "Generative agents: Interactive simulacra of human behavior," in *Proceedings of the 36th Annual ACM Symposium on User Interface Software and Technology*, 2023, pp. 1–22.

[22] G. V. Aher, R. I. Arriaga, and A. T. Kalai, "Using large language models to simulate multiple humans and replicate human subject studies," in *International Conference on Machine Learning*. PMLR, 2023, pp. 337–371.

[23] Z. Hussain, M. Binz, R. Mata, and D. U. Wulff, "A tutorial on open-source large language models for behavioral science," *Behavior Research Methods*, vol. 56, no. 8, pp. 8214–8237, 2024.

[24] R. Ren, P. Qiu, Y. Qu, J. Liu, W. X. Zhao, H. Wu, J.-R. Wen, and H. Wang, "Bases: Large-scale web search user simulation with large language model based agents," *arXiv preprint arXiv:2402.17505*, 2024.

[25] Z. Zhao, W. Fan, J. Li, Y. Liu, X. Mei, Y. Wang, Z. Wen, F. Wang, X. Zhao, J. Tang *et al.*, "Recommender systems in the era of large language models (llms)," *arXiv preprint arXiv:2307.02046*, 2023.

[26] X. Huang, J. Lian, Y. Lei, J. Yao, D. Lian, and X. Xie, "Recommender ai agent: Integrating large language models for interactive recommendations," *arXiv preprint arXiv:2308.16505*, 2023.

[27] T.-Y. Huang, R. Johari, N. McKeown, M. Trunnell, and M. Watson, "A buffer-based approach to rate adaptation: Evidence from a large video streaming service," in *Proceedings of the 2014 ACM conference on SIGCOMM*, 2014, pp. 187–198.

[28] K. Spiteri, R. Urgaonkar, and R. K. Sitaraman, "Bola: Near-optimal bitrate adaptation for online videos," *IEEE/ACM transactions on networking*, vol. 28, no. 4, pp. 1698–1711, 2020.

[29] X. Yin, A. Jindal, V. Sekar, and B. Sinopoli, "A control-theoretic approach for dynamic adaptive video streaming over http," in *Proceedings of the 2015 ACM Conference on Special Interest Group on Data Communication*, 2015, pp. 325–338.

[30] Z. Akhtar, Y. S. Nam, R. Govindan, S. Rao, J. Chen, E. Katz-Bassett, B. Ribeiro, J. Zhan, and H. Zhang, "Oboe: Auto-tuning video abr algorithms to network conditions," in *Proceedings of the 2018 Conference of the ACM Special Interest Group on Data Communication*, 2018, pp. 44–58.

[31] Z. Meng, J. Chen, Y. Guo, C. Sun, H. Hu, and M. Xu, "Pitree: Practical implementation of abr algorithms using decision trees," in *Proceedings of the 27th ACM International Conference on Multimedia*, 2019, pp. 2431–2439.

[32] H. Mao, R. Netravali, and M. Alizadeh, "Neural adaptive video streaming with pensieve," in *Proceedings of the Conference of the ACM Special Interest Group on Data Communication*, 2017, pp. 197–210.

[33] M. Zaharia, D. Borthakur, J. Sen Sarma, K. Elmeleegy, S. Shenker, and I. Stoica, "Delay scheduling: a simple technique for achieving locality and fairness in cluster scheduling," in *Proceedings of the 5th European conference on Computer systems*, 2010, pp. 265–278.

[34] M. Isard, V. Prabhakaran, J. Currey, U. Wieder, K. Talwar, and A. Goldberg, "Quincy: fair scheduling for distributed computing clusters," in *Proceedings of the ACM SIGOPS 22nd symposium on Operating systems principles*, 2009, pp. 261–276.

[35] R. Grandl, G. Ananthanarayanan, S. Kandula, S. Rao, and A. Akella, "Multi-resource packing for cluster schedulers," *ACM SIGCOMM Computer Communication Review*, vol. 44, no. 4, pp. 455–466, 2014.

[36] R. Zhang, C. Yang, X. Wang, T. Huang, C. Wu, J. Liu, and L. Sun, "Practical cloud-edge scheduling for large-scale crowdsourced live streaming," *IEEE Transactions on Parallel and Distributed Systems*, vol. 34, no. 7, pp. 2055–2071, 2023.

[37] S. Shen, V. Van Beek, and A. Iosup, "Statistical characterization of business-critical workloads hosted in cloud datacenters," in *2015 15th IEEE/ACM international symposium on cluster, cloud and grid computing*. IEEE, 2015, pp. 465–474.

[38] J. Snoek, H. Larochelle, and R. P. Adams, "Practical bayesian optimization of machine learning algorithms," *Advances in neural information processing systems*, vol. 25, 2012.

[39] T. Yu and H. Zhu, "Hyper-parameter optimization: A review of algorithms and applications," *arXiv preprint arXiv:2003.05689*, 2020.

[40] M. Črepinšek, S.-H. Liu, and M. Mernik, "Exploration and exploitation in evolutionary algorithms: A survey," *ACM computing surveys (CSUR)*, vol. 45, no. 3, pp. 1–33, 2013.

[41] A. A. Freitas, "Comprehensible classification models: a position paper," *ACM SIGKDD explorations newsletter*, vol. 15, no. 1, pp. 1–10, 2014.

[42] D. Martens, J. Vanthienen, W. Verbeke, and B. Baesens, "Performance of classification models from a user perspective," *Decision Support Systems*, vol. 51, no. 4, pp. 782–793, 2011.

[43] J. Wei, X. Wang, D. Schuurmans, M. Bosma, F. Xia, E. Chi, Q. V. Le, D. Zhou *et al.*, "Chain-of-thought prompting elicits reasoning in large language models," *Advances in neural information processing systems*, vol. 35, pp. 24 824–24 837, 2022.

[44] Anthropic, "Claude 3.7 sonnet," https://www.anthropic.com/claude/sonnet, 2025, accessed May 1, 2025.

[45] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba, "Openai gym," *arXiv preprint arXiv:1606.01540*, 2016.

[46] M. J. Willis, "Proportional-integral-derivative control," *Dept. of Chemical and Process Engineering University of Newcastle*, vol. 6, 1999.

[47] A. Bemporad, M. Morari, V. Dua, and E. N. Pistikopoulos, "The explicit linear quadratic regulator for constrained systems," *Automatica*, vol. 38, no. 1, pp. 3–20, 2002.

[48] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski *et al.*, "Human-level control through deep reinforcement learning," *nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[49] M. F. B. Report, "Raw data measuring broadband america 2016," https://www.fcc.gov/reports-research/reports/measuring-broadband-america/raw-data-measuring-broadband-america-2016, 2016, [Online; accessed 19-July-2016].

[50] H. Riiser, P. Vigmostad, C. Griwodz, and P. Halvorsen, "Commute path bandwidth traces from 3g networks: Analysis and applications," in *Proceedings of the 4th ACM Multimedia Systems Conference*, 2013, pp. 114–118.

[51] F. Y. Yan, H. Ayers, C. Zhu, S. Fouladi, J. Hong, K. Zhang, P. A. Levis, and K. Winstein, "Learning in situ: a randomized experiment in video streaming." in *NSDI*, vol. 20, 2020, pp. 495–511.

[52] DASH Industry Forum, "dash.js," https://dashjs.org/, 2024, [Online; accessed 15-February-2025]. [Online]. Available: https://dashjs.org/

[53] H. Mao, M. Schwarzkopf, S. B. Venkatakrishnan, Z. Meng, and M. Alizadeh, "Learning scheduling algorithms for data processing clusters," in *Proceedings of the ACM special interest group on data communication*, 2019, pp. 270–288.

[54] Transaction Processing Performance Council, "Tpc-h: A decision support benchmark," https://www.tpc.org/tpch/, 2025, accessed May 1, 2025.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: As shown in the abstract and introduction.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: We discuss limitations as Section 5

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [NA]

Justification: The paper does not include theoretical results.

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: Our code is available in the supplementary materials.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: Our code is available in the supplementary materials.

Guidelines:

- The answer NA means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

   Answer: [Yes]

   Justification: We introduce our experimental setup at Section 4.1 and Appendix A

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: As shown in Table 4, we report standard error, and the main results are displayed using CDFs.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).

- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: As described in Section 4.1, we introduce the token API.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: The research fully complies with the NeurIPS Code of Ethics.

   Guidelines:

   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: We discuss broader impacts as Section 5

    Guidelines:

    - The answer NA means that there is no societal impact of the work performed.
    - If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.

- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

    Answer: [NA]

    Justification: The paper poses no such risks.

    Guidelines:

    - The answer NA means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: Assets used in this paper comply with Licenses and Terms of Use.

    Guidelines:

    - The answer NA means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.

- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: Our code is available in the supplementary materials.

    Guidelines:

    - The answer NA means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [NA]

    Justification: We did not perform crowdsourcing experiments.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [NA]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does not impact the core methodology, scientific rigorousness, or originality of the research, declaration is not required.

Answer: [Yes]

Justification: We introduce the usage of LLMs as Section 3.

Guidelines:

- The answer NA means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy (`https://neurips.cc/Conferences/2025/LLM`) for what should or should not be described.

Dataset	Traces	Avg. BW (Mbit/s)	BW Range (Mbit/s)
3G	142	1.52 ± 0.72	[0.60, 4.59]
Oboe	428	2.77 ± 1.32	[0.34, 5.70]
FCC	264	1.33 ± 0.55	[0.19, 3.43]
Puffer	500	1.60 ± 0.88	[0.30, 3.60]

Table 5: Details of Network Trace Datasets

Appendix

A Details Experimental Setup

A.1 ABR Experimental Setup

A.1.1 Network Traces

We summarize the characteristics of the four public network traces utilized in this study in Table 5, including the number of traces, average bandwidth per trace, and bandwidth range.

A.1.2 Video Samples

We use the "EnvivoDash3" video from the "MPEG-DASH reference videos," consistent with previous works [32, 29]. The video is 193 seconds in length and segmented into 4-second chunks with bitrates of {300, 750, 1200, 1850, 2850, 4300} kbps.

A.1.3 Player Configuration

We employ a classical VoD scenario player simulator as used in [32, 29], with an initial default video quality of 750 kbps.

A.1.4 Quality of Experience (QoE) Metrics

For the video-on-demand scenario, we adopt the classical QoE_{lin} as our QoE metric [32, 29]. This model is defined as follows:

$$QoE_{lin} = \sum_{n=1}^{N} q(R_n) - \mu \sum_{n=1}^{N} T_n - \sum_{n=1}^{N-1} |q(R_{n+1}) - q(R_n)| \tag{6}$$

where N represents the total number of video segments, R_n denotes the bitrate of the n-th segment, and $q(R_n)$ is a function mapping bitrate to perceived user quality. T_n represents the buffering time incurred while downloading the n-th segment, and μ is the weight coefficient for buffering time. The final term penalizes variations in video quality to ensure playback smoothness. Specifically, in our implementation, we set μ to 4.3.

A.2 Scheduling Experimental Setup

A.2.1 Spark Simulator

We utilize a high-fidelity simulator [53] to evaluate the performance of our scheduling policies. This simulator is designed to faithfully emulate the behavior of a real Spark cluster by capturing several key real-world phenomena:

1. **Initial Task Wave Effects**: The first wave of tasks in a particular stage often exhibits slower execution compared to subsequent tasks. This slowdown arises from factors such as Spark's pipelined task execution, just-in-time (JIT) compilation of task code, and initial overheads like establishing TCP connections between executors. The simulator accounts for this by

drawing the runtime of first-wave tasks from a separate distribution distinct from that of later waves.

2. **Executor Startup Delays**: Adding an executor to a Spark job involves starting a new JVM process, which typically incurs a delay of 2–3 seconds. To accurately reflect this behavior, the simulator imposes a startup delay whenever an executor is reallocated across jobs, mirroring the real-world costs associated with executor initialization.
3. **Impact of High Parallelism**: High degrees of parallelism can negatively impact the performance of individual tasks. Wider shuffle operations require more TCP connections and introduce additional computational overhead when merging data from a large number of shards. The simulator captures these effects by sampling task durations from distributions corresponding to different levels of parallelism, provided such data is available.

Through these mechanisms, the simulator effectively replicates the dynamic behavior of a real-world Spark cluster. This enables rigorous testing and validation of scheduling policies under realistic conditions, ensuring that the results are representative of practical deployment scenarios.

A.2.2 Workload Setup

For the input workload, we use 2GB of data consisting of 10 TPC-H standard query tasks [54]. TPC-H is a benchmark suite widely used to evaluate decision support systems. It consists of complex analytical queries that reflect real-world workloads in distributed data processing environments. Each query involves a combination of computations, such as joins, aggregations, and data filtering, which are represented as Directed Acyclic Graphs (DAGs) of tasks in the Spark environment. The DAG structure introduces dependencies between tasks, making it ideal for testing the efficacy of advanced scheduling strategies.

B `Crucible` Pseudocode

The detailed interaction logic between `Crucible` and the control algorithm is illustrated in Algorithm 1.

C Optimization for BBA

We modify BBA to incorporate two control logics based on bandwidth and buffer occupancy. The bandwidth-based control selects the maximum bitrate that does not cause stalling, while the original buffer-based logic selects its own bitrate. The algorithm then chooses the smaller of these two bitrates. The detailed pseudocode can be found in 2.

Algorithm 1: `Crucible` Algorithm

Input: Algorithm Alg_{cur}, Reference algorithm Alg_{ref}, Number of LLM adjustments N_{llm}, Number of Bayesian optimizations N_{BO}, Test environments $Envs$

Output: Potentials

Function `ApplyBayesianOptimization`(Alg, N_{BO}, E)**:**
  if $N_{BO} > 0$ **then**
    $params, ranges = LLMIdentifyParameters(Alg)$
    $Alg, score = BO(Alg, params, ranges, N_{BO}, E)$
  else
    $score = E(Alg)$
  return $Alg, score$

for $E^i \in Envs$ **do**
  $Alg^i_{cur} = Alg_{test}, score^i_{init} = E^i(Alg^i_{cur}), score^i_{cur} = score^i_{init}$
  $Alg^i_{cur}, score^i_{cur} =$ `ApplyBayesianOptimization`$(Alg^i_{cur}, N_{BO}, E^i)$
  $BadCases^i = \{\}$
  for $j = 1$ **to** N_{llm} **do**
    /* Compare with reference algorithm and collect bad cases */
    $newBadCases^i = CompareAlgs(Alg^i_{cur}, Alg_{ref}, E^i)$
    $BadCases^i = BadCases^i \cup newBadCases^i$
    /* Get optimization suggestions from LLM */
    $suggestions^i, reasons^i = LLMOptimizationSuggestions(Alg^i_{cur}, BadCases^i)$
    /* Apply optimization suggestions */
    $Alg^i_{new} = ApplySuggestions(Alg^i_{cur}, suggestions^i)$
    $Alg^i_{new}, score^i_{new} =$ `ApplyBayesianOptimization`$(Alg^i_{new}, N_{BO}, E^i)$
    if $score^i_{new} > score^i_{cur}$ **then**
      /* Score comparison and update */
      $Alg^i_{cur} = Alg^i_{new}$
      $score^i_{cur} = score^i_{new}$

/* Find ideal environment and calculate potentials */
$Potentials = \{\}$
$E_{best} =$ null
$Potential_{max} = 0$
for $E^i \in Envs$ **do**
  $E_i = GetIdealEnv(Alg_{cur}, Envs)$
  $D^i = GetDistance(E^i, E_i)$
  $Potential^i = \frac{score^i_{cur} - score^i_{init}}{D^i}$
  $Potentials = Potentials \cup \{Potential^i\}$
return $Potentials$

Algorithm 2: `BBA_C` Algorithm

Input: Current buffer size $buffer$, chunk length $chunk\_len$, download speed $speed$, video bitrate array $bitrates$, dimension dim

Output: Selected bitrate index $rate$

$cushion \leftarrow 10$ $reservoir \leftarrow$ predefined value $beta \leftarrow 0.95$

/* Calculate bandwidth-based bitrate */

$bw\_rate \leftarrow 0$ **for** $i = dim - 1, dim - 2, \ldots, 0$ **do**

  if $bitrates[i] \times chunk\_len < beta \times speed \times buffer$ **then**

    $bw\_rate \leftarrow i$ **break**

/* Calculate buffer-based bitrate */

$buf\_rate \leftarrow 0$ **if** $buffer < reservoir$ **then**

  $buf\_rate \leftarrow 0$

else

  if $buffer \geq reservoir + cushion$ **then**

    $buf\_rate \leftarrow dim - 1$

  else

    $buf\_rate \leftarrow \lfloor (dim - 1) \times \frac{buffer - reservoir}{cushion} \rfloor$

/* Choose the minimum of both bitrates */

$rate \leftarrow \min(bw\_rate, buf\_rate)$

$rate \leftarrow \max(0, \min(rate, dim - 1))$

return $rate$