PRELIMINARY PREPRINT VERSION – DO NOT CITE
This is not the version of record. The final version will appear in the Proceedings of IJCAI-ECAI 2026.

CDS4RAG: Cyclic Dual-Sequential Hyperparameter Optimization for RAG

Pengzhou Chen[1*], **Tao Chen**[2†]
[1]School of Computer Science and Engineering, UESTC, Chengdu, China
[2]IDEAS Lab, University of Birmingham, Birmingham, UK
t.chen@bham.ac.uk

Abstract

Retrieval-Augmented Generation (RAG) is sensitive to the vast hyperparameters of the retriever and generator, yet optimizing them using given queries is a challenging task due to the complex interactions and expensive evaluation costs. Existing algorithms are ineffective and slow in convergence, since they often treat RAG as a monolithic black box or only optimize partial hyperparameters. In this paper, we propose CDS4RAG, a framework that optimizes the full RAG hyperparameters using given queries via a new cyclic dual-sequential formulation. CDS4RAG is special in the sense that it distinguishes the hyperparameters of the retriever and generator, cyclically optimizing them in turn. Such a paradigm allows us to design fine-grained within-cycle budget provision and expedite the optimization via cross-cycle seeding when optimizing the generator. CDS4RAG is also an algorithm-agnostic framework that can be paired with diverse general algorithms. Through experiments on four common benchmarks and two backbone LLMs, we reveal that CDS4RAG considerably boosts the vanilla algorithms in 21/24 cases while significantly outperforming state-of-the-art algorithms in all cases with up to $1.54\times$ improvements of generation quality and better speedup.

1 Introduction

Retrieval-Augmented Generation (RAG) has emerged as a dominant paradigm for enhancing Large Language Models (LLMs) based on external knowledge [Qiao *et al.*, 2025; Guo *et al.*, 2025], serving as a foundation in many AI applications [Mridul *et al.*, 2025; Niu *et al.*, 2024]. As from Figure 1, typically the RAG contains a retriever (i.e., the retrieval stage linked with a vector/graph database) and a generator (i.e., the generation stage by a LLM): for a given batch of queries, the retriever retrieves relevant contexts that consolidate the query prompts for the generator to produce better answers.

Like many other systems [Chen *et al.*, 2018; Chen *et al.*, 2025; Ye *et al.*, 2026], the effectiveness of RAG in handling

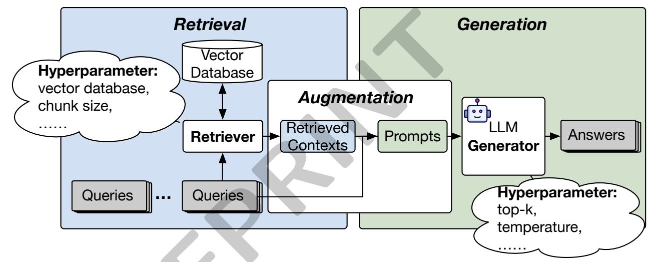


Figure 1: General workflow and hyperparameters of RAG.

queries highly depends on its hyperparameter values [Fu *et al.*, 2024], i.e., those for the retrieval (e.g., chunk size) and the generation stage (e.g., generation temperature), thus the their optimization is crucial. However, optimizing RAG hyperparameters using given queries is challenging, because:

- With the combined retriever and generator in RAG, the dimension of tunable hyperparameters increases dramatically with more complex interactions, making optimization difficult [Barker *et al.*, 2025; Stefano *et al.*, 2024].
- Evaluating the quality of a hyperparameter configurations as the feedback under full retrieval-generation round can be expensive, e.g., configuration evaluation on a RAG with Llama-3.1-8B over the *Agriculture* benchmark can take up to 30 minutes [Guo *et al.*, 2025].

To that end, many algorithms have been proposed. For example, there exist general algorithms for tuning hyperparameters [Bergstra *et al.*, 2011; Cowen-Rivers *et al.*, 2022]; specific ones such as AutoRAG-HP [Fu *et al.*, 2024] and METIS [Ray *et al.*, 2025] for RAG that focus on the entire retrieval-generation process as a whole; as well as others that solely perform hyperparameter optimization/HPO related to the prompt or the retriever of RAG [Agrawal *et al.*, 2025; Jeong *et al.*, 2024]. Yet, those algorithms are limited in the sense that they merely treat RAG as a monolithic black box or restrict themselves only to the hyperparameters at a certain stage. This ignores the highly structural internal information of RAG, causing slow convergence and devastating results.

In this paper, we take a different perspective to bridge these gaps: drawing on our understandings and empirical experiments, we observe that the retriever and generator in RAG are interrelated but also partially separable, each plays dif-

[*]Pengzhou Chen is also supervised in the IDEAS Lab.
[†]Tao Chen is the corresponding author.

ferent roles that determine the final generation outcomes of RAG. As such, we hypothesize that *"if the hyperparameters of retriever and generator can be optimized loosely, in a way that they maintain merely the basic interactions, then it should yield better results using less budget."* Thus, we propose CDS4RAG, a cyclic dual-sequential framework that heuristically optimizes RAG's hyperparameters using given queries. What make CDS4RAG special is that it distinguishes the retriever and generator in RAG, optimizing their corresponding hyperparameters sequentially with different objectives, but doing so in a cyclic manner[1], permitting a fine-grained budget provision and seeding[2] therein. This allows CDS4RAG to **focus on the sliced hyperparameter space**, hence mitigating the issue of complex hyperparameter interactions, while **relieving the optimization from evaluating the costly full retrieval-generation** via a good proportion of retrieval-only evaluations.

Since the cyclic dual-sequential hyperparameter optimization is compatible with any general algorithms, CDS4RAG is also algorithm-agnostic. Specifically, our contributions are:

- We newly reformulate the RAG hyperparameter optimization as a cyclic dual-sequential problem.
- We present a way to sequentially optimize the hyperparameters of the retriever and generator for retrieval and generation quality, respectively, in a cyclic manner.
- To achieve fine-grained control over the budget provisioning of hyperparameter optimization between retrieval and generation within a cycle, we propose an adaptive mechanism that stops optimizing the retriever when it detects a retrieval plateau, while capping that for the generator with a fixed amount of evaluations. This reduces the costly full retrieval-generation evaluations needed.
- To expedite the optimization with a warm-start, we design a cross-cycle seeding, such that promising generation hyperparameter configurations in previous cycles, based on knowledge retrieved under different retrieval configurations, are reused in the current cycle for the generator.

Experimental results on four common benchmarks and two backbone LLMs reveal that CDS4RAG can considerably boost general algorithms in 21/24 cases while significantly outperforms other state-of-the-art hyperparameter optimization algorithms for RAG in all cases with quality improvement of up to $1.54\times$ and better speedup. All data, code, and appendices can be accessed at: https://github.com/ideas-labo/cds4rag.

2 Related Work

2.1 General Hyperparameter Optimization

Over the decades, general algorithms like Bayesian Optimization (BO) [Snoek *et al.*, 2012] and its variants [Snoek *et al.*, 2012; Bergstra *et al.*, 2011; Jiang *et al.*, 2024; Chen and Chen, 2026; Li *et al.*, 2020] have been proposed for hyperparameter optimization. Those methods rely on a surrogate model [Gong *et al.*, 2025; Xiang *et al.*, 2026; Gong and Chen, 2023; Gong and Chen, 2024] to predict the next evaluated configuration, balancing the exploration and exploitation with reduced evaluation cost. State-of-the-art algorithms like HEBO [Cowen-Rivers *et al.*, 2022] have also been developed to handle more complex and noisy search spaces robustly. Other meta-heuristics without a model also exist [Xiong and Chen, 2025; Chen *et al.*, 2024; Chen and Li, 2021; Chen and Li, 2024; Ye *et al.*, 2025; Chen, 2022].

However, the above often do not use any specific knowledge/information of the algorithm/system to tune, which can waste the valuable information of the known structure in RAG.

2.2 Full Hyperparameter Optimization for RAG

There are algorithms designed/tailored for optimizing RAG hyperparameters using given queries. Among others, RayTune [Liaw *et al.*, 2018] tailors the general black-box optimization algorithms TPE to optimize the RAG hyperparameters. AutoRAG-HP [Fu *et al.*, 2024] utilizes Hierarchical Multi-Armed Bandits to optimize the RAG hyperparameters. While effective for discrete/combinatorial hyperparameters, it struggles with continuous ones. METIS [Ray *et al.*, 2025] is a multi-objective optimization approach to balance quality and latency in RAG, but it can struggle under high-dimensionality.

Yet, again those works often treat the RAG pipeline as a monolithic black box, relying on the evaluation of the full retrieval-generation rounds throughout the optimization.

2.3 Partial Hyperparameter Optimization for RAG

A parallel thread of research focuses on optimizing the hyperparameters of a specific stage for RAG, e.g., GEPA [Agrawal *et al.*, 2025] and MIPROv2 [Opsahl-Ong *et al.*, 2024] use LLMs to iteratively refine instructions for the prompt at the generator, but overlook the hyperparameters of the retriever (e.g., chunk size) that dictate the underlying context. In contrast, methods like AdaptiveRAG [Jeong *et al.*, 2024] dynamically adjust retrieval intensity only (e.g., top-k) based on query complexity. Other studies justify the importance of hyperparameters at diverse stages in RAG [Orbach *et al.*, 2025].

However, focusing on a single stage in RAG hyperparameter optimization can miss the valuable configurations that can only be found when different stages are considered together. Yet, the complex interaction and expensive evaluation make a direct joint optimization complex. This work tackles exactly such via cyclic dual-sequential hyperparameter optimization.

3 Problem Formulation and Motivation

3.1 Classic RAG Hyperparameter Optimization

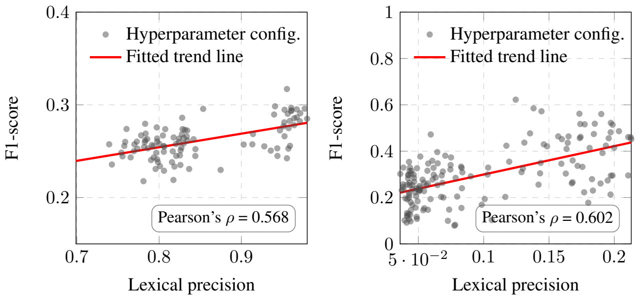
A hyperparameter in RAG can be a continuous, enumerated, or categorical variable. Traditionally, hyperparameter optimization for RAG works in a combined hyperparameter space of configurations $\Omega = \Phi \times \Theta$ from that of the retriever Φ and generator Θ. Formally, given a query batch $q \in \mathcal{Q}$ and a knowledge corpus $\mathcal{D}$, this can be expressed as:

$$\arg\max_{\boldsymbol{\omega} \in \Omega} [\mathbf{M}(f(\mathcal{Q}, \mathcal{D}, \boldsymbol{\omega}), \mathcal{G}^*)] \tag{1}$$

whereby f is the RAG system; $\mathbf{M}$ is the end-to-end performance metric over $\mathcal{Q}$; $\mathcal{G}^*$ is the ground truth for answering all

[1] A cycle means the completion of one sequence of optimizing retriever and then generator.

[2] Seeding means initialize an optimization with pre-defined hyperparameter configurations to start working with, enabling a warm-start.



(a) Qwen-8B on *Agriculture* (b) Llama-8B on *HotpotQA*

Figure 2: Exampled correlations between the quality of retrieval and generation over different RAG hyperparameter configurations under some queries. More cases can be found at Appendix A.

the queries. The goal is to jointly find the best hyperparameter configuration for both retriever and generator using $\mathcal{Q}$, hence $\mathbf{M}$, e.g., the F1-score of the generated answers, is maximized.

Albeit straightforward and widely adopted in existing work, this formulation forces one to work on the full combined space of retriever and generator under the costly full retrieval-generation evaluations, which can be slow and ineffective [Orbach *et al.*, 2025; Fu *et al.*, 2024; Barker *et al.*, 2025].

3.2 Cyclic Dual-Sequential Hyperparameter Optimization for RAG

While we can intuitively understand that the retriever and generator are two independent components in RAG, there is also stronger empirical evidence to motivate this work. From Figure 2, we see that the quality of contexts retrieved and answers generated under some queries, i.e., lexical precision and F1-score, commonly exhibit good positively monotonic correlation, which is also evidenced by prior studies [Salemi and Zamani, 2024; Orbach *et al.*, 2025]. Further, we found that the evaluation cost of retrievers is often $\approx 50\%$ cheaper than that of the generator. This motivates us to reformulate Equation 1 as:

$$\arg \max_{(\boldsymbol{\phi},\boldsymbol{\theta})\in\{(\boldsymbol{\phi}_1^*,\boldsymbol{\theta}_1^*),\ldots,(\boldsymbol{\phi}_T^*,\boldsymbol{\theta}_T^*)\}} [\mathbf{M}(f(\mathcal{Q},\mathcal{D},\boldsymbol{\phi},\boldsymbol{\theta}),\mathcal{G}^*)] \quad (2a)$$

$$\text{s.t. } \boldsymbol{\theta}_t^* \in \arg \max_{\boldsymbol{\theta}\in\Theta} [\mathbf{M}(f_g(\mathcal{Q},\mathcal{C}_t^*,\boldsymbol{\theta}),\mathcal{G}^*)] \quad (2b)$$

$$\boldsymbol{\phi}_t^* \in \arg \max_{\boldsymbol{\phi}\in\Phi} [\mathbf{P}(f_r(\mathcal{Q},\mathcal{D},\boldsymbol{\phi}),\mathcal{R}^*)] \quad (2c)$$

$$\mathcal{C}_t^* = f_r(\mathcal{Q},\mathcal{D},\boldsymbol{\phi}_t^*) \quad (2d)$$

where $\boldsymbol{\omega} = (\boldsymbol{\phi},\boldsymbol{\theta})$; f_g and f_r are the generator and retriever, respectively; $\boldsymbol{\phi}_t^*$ and $\boldsymbol{\theta}_t^*$ are their hyperparameter configurations optimized at cycle t out of T cycles. $\mathbf{P}$ is the quality metric for the retriever, e.g., the lexical precision; $\mathcal{R}^*$ is the ground truth of the contexts for all queries. Using $\mathcal{Q}$, we seek to tackle Equation 2a by exploring the best hyperparameter configuration $\boldsymbol{\phi}_t^*$ for retriever under $\mathbf{P}$ first, and then use the retrieved contexts $\mathcal{C}_t^*$ under $\boldsymbol{\phi}_t^*$ to jointly optimize and find the optimal generator configuration, leading to the best $\boldsymbol{\theta}_t^*$ on $\mathbf{M}$ thereunder at cycle t. This process is then repeated in a cyclic manner, eventually resulting in the best global $(\boldsymbol{\phi}^*,\boldsymbol{\theta}^*)$ for $\mathcal{Q}$. **Note that the above is not a classic bi-level/block-coordinate optimization, but rather a newly emerged, cyclic dual-sequential optimization at the same level for RAG.** This is because in those classic paradigms, each step often optimizes one block/level of variables with respect to the same global objective [Sinha *et al.*, 2018]. In our case, the cyclic dual-sequential formula uses distinct stage-specific objectives (Equations 2b and 2c), which are implicitly related to each other in the ultimate goal (Equation 2a).

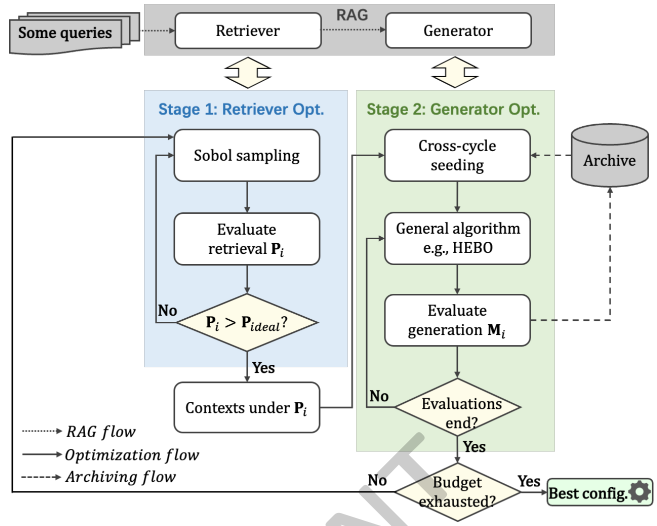


Figure 3: Overview of CDS4RAG to optimize RAG hyperparameters.

4 CDS4RAG Designs

CDS4RAG is precisely designed to address the reformulated problem of cyclic dual-sequential hyperparameter optimization using some queries to RAG, as shown in Figure 3 and Algorithm 1. For that, it has several main mechanisms:

- **Cyclic Dual-Sequential Optimization:** Given a batch of queries, CDS4RAG cyclically optimizes the hyperparameters of the retriever (by Sobol sampling [Sobol', 1967]) and generator (by any general algorithm, e.g., HEBO or TPE) in turn (lines 7-11 and 20-23).
- **Within-Cycle Budget Provisioning:** To determine how the optimization of retriever and generator share the budget, CDS4RAG uses an adaptive mechanism to decide when to stop optimizing the retriever (lines 6 and 13), while for the generator, we set a cap of evaluations (line 20), reducing the costly full retrieval-generation evaluations.
- **Cross-Cycle Generator Seeding** To expedite the expensive optimization for the generator with a warm-start, CDS4RAG selects high-quality "seeds" of the hyperparameter configurations for generators archived across the cycles to be reused under the current cycle (lines 15-19).

The above naturally enable CDS4RAG to be a general framework, such that the underlying algorithm for optimizing generator can be replaced in a plug-and-play manner.

4.1 Dual-Sequential Optimization in Cycles

Optimization in CDS4RAG occurs in two sequential stages with multiple cycles, for which we provide the details below.

Algorithm 1 Pseudo code of CDS4RAG

Input: Budget $\mathcal{B}$; generator cap N; archive of seeds $\mathcal{A}$; retriever quality $\mathbf{P}$; generator quality $\mathbf{M}$; corpus $\mathcal{D}$; given query batch $\mathcal{Q}$; retrieval ground truth $\mathcal{R}^*$; generation ground truth $\mathcal{G}^*$
Output: Best hyperparameter configuration $(\boldsymbol{\phi}^*, \boldsymbol{\theta}^*)$
while $\mathcal{B}$ has not been exhausted **do**
  $t = 1$
  $\boldsymbol{\phi}_t^* = 0; \mathcal{C}_t^* = \emptyset$
  ▷ Optimizing retriever
  while $\mathbf{P}_i$ of $\boldsymbol{\phi}_i < \mathbf{P}_{ideal}$ **do**
    $\boldsymbol{\phi}_i \leftarrow$ find the next retriever configuration using Sobol sampling
    Evaluate the $\mathbf{P}_i$ of $\boldsymbol{\phi}_i$ via $\mathcal{D}$, $\mathcal{Q}$, $\mathcal{R}^*$, Equations 3
    $\mathcal{C}_i \leftarrow$ retrieve contexts with $\boldsymbol{\phi}_i$.
    if $\mathbf{P}_i$ of $\boldsymbol{\phi}_i >$ the $\mathbf{P}$ of $\boldsymbol{\phi}_t^*$ **then**
      $\boldsymbol{\phi}_t^* = \boldsymbol{\phi}_i; \mathcal{C}_t^* = \mathcal{C}_i$
    end if
    ▷ Provisioning budget for retriever
    Update $\mathbf{P}_{ideal}$ using Equations 4 and 5
  end while
  ▷ Seeding the optimization of generator
  $\mathcal{A}' \leftarrow$ De-duplicating $\mathcal{A}$
  $\mathcal{H} = \emptyset$
  $\mathcal{H} \leftarrow \frac{N}{4}$ generator configurations with top 50% $\mathbf{M}$ from $\mathcal{A}'$
  $\mathcal{H} = \mathcal{H} \cup$ randomly generate $\frac{N}{4}$ generator configurations
  Re-evaluate the $\mathbf{M}$ of generator configurations in $\mathcal{H}$ with $\mathcal{C}_i^*$
  ▷ Optimizing generator with provisioned budget
  $\mathcal{H} = \mathcal{H} \cup \leftarrow$ run, e.g., HEBO, with $\mathcal{C}_i^*$, $\mathcal{Q}$, $\mathcal{G}^*$, and $\frac{N}{2}$
  if $\mathbf{M}_i$ of $\forall \boldsymbol{\theta}_i \in \mathcal{H} >$ the $\mathbf{M}$ of $\boldsymbol{\theta}^*$ **then**
    $\boldsymbol{\theta}^* = \boldsymbol{\theta}_i; \boldsymbol{\phi}^* = \boldsymbol{\phi}_t^*$
  end if
  $\mathcal{A} = \mathcal{A} \cup \mathcal{H}; t = t + 1$
end while
Return $(\boldsymbol{\phi}^*, \boldsymbol{\theta}^*)$

Retriever Optimization. Recall from Equation 2c, the goal of optimizing retriever is to find its hyperparameter configuration with the best $\mathbf{P}$ value, for which we define as the average similarity to the ground truth across the top K documents retrieved from the corpus $\mathcal{D}$ for the given query batch $\mathcal{Q}$:

$$\mathbf{P} = \frac{1}{K} \sum_{k=1}^{K} \sum_{q \in \mathcal{Q}}^{|\mathcal{Q}|} \frac{|\mathcal{W}_{d_{q,k}} \cap \mathcal{W}_{r_q^*}|}{|\mathcal{W}_{d_{q,k}}|} \tag{3}$$

Here, we use lexical precision to measure the similarity between a retrieved document $d_{q,k}$ and the ground truth of context r_q^* for a query q, where $\mathcal{W}$ denotes the unique tokens in the corresponding text. The lexical precision is chosen because it is a more efficient signal for finding useful configurations for the generator later on. Importantly, we prefer such over other semantic metrics as it is more strict, biased to prevent false positives, which is more critical for optimizing the generator.

To solve Equation 2c, we use Sobol sampling [Sobol', 1967] in CDS4RAG. This is because, since optimizing retriever only implicitly influences the final generation outcome, we seek to find its diverse yet promising hyperparameter configurations, hence increasing the likelihood to jump out of local optima when optimizing the generator later on. To that end, the Sobol sampling works as a quasi-random low-discrepancy sequence to outperform uniform sampling in high-dimensional cases. Importantly, it provides a distribution-aware exploration and stronger convergence/exploitation than random search. This is crucial for providing the subsequent generation stage with a robust variety of high-quality retrieval foundations. To adopt Sobol sampling to cope with the mixed hyperparameter for retriever, we apply randomized digital scrambling and density-preserving quantization to the Sobol sequences [Cowen-Rivers *et al.*, 2022], ensuring that the low-discrepancy benefits are preserved across the mixed-integer boundary.

The $\boldsymbol{\phi}_t^*$, which is the best hyperparameter configuration of the retriever found at cycle t, evaluated under the retrieval quality $\mathbf{P}$, would be fixed, and its retrieved contexts $\mathcal{C}_t^*$ would be applied in the subsequent generator optimization.

Generator Optimization with Fixed Retrieved Contexts. At cycle t, unlike optimizing the retriever hyperparameters, which focuses on exploration, here we need better exploitation for Equation 2b as the evaluation of the generator is much more expensive. CDS4RAG can be seamlessly paired with any general algorithm, e.g., Bayesian optimization, to optimize the generation quality of RAG using the given queries, e.g., the F1-score. Yet, the retriever hyperparameters are frozen to be $\boldsymbol{\phi}_t^*$ and so do the contexts $\mathcal{C}_t^*$ retrieved thereunder for query batch $\mathcal{Q}$; only the generator's hyperparameters $\boldsymbol{\theta}_i$ are perturbed. The best configuration at cycle t $(\boldsymbol{\phi}_t^*, \boldsymbol{\theta}_t^*)$ is stored.

When all budgets are exhausted, the best generator configuration found and its corresponding retrieval configuration across all cycles, $(\boldsymbol{\phi}^*, \boldsymbol{\theta}^*)$, are returned. Otherwise, we repeat from the retriever optimization in the next cycle $t + 1$. All explored hyperparameter configurations of the generators and their generation quality $(\boldsymbol{\theta}_i, \mathbf{M}_i)$ are archived for the possible cross-cycle seeding next.

4.2 Within-Cycle Budget Provisioning

Since the nature of dual-sequential hyperparameter optimization offers fine-grained control for budget provision, we set different mechanisms for the retriever and generator.

Adaptive Termination for Retriever. For the retriever, we design an adaptive termination mechanism based on the expected progress of retrieval quality, considering all cycles so far. This makes sense since optimizing the retriever is cheap, and it often reaches an ideal retrieval plateau early, after which any more budgets spent on it would not be much beneficial.

At each cycle, we compute an ideal threshold $\mathbf{P}_{ideal}$ of the obtained retrieval quality over query batch $\mathcal{Q}$:

$$\mathbf{P}_{ideal} = \alpha \cdot median(\mathbf{P}_1, \mathbf{P}_2, \cdots, \mathbf{P}_n) + (1 - \alpha) \cdot \Delta \tag{4}$$

where $(\mathbf{P}_1, \mathbf{P}_2, \cdots, \mathbf{P}_n)$ denote the locally achieved values of retrieval quality for all n explored hyperparameter configurations of the retriever at the current cycle. Δ denotes the globally expected quality considering the current and all previous cycles. α controls the relative contribution, and we set equal weights $\alpha = 0.5$ by default. Δ is formulated as:

$$\Delta = \mathbf{P}_{best} \cdot (1 - \frac{t}{t + \beta}) + \mathbf{P}_{95th} \cdot \frac{t}{t + \beta} \tag{5}$$

whereby $\mathbf{P}_{95th}$ and $\mathbf{P}_{best}$ denote the 95th percentile and the best value of all retrieval quality found so far, respectively. t denotes the tth cycle; $\frac{t}{t+\beta}$ is the function that produces a cycle-progress aware weight; we found that $\beta = 2$ is appropriate.

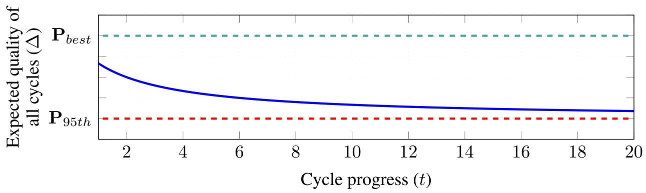


Figure 4: The function of Δ changes with cycle progress t ($\beta = 2$).

As from Figure 4, essentially, $\mathbf{P}_{95th}$ is a more relaxed value compared with $\mathbf{P}_{best}$, serving as the lower bound of Δ. As the cycle increases, $\frac{t}{t+\beta}$ would become closer to 1, hence Δ would be closer to $\mathbf{P}_{95th}$. In this way, we can reduce the contribution of Δ for $\mathbf{P}_{ideal}$ at the latter stage of the optimization, since the results of the current cycle would gradually become more important as the optimization converges.

Finally, if a newly explored configuration of the retriever has a retrieval quality greater than or equal to $\mathbf{P}_{ideal}$, then we terminate the retriever optimization at the current cycle.

Fixed Evaluation Count for Generator. Since optimizing the generator aligned with our main objective, we cap its budget via a fixed number of evaluations N, for which we set $N = 10$ in this work. This is a more intuitive way for one to directly control the trade-off between budget and RAG quality. Clearly, an N that is too large would reduce the contribution from optimizing/exploring the retriever, while a too small N can restrict the exploitation in optimizing the generator. The sensitivity of `CDS4RAG` to N can be found in Appendix C.3.

4.3 Cross-Cycle Generator Seeding

A naive method would be to treat the dual-sequential optimization between cycles as independent runs, but this ignores the optimization knowledge that has been accumulated so far. Further, we observe that the hyperparameter performance of the generator exhibits significant cross-context transferability even when the frozen hyperparameters of the retriever differ.

To capitalize on that, we reuse/seed certain hyperparameter configurations, found in previous cycles, to the generator optimization at the current cycle for a warm start. The seeds can also initialize the surrogate model in, e.g., `BO`. The steps are:

Step 1—Archive de-duplication: In previous cycles, it is possible to have multiple identical hyperparameter configurations of the generator with different generation quality due to their different contexts retrieved, thus we perform a de-duplication that merges those and averages their $\mathbf{M}_i$ values. Thus, all unique hyperparameter configurations in the archive would correspond to a single generation quality $(\boldsymbol{\theta}_i, \mathbf{M}_i)$.

Step 2—Probabilistic seeding: To select which hyperparameter configuration $\boldsymbol{\theta}_i$ of the generator from previous cycles to seed, we rank each configuration, r_i, according to its (average) generation quality. For all J top 50% configurations, we assign a probability to each based on reciprocal of its rank:

$$p(\boldsymbol{\theta}_i) = \frac{r_i^{-1}}{\sum_{j=1}^{J} r_j^{-1}} \tag{6}$$

$p(\boldsymbol{\theta}_i)$ indicates the likelihood of $\boldsymbol{\theta}_i$ to be used for seeding. This ensures that better performing $\boldsymbol{\theta}_i$ would have a higher probability to seed, while there is also a chance of choosing slightly worse ones, hence ensuring diversity.

Stage	Hyperparameter	Type	Range / Values
Retriever (Φ)	Database Choice	Categorical	{DuckDB, Chroma, FAISS}
	Chunk Size	Integer	$[256, 1024]$
	Chunk Overlap	Integer	$[32, 128]$
	Embedding Temperature	Float	$[0.0, 1.0]$
	Embedding Window	Integer	$[512, 2048]$
	Embedding Repeat Penalty	Float	$[0.9, 1.5]$
	Embedding Top-k	Integer	$[10, 100]$
Generator (Θ)	Retrieval Numbers (K)	Integer	$[1, 10]$
	Generation Temperature	Float	$[0.0, 1.0]$
	Generation Window	Integer	$[512, 8192]$
	Generation Repeat Penalty	Float	$[0.9, 1.5]$
	Generation Top-k	Integer	$[10, 100]$

Table 1: RAG system and the hyperparameter space studied.

Recall that N is the cap for the evaluation counts to optimize the generator, we set the number of configurations to be seeded as $\frac{N}{2}$, within which $\frac{N}{4}$ is seeded from the archive of previous cycles, while the remaining $\frac{N}{4}$ are randomly generated configurations. This ensures that the seeds can guide the hyperparameter optimization of the generator without trapping at local optima. Note that if $\frac{N}{4}$ is not an integer, we prefer the random seeds over the archived seeds, e.g., $N = 10$ means that 5 would be the seeds, in which 2 come from the archive.

Step 3—Contextual re-evaluation: Since the generation quality of generator configurations in the archive might be evaluated with a retriever hyperparameter configuration differs from the $\boldsymbol{\phi}_t^*$ at the current cycle t, all seeds for the generator should be re-evaluated with contexts $\mathcal{C}_t^*$ retrieved under $\boldsymbol{\phi}_t^*$.

5 Experiment Setup

5.1 Experiment Subjects

Benchmarks and Knowledge Bases. As the common practices for RAG, we use diverse/widely-used benchmarks [Guo *et al.*, 2025; Stefano *et al.*, 2024; Yang *et al.*, 2018]:

- **Agriculture and Biography [Guo *et al.*, 2025]:** Two domain-specific benchmarks requiring precise entity retrieval. We send all 100 and 180 queries to RAG.
- **BioASQ [Krithara *et al.*, 2023]:** Biomedical questions/answers with complex terminology. We randomly sample 100 queries per run for sending to RAG.
- **HotpotQA [Yang *et al.*, 2018]:** A multi-hop reasoning benchmark for high-fidelity retrieval. Again, we randomly sample 100 queries per run for sending to RAG.

For each benchmark, we consolidate the contextual information of source corpora for the queries into a knowledge base, which is then merged into a unified vector database.

RAG System. Like prior work [Fu *et al.*, 2024; Lewis *et al.*, 2020; Orbach *et al.*, 2025; Guo *et al.*, 2025], we focus on a refined backbone common to most RAG systems to ensure findings are generalizable without the confounding effects of auxiliary modules like re-rankers. The joint space $\Phi \times \Theta$ in Table 1 has up to 12 hyperparameters—a much more complex RAG case compared with the 2-7 hyperparameters used in existing work [Fu *et al.*, 2024; Barker *et al.*, 2025]—including discrete architectural choices and continuous options. This leads to a search space of at least 1.3×10^{18}, if discretized.

Algorithm	Llama-3.1-8B				Qwen-3-8B			
	Agriculture	**Biography**	**HotpotQA**	**BioASQ**	**Agriculture**	**Biography**	**HotpotQA**	**BioASQ**
HEBO	$0.380_{\pm 0.016}$	$0.335_{\pm 0.011}$	$0.571_{\pm 0.065}$	$0.282_{\pm 0.010}$	$0.296_{\pm 0.004}$	$0.299_{\pm 0.003}$	$0.622_{\pm 0.128}$	$0.228_{\pm 0.002}$
CDS4RAG (HEBO)	**$0.383_{\pm 0.013}$**	**$0.352_{\pm 0.007}$**	**$0.626_{\pm 0.017}$**	**$0.293_{\pm 0.011}$**	**$0.305_{\pm 0.009}$**	**$0.317_{\pm 0.007}$**	**$0.759_{\pm 0.036}$**	**$0.235_{\pm 0.004}$**
BO	$0.324_{\pm 0.025}$	$0.322_{\pm 0.019}$	$0.550_{\pm 0.103}$	**$0.266_{\pm 0.016}$**	$0.285_{\pm 0.011}$	$0.303_{\pm 0.009}$	$0.475_{\pm 0.141}$	**$0.236_{\pm 0.003}$**
CDS4RAG (BO)	**$0.366_{\pm 0.019}$**	**$0.343_{\pm 0.014}$**	**$0.612_{\pm 0.043}$**	$0.263_{\pm 0.013}$	**$0.294_{\pm 0.014}$**	**$0.309_{\pm 0.009}$**	**$0.619_{\pm 0.041}$**	$0.231_{\pm 0.006}$
TPE	$0.359_{\pm 0.026}$	$0.335_{\pm 0.011}$	$0.548_{\pm 0.042}$	$0.276_{\pm 0.012}$	$0.286_{\pm 0.008}$	$0.299_{\pm 0.004}$	$0.639_{\pm 0.145}$	**$0.231_{\pm 0.003}$**
CDS4RAG (TPE)	**$0.373_{\pm 0.009}$**	**$0.345_{\pm 0.004}$**	**$0.597_{\pm 0.014}$**	**$0.284_{\pm 0.008}$**	**$0.289_{\pm 0.006}$**	**$0.300_{\pm 0.002}$**	**$0.734_{\pm 0.037}$**	$0.228_{\pm 0.002}$

Table 2: Comparing CDS4RAG paired with HEBO, and BO, and TPE against their vanilla original versions for optimizing RAG over 10 runs. Values show mean$_{\pm \text{Std}}$ of F1-scores across the runs; the best results are highlighted in **bold**. Details of the trajectories can be found in Appendix C.1.

LLMs. We employ Llama-3.1-8B [Meta, 2024] and Qwen3-8B [Alibaba, 2025] as the generator for RAG due to their robust instruction-following capabilities. The prompt for RAG in evaluation can be found at Appendix B.1.

5.2 Compared Approaches and Metric

Compared Approaches. We compare CDS4RAG against several types of hyperparameter optimization algorithms:

- **Baseline:** We use Random Search (Random) and Greedy Search (Greedy) [Orbach *et al.*, 2025] as the baselines.
- **General algorithms:** We also assess the state-of-the-art general algorithm for hyperparameter optimization, including classic BO [Snoek *et al.*, 2012], TPE [Bergstra *et al.*, 2011], the advanced HEBO [Cowen-Rivers *et al.*, 2022]. They jointly optimize all RAG hyperparameters.
- **RAG-specific algorithms:** We compare two state-of-the-art algorithms designed for optimizing RAG hyperparameters: RayTune [Liaw *et al.*, 2018] and AutoRAG-HP [Fu *et al.*, 2024]: the former is a tailored TPE for RAG, while the latter relies on hierarchical multi-armed bandit. Again, they do not distinguish the hyperparameters of the retriever and generator in the optimization.

Despite being relevant, we have had to omit some algorithms due to different reasons: (1) Prompt-only algorithms (e.g., MIPROv2 [Opsahl-Ong *et al.*, 2024]) and single-stage approaches (e.g., AdaptiveRAG [Jeong *et al.*, 2024]) are omitted, as it is unfair to compare with them and they are orthogonal to this work. (2) Multi-fidelity algorithms (e.g., BOHB [Falkner *et al.*, 2018] and DEHB [Awad *et al.*, 2021]) are also ruled out, because multi-fidelity is an additional feature that can easily be paired with CDS4RAG under its cyclic dual-sequential optimization. (3) Since this work focuses on single objective optimization, we do not consider multi-objective RAG optimization algorithms such as METIS [Ray *et al.*, 2025].

Evaluation Metric. To evaluate the generation quality of optimized RAG, we focus on the token-level F1-score (kindly see Appendix B.2) based on the ground truth answers of the given queries of a benchmark as they are sent to RAG, which is also the main optimization objective. F1-score is chosen because it is robust to data imbalance and has been widely used for RAG evaluation [Ray *et al.*, 2025; Rajpurkar *et al.*, 2016; Yu *et al.*, 2024]. Since RAG follows the zero-shot learning paradigm without the need for any (re-)training on the given queries when changing hyperparameters, the above can evaluate/test the generalizability of RAG under optimized hyperparameters, resembling the common way of using a validation loss, which is both an optimization objective and evaluation metric, in classic AutoML [Santu *et al.*, 2022]—the same is also applied in AutoRAG-HP [Fu *et al.*, 2024]. Having said that, we have indeed additionally evaluated the optimized configurations on holdout queries from *BioASQ* and *HotpotQA*, which have sufficient samples (Appendix C.4).

5.3 Implementation and Setting Details

The RAG is implemented via LangChain[3], with OpenBox [Jiang *et al.*, 2024] for CDS4RAG and other hyperparameter optimization algorithms, or otherwise we directly use the code provided by their authors. All experiments run on a cluster of 10 machines, each equipped with an Intel CPU of 24 cores and 32 threads, 128GB RAM, and a NVIDIA RTX 5090 GPU. We set the hyperparameters of CDS4RAG as the aforementioned defaults, and those of the other algorithms as what used by their authors. The details can be found in Appendix B.3. To complete the experiment within a reasonable time while ensuring realism, for each algorithm/run, we set a budget of one hour wall-clock time (where most algorithms have already converged reasonably) and repeat 10 independent runs. In total, the entire experiment took $\approx 1,000$ CPU/GPU hours.

6 Evaluation Results

6.1 Benefits of Cyclic Dual-Sequential Optimization

To assess the benefits of cyclic dual-sequential optimization, we pair CDS4RAG with HEBO, BO, and TPE while comparing them against their respective single-stage optimization (jointly optimizing retriever and generator). As shown in Table 2, CDS4RAG yields considerably better results than its counterparts in 21/24 cases. For both LLMs, the improvement of CDS4RAG appears to be more significant on *HotpotQA* than on others, due to the queries therein being more complex. This makes sense, since if the queries in a benchmark are easy, then the room to be improved is limited. We see that CDS4RAG paired with HEBO has the best result than other variants, since HEBO perform better than BO and TPE in general. On holdout queries, CDS4RAG also lead to significant improvements overall (see Appendix C.4). All those confirm the benefits of distinguishing the hyperparameters of retriever and generator in CDS4RAG and its algorithm-agnostic capability.

[3]https://github.com/langchain-ai/langchain

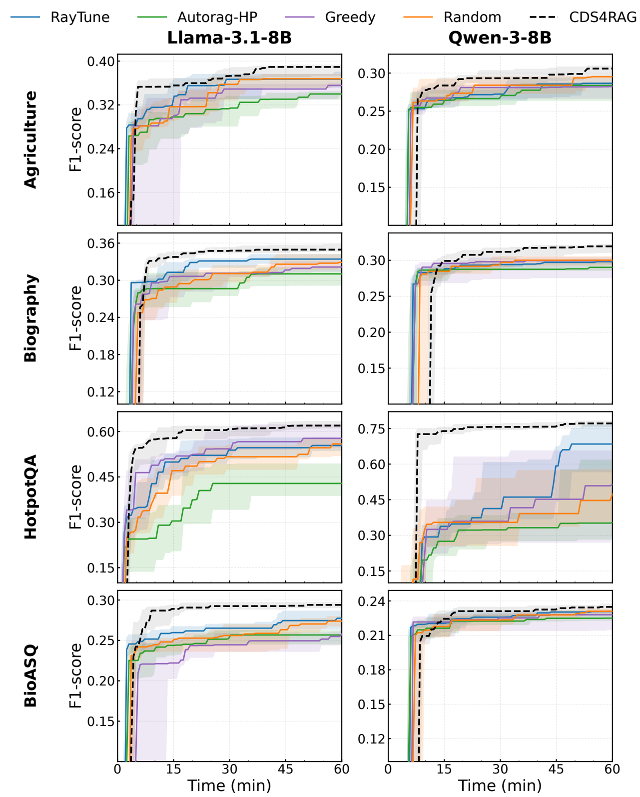


Figure 5: Optimization trajectories of CDS4RAG against the state-of-the-art algorithms for RAG over 10 runs

6.2 Comparison with State-of-the-Art Algorithms

We further evaluate CDS4RAG paired with HEBO against four baselines/state-of-the-art algorithms. As shown in Figure 5, remarkably, CDS4RAG achieves the significantly better results on all cases. Compared with RayTune, CDS4RAG shows a clear advantage, reaching 0.759 F1-score on *HotpotQA* for Qwen-3-8B versus the 0.639 of RayTune. The superiority over RAG-specific algorithms like Autorag-HP is even more pronounced; for example, on *BioASQ* with Llama-3.1-8B, CDS4RAG attains 0.293 compared with 0.259; on *HotpotQA* with Qwen-3-8B, CDS4RAG improves Autorag-HP by $1.54\times$. Most importantly, after the early phase of retriever optimization, the trajectory of CDS4RAG improves much steeper than those of the others with a faster and better convergence/speedup in a matter of a few minutes, showing its superiority even under a limited budget. On holdout queries, the conclusion remains unchanged: over all LLMs/datasets, CDS4RAG (with HEBO) achieves 0.4180 F1-score compared with the 0.3306–0.3853 of the other state-of-the-art algorithms (10%–25% better); see Appendix C.4.

6.3 Ablation Analysis

To confirm the contributions of individual design in CDS4RAG, we replace the within-cycle budget provisioning with fixed, equally dividing budgets to retriever and generator, denoted as `w/o pro`; we also remove the cross-cycle generator seeding, denoted as `w/o seeding`. As shown in Table 3, we see that removing either of the above mechanisms is harmful in general.

Algorithm	Agriculture	Biography	HotpotQA	BioASQ
Llama-3.1-8B				
w/o pro	**0.387**$_{\pm 0.009}$	0.348$_{\pm 0.009}$	0.621$_{\pm 0.009}$	0.289$_{\pm 0.012}$
w/o seeding	0.375$_{\pm 0.009}$	0.347$_{\pm 0.008}$	0.593$_{\pm 0.021}$	0.278$_{\pm 0.007}$
CDS4RAG	0.383$_{\pm 0.013}$	**0.352**$_{\pm 0.007}$	**0.626**$_{\pm 0.017}$	**0.293**$_{\pm 0.011}$
Qwen-3-8B				
w/o pro	0.291$_{\pm 0.009}$	0.299$_{\pm 0.003}$	0.727$_{\pm 0.053}$	0.229$_{\pm 0.003}$
w/o seeding	0.289$_{\pm 0.006}$	0.300$_{\pm 0.004}$	0.752$_{\pm 0.034}$	0.230$_{\pm 0.003}$
CDS4RAG	**0.305**$_{\pm 0.009}$	**0.317**$_{\pm 0.007}$	**0.759**$_{\pm 0.036}$	**0.235**$_{\pm 0.004}$

Table 3: Ablation results on CDS4RAG over 10 runs. The formate is the same as Table 2. Details of the trajectories can be found in Appendix C.2.

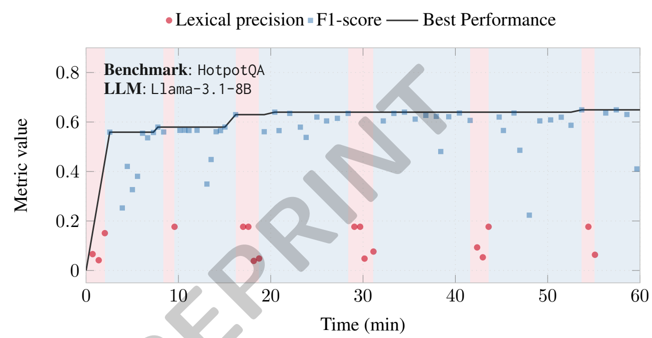


Figure 6: A case of CDS4RAG for RAG that completes $T = 6$ cycles.

Notably, interactions exist between the two mechanisms: the budget provision determines the number of possible cycles, which then influences the cross-cycle seeding that impacts the convergence speed of RAG hyperparameter optimization.

6.4 A Case Study

For the case in Figure 6, CDS4RAG completes 6 cycles with $\approx 17\%$ budgets spent on retrieval evaluations only. In less than 10 minutes, it obtains an excellent generation quality already, followed by F1-score increases at a slow, but steady pace: the marginal improvements of the overall lexical precision for the retriever always help to yield better/more promising overall F1-score for the generator—well-aligned with our hypothesis.

7 Conclusion

This paper presents CDS4RAG, a framework that uniquely distinguishes the hyperparameters of the retriever and generator, optimizing full hyperparameters in a sequential and cyclic manner. Such a paradigm is agnostic to the underlying general algorithm for the generator, enabling finer-grained budget provision within each cycle and cross-cycle generator seeding that can expedite the optimization via reusing promising hyperparameter configurations. Experiments on four benchmarks and two LLMs show that CDS4RAG can boost vanilla algorithms in general while outperforming state-of-the-art algorithms in all cases with up to $1,54\times$ improvement and better speedup. In future work, we seek to extend CDS4RAG with more optional stages of RAG and consider multiple optimization objectives.

Acknowledgments

This work was supported by a NSFC Grant (62372084).

References

[Agrawal *et al.*, 2025] Lakshya A. Agrawal, Shangyin Tan, Dilara Soylu, Noah Ziems, Rishi Khare, Krista Opsahl-Ong, Arnav Singhvi, Herumb Shandilya, Michael J. Ryan, Meng Jiang, Christopher Potts, Koushik Sen, Alexandros G. Dimakis, Ion Stoica, Daniel Klein, Matei Zaharia, and Omar Khattab. GEPA: reflective prompt evolution can outperform reinforcement learning. *CoRR*, abs/2507.19457, 2025.

[Alibaba, 2025] Qwen Team Alibaba. Qwen3 technical report. *CoRR*, abs/2505.09388, 2025.

[Awad *et al.*, 2021] Noor H. Awad, Neeratyoy Mallik, and Frank Hutter. DEHB: evolutionary hyberband for scalable, robust and efficient hyperparameter optimization. In *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI*, pages 2147–2153, 2021.

[Barker *et al.*, 2025] Matthew Barker, Andrew Bell, Evan Thomas, James Carr, Thomas Andrews, and Umang Bhatt. Faster, cheaper, better: Multi-objective hyperparameter optimization for LLM and RAG systems. In *ICLR 2025 Workshop on Foundation Models in the Wild*, 2025.

[Bergstra *et al.*, 2011] James Bergstra, Rémi Bardenet, Yoshua Bengio, and Balázs Kégl. Algorithms for hyper-parameter optimization. In *Annual Conference on Neural Information Processing Systems*, pages 2546–2554, 2011.

[Chen and Chen, 2026] Pengzhou Chen and Tao Chen. Promisetune: Unveiling causally promising and explainable configuration tuning. In *48th IEEE/ACM International Conference on Software Engineering (ICSE)*. ACM, 2026.

[Chen and Li, 2021] Tao Chen and Miqing Li. Multi-objectivizing software configuration tuning. In *ESEC/FSE '21: 29th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, pages 453–465. ACM, 2021.

[Chen and Li, 2024] Tao Chen and Miqing Li. Adapting multi-objectivized software configuration tuning. *Proc. ACM Softw. Eng.*, 1(FSE):539–561, 2024.

[Chen *et al.*, 2018] Tao Chen, Ke Li, Rami Bahsoon, and Xin Yao. FEMOSAA: feature-guided and knee-driven multi-objective optimization for self-adaptive software. *ACM Trans. Softw. Eng. Methodol.*, 27(2):5:1–5:50, 2018.

[Chen *et al.*, 2024] Pengzhou Chen, Tao Chen, and Miqing Li. MMO: meta multi-objectivization for software configuration tuning. *IEEE Trans. Software Eng.*, 50(6):1478–1504, 2024.

[Chen *et al.*, 2025] Pengzhou Chen, Jingzhi Gong, and Tao Chen. Accuracy can lie: On the impact of surrogate model in configuration tuning. *IEEE Trans. Software Eng.*, 51(2):548–580, 2025.

[Chen, 2022] Tao Chen. Lifelong dynamic optimization for self-adaptive systems: Fact or fiction? In *IEEE International Conference on Software Analysis, Evolution and Reengineering, SANER*, pages 78–89. IEEE, 2022.

[Cowen-Rivers *et al.*, 2022] Alexander Imani Cowen-Rivers, Wenlong Lyu, Rasul Tutunov, Zhi Wang, Antoine Grosnit, Ryan-Rhys Griffiths, Alexandre Max Maraval, Jianye HAO, Jun Wang, Jan Peters, and Haitham Bou Ammar. HEBO: Pushing the limits of sample-efficient hyperparameter optimisation. In *First Conference on Automated Machine Learning (Journal Track)*, 2022.

[Falkner *et al.*, 2018] Stefan Falkner, Aaron Klein, and Frank Hutter. BOHB: robust and efficient hyperparameter optimization at scale. In *Proceedings of the 35th International Conference on Machine Learning, ICML*, volume 80 of *Proceedings of Machine Learning Research*, pages 1436–1445. PMLR, 2018.

[Fu *et al.*, 2024] Jia Fu, Xiaoting Qin, Fangkai Yang, Lu Wang, Jue Zhang, Qingwei Lin, Yubo Chen, Dongmei Zhang, Saravan Rajmohan, and Qi Zhang. Autorag-hp: Automatic online hyper-parameter tuning for retrieval-augmented generation. In *Findings of the Association for Computational Linguistics: EMNLP*, pages 3875–3891, 2024.

[Gong and Chen, 2023] Jingzhi Gong and Tao Chen. Predicting software performance with divide-and-learn. In *Proceedings of the 31st ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering, ESEC/FSE*, pages 858–870, 2023.

[Gong and Chen, 2024] Jingzhi Gong and Tao Chen. Predicting configuration performance in multiple environments with sequential meta-learning. *Proc. ACM Softw. Eng.*, 1(FSE):359–382, 2024.

[Gong *et al.*, 2025] Jingzhi Gong, Tao Chen, and Rami Bahsoon. Dividable configuration performance learning. *IEEE Trans. Software Eng.*, 51(1):106–134, 2025.

[Guo *et al.*, 2025] Zirui Guo, Lianghao Xia, Yanhua Yu, Tu Ao, and Chao Huang. LightRAG: Simple and fast retrieval-augmented generation. In *Findings of the Association for Computational Linguistics: EMNLP*, 2025.

[Jeong *et al.*, 2024] Soyeong Jeong, Jinheon Baek, Sukmin Cho, Sung Ju Hwang, and Jong Park. Adaptive-rag: Learning to adapt retrieval-augmented large language models through question complexity. In *the Conference of the North American Chapter of the Association for Computational Linguistics: NAACL*, pages 7036–7050, 2024.

[Jiang *et al.*, 2024] Huaijun Jiang, Yu Shen, Yang Li, Beicheng Xu, Sixian Du, Wentao Zhang, Ce Zhang, and Bin Cui. Openbox: A python toolkit for generalized black-box optimization. *J. Mach. Learn. Res.*, 25:120:1–120:11, 2024.

[Krithara *et al.*, 2023] Anastasia Krithara, Anastasios Nentidis, Konstantinos Bougiatiotis, and Georgios Paliouras. Bioasq-qa: A manually curated corpus for biomedical question answering. *Scientific Data*, 10:170, 2023.

[Lewis *et al.*, 2020] Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. Retrieval-augmented generation for knowledge-intensive NLP tasks.

In *Annual Conference on Neural Information Processing Systems*, 2020.

[Li *et al.*, 2020] Ke Li, Zilin Xiang, Tao Chen, Shuo Wang, and Kay Chen Tan. Understanding the automated parameter optimization on transfer learning for cross-project defect prediction: an empirical study. In *International Conference on Software Engineering (ICSE)*, pages 566–577, 2020.

[Liaw *et al.*, 2018] Richard Liaw, Eric Liang, Robert Nishihara, Philipp Moritz, Joseph E. Gonzalez, and Ion Stoica. Tune: A research platform for distributed model selection and training. *CoRR*, abs/1807.05118, 2018.

[Meta, 2024] Llama Team Meta. The llama 3 herd of models. *CoRR*, abs/2407.21783, 2024.

[Mridul *et al.*, 2025] Maruf Ahmed Mridul, Ian Sloyan, Aparna Gupta, and Oshani Seneviratne. Ai4contracts: LLM & rag-powered encoding of financial derivative contracts. In *the Thirty-Fourth International Joint Conference on Artificial Intelligence, IJCAI*, pages 9305–9312, 2025.

[Niu *et al.*, 2024] Cheng Niu, Yuanhao Wu, Juno Zhu, Siliang Xu, Kashun Shum, Randy Zhong, Juntong Song, and Tong Zhang. Ragtruth: A hallucination corpus for developing trustworthy retrieval-augmented language models. In *the 62nd Annual Meeting of the Association for Computational Linguistics: ACL*, pages 10862–10878, 2024.

[Opsahl-Ong *et al.*, 2024] Krista Opsahl-Ong, Michael J. Ryan, Josh Purtell, David Broman, Christopher Potts, Matei Zaharia, and Omar Khattab. Optimizing instructions and demonstrations for multi-stage language model programs. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 9340–9366, 2024.

[Orbach *et al.*, 2025] Matan Orbach, Ohad Eytan, Benjamin Sznajder, Ariel Gera, Odellia Boni, Yoav Kantor, Gal Bloch, Omri Levy, Hadas Abraham, Nitzan Barzilay, Eyal Shnarch, Michael Factor, Shila Ofek-Koifman, Paula Ta-Shma, and Assaf Toledo. An analysis of hyper-parameter optimization methods for retrieval augmented generation. *CoRR*, abs/2505.03452, 2025.

[Qiao *et al.*, 2025] Yuming Qiao, Yuechen Wang, Dan Meng, Haonan Lu, Zhenyu Yang, and Xudong Zhang. Msrag: Knowledge augumented image captioning with object-level multi-source RAG. In *Proceedings of the Thirty-Fourth International Joint Conference on Artificial Intelligence, IJCAI*, pages 6093–6101, 2025.

[Rajpurkar *et al.*, 2016] Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. Squad: 100, 000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, EMNLP*, pages 2383–2392, 2016.

[Ray *et al.*, 2025] Siddhant Ray, Rui Pan, Zhuohan Gu, Kuntai Du, Shaoting Feng, Ganesh Ananthanarayanan, Ravi Netravali, and Junchen Jiang. METIS: fast quality-aware RAG systems with configuration adaptation. In *Proceedings of the ACM SIGOPS 31st Symposium on Operating Systems Principles, SOSP*, pages 606–622. ACM, 2025.

[Salemi and Zamani, 2024] Alireza Salemi and Hamed Zamani. Evaluating retrieval quality in retrieval-augmented generation. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR*, pages 2395–2400, 2024.

[Santu *et al.*, 2022] Shubhra Kanti Karmaker Santu, Md. Mahadi Hassan, Micah J. Smith, Lei Xu, Chengxiang Zhai, and Kalyan Veeramachaneni. Automl to date and beyond: Challenges and opportunities. *ACM Comput. Surv.*, 54(8):175:1–175:36, 2022.

[Sinha *et al.*, 2018] Ankur Sinha, Pekka Malo, and Kalyanmoy Deb. A review on bilevel optimization: From classical to evolutionary approaches and applications. *IEEE Trans. Evol. Comput.*, 22(2):276–295, 2018.

[Snoek *et al.*, 2012] Jasper Snoek, Hugo Larochelle, and Ryan P. Adams. Practical bayesian optimization of machine learning algorithms. In *Annual Conference on Neural Information Processing Systems*, pages 2960–2968, 2012.

[Sobol', 1967] I.M Sobol'. On the distribution of points in a cube and the approximate evaluation of integrals. *USSR Computational Mathematics and Mathematical Physics*, 7(4):86–112, 1967.

[Stefano *et al.*, 2024] Gianluca De Stefano, Lea Schönherr, and Giancarlo Pellegrino. Rag and roll: An end-to-end evaluation of indirect prompt manipulations in llm-based application frameworks. *CoRR*, abs/2408.05025, 2024.

[Xiang *et al.*, 2026] Zezhen Xiang, Jingzhi Gong, and Tao Chen. Dually hierarchical drift adaptation for online configuration performance learning. In *48th International Conference on Software Engineering (ICSE)*, 2026.

[Xiong and Chen, 2025] Gangda Xiong and Tao Chen. Cotune: Co-evolutionary configuration tuning. In *40th IEEE/ACM International Conference on Automated Software Engineering, ASE*, pages 1490–1502. IEEE, 2025.

[Yang *et al.*, 2018] Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William W. Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. Hotpotqa: A dataset for diverse, explainable multi-hop question answering. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 2369–2380, 2018.

[Ye *et al.*, 2025] Yulong Ye, Tao Chen, and Miqing Li. Distilled lifelong self-adaptation for configurable systems. In *47th IEEE/ACM International Conference on Software Engineering, ICSE*, pages 1333–1345. IEEE, 2025.

[Ye *et al.*, 2026] Yulong Ye, Hongyuan Liang, Chao Jiang, Miqing Li, and Tao Chen. Revealing domain-spatiality patterns for configuration tuning: Domain knowledge meets fitness landscapes. *ACM Trans. Softw. Eng. Methodol.*, 2026.

[Yu *et al.*, 2024] Hao Yu, Aoran Gan, Kai Zhang, Shiwei Tong, Qi Liu, and Zhaofeng Liu. Evaluation of retrieval-augmented generation: A survey. *CoRR*, abs/2405.07437, 2024.