CoEvo: Continual Evolution of Symbolic Solutions Using Large Language Models

Ping Guo[1], **Qingfu Zhang**[1*], **Xi Lin**[1]

[1]City University of Hong Kong, No. 83 Tat Chee Road, Hong Kong SAR, China

Abstract

The discovery of symbolic solutions—mathematical expressions, logical rules, and algorithmic structures—is fundamental to advancing scientific and engineering progress. However, traditional methods often struggle with search efficiency and fail to integrate knowledge effectively. While recent large language model-based (LLM-based) approaches have demonstrated improvements in search efficiency, they lack the ability to continually refine and expand upon discovered solutions and their underlying knowledge, limiting their potential for *open-ended innovation*. To address these limitations, we introduce CoEvo, a novel framework that leverages large language models within an evolutionary search methodology to continually generate and refine symbolic solutions. CoEvo integrates a dynamic knowledge library, enabling open-ended innovation of solutions through effective knowledge management. Additionally, CoEvo leverages multiple representations of solutions—including natural language, mathematical expressions, and code—to further enhance search efficiency. By combining the reasoning capabilities of LLMs with the exploratory power of evolutionary algorithms, CoEvo significantly improves the efficiency and scope of symbolic discovery. Our experimental results demonstrate that this method not only enhances the efficiency of searching for symbolic solutions but also supports the ongoing discovery process, akin to human scientific endeavors. This study represents a first effort in conceptualizing the search for symbolic solutions as a lifelong, iterative process, marking a significant step towards harnessing LLMs in the perpetual pursuit of scientific and engineering breakthroughs.

Code — https://github.com/pgg3/CoEvo

Introduction

The pursuit of symbolic solutions—mathematical models, logical rules, and algorithmic structures—lies at the heart of scientific and engineering progress, underpinning both theoretical frameworks and practical applications (Wigner 1990; Newell 1980). From designing complex engineering systems to formulating new scientific theories, the discovery of such solutions drives innovation and technological advancement (Gielen and Sansen 2012; Schmidt and Lipson 2009). For instance, symbolic solutions like Intellectual Property (IP) blocks are essential for optimizing system performance and accelerating development cycles (Brown and Vranesic 2000; Wolf 2002). However, the process of discovering these solutions is often hindered by the difficulties of navigating vast representation spaces and integrating new knowledge into the search process.

A promising approach to advancing symbolic discovery involves emulating the open-ended, iterative nature of human scientific and engineering endeavors, where solutions and foundational knowledge co-evolve over time. This approach has the potential to unlock innovative ideas that are inaccessible through conventional methods (Stanley, Lehman, and Soros 2017; Lenat 1983). In artificial intelligence, open-ended exploration has led to significant breakthroughs by establishing environments where algorithms endlessly generate and refine solutions without the constraints of predefined goals (Lehman and Stanley 2011; Faldor et al. 2024). This paradigm mirrors the iterative essence of human creativity and scientific discovery, where each insight sparks further questions and exploration (Boden 2004).

Recent advances in large language models (LLMs) have demonstrated their ability to automate problem-solving tasks across diverse domains, including code generation and scientific reasoning (OpenAI 2023a; Liu et al. 2024c; Ifargan et al. 2024). However, their application to open-ended symbolic discovery—where solutions and knowledge continually evolve—remains underexplored. While LLMs incorporate baseline human knowledge, they struggle to integrate newly discovered insights and generate novel solutions in an open-ended manner, not to mention traditional methods such as evolutionary algorithms (Cranmer 2023a) and deep learning approaches (Biggio et al. 2021a; Kamienny et al. 2022). Although effective in specific contexts, these methods often fall short for achieving the open-ended innovation characteristics of human scientific endeavors.

Efforts to enhance LLMs for symbolic discovery have focused on improving their problem-solving capabilities through techniques such as retrieval-augmented generation (RAG) (Huang and Huang 2024) and domain-specific fine-tuning (Roziere et al. 2023). While these approaches enhance search efficiency, they primarily enable LLMs to reuse prior knowledge rather than create or refine new knowledge. This raises a critical question:

Can LLMs uncover new knowledge rather than merely

*Corresponding author: qingfu.zhang@cityu.edu.hk

replicate existing information? In addition, can LLMs summarize and evolve knowledge to support an open-ended search for symbolic solutions akin to human endeavors?

To address these challenges, we introduce CoEvo, a novel framework that combines LLMs with an evolutionary search methodology to continually generate and refine symbolic solutions alongside underlying knowledge. CoEvo incorporates a dynamic knowledge library and multiple representation spaces, enabling the open-ended evolution of solutions across diverse formats, including natural language, mathematical expressions, and code. This approach tackles two key challenges: (1) the management of evolving knowledge and (2) the exploration of diverse representation spaces in an open-ended manner. To our knowledge, this represents the first effort to frame symbolic discovery as a continual, open-ended process powered by LLMs.

Our contributions are as follows:

- We make a first attempt to extract knowledge and apply it in the endless search for symbolic solutions for scientific and engineering challenges. We believe that finding symbolic solutions in domains such as scientific discovery should be perceived as a continual open-ended process.
- We propose a framework for harvesting and applying knowledge to search for symbolic solutions in multiple search spaces, with the understanding that the knowledge is continually evolving.
- Our extensive experimental outcomes demonstrate that incorporating the newly generated knowledge enables a lifetime of continual searching for symbolic solutions in scientific and engineering domains.

Background

Symbolic Regression

Symbolic regression has been widely employed to uncover mathematical equations that capture underlying relationships in datasets (Udrescu and Tegmark 2019). Before the rise of LLMs, techniques in this field were broadly categorized into three main approaches: search-based, learning-based, and hybrid methods (Cranmer 2023a).

Conventional search-based approaches primarily employ evolutionary algorithms to explore the space of equation structures and parameters. These methods rely on carefully designed solution representations, such as expression trees, to iteratively evolve candidate solutions (Schmidt and Lipson 2009; Cranmer 2023a). However, designing such representations is challenging, and their efficiency in traversing the search space remains uncertain (Kronberger et al. 2024).

With the advances in transformer-based architectures, learning-based models have become increasingly prominent in symbolic regression. This shift has also led to hybrid methods that combine the strengths of both search-based and learning-based approaches (Biggio et al. 2021a; Kamienny et al. 2022). While these methods have addressed some efficiency concerns, they still struggle to effectively incorporate prior knowledge, which could further enhance both the efficiency and interpretability of the discovered solutions.

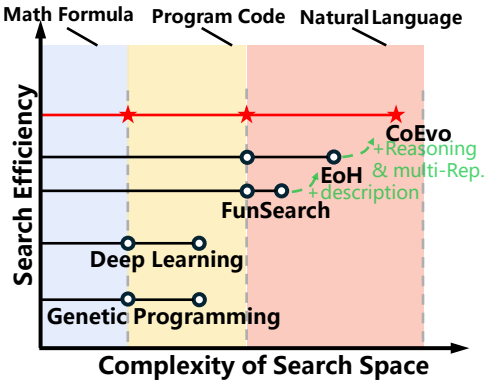


Figure 1: Conceptual comparison of symbolic discovery methods across search spaces of increasing complexity and knowledge richness. Traditional approaches (pre-LLM) operate in constrained mathematical/code spaces. LLM-based methods (FunSearch, LLM-SR) leverage iterative evolution in code space, while EoH incorporates natural language heuristics. CoEvo (proposed) fully exploits LLMs' reasoning in natural language space for open-ended evolution.

LLMs and Symbolic Discovery

LLMs have been increasingly employed in scientific discovery tasks due to their ability to process and generate natural language, code, and mathematical expressions (OpenAI 2023a; Abramson et al. 2024). Initially, LLMs were applied to discover mathematical equations and algorithms through iterative evolution of solutions implemented in Python code (Liu et al. 2024c; Romera-Paredes et al. 2024; Liu et al. 2024a). This approach, which equips LLMs with a systematic evaluator and enables interaction via program code, has demonstrated promising results in deriving symbolic solutions. For instance, LLM-SR (Shojaee et al. 2024) adopts the methodology of FunSearch (Romera-Paredes et al. 2024) to iteratively evolve solutions, which are then forwarded to specialized evaluators.

The state-of-the-art method, LLM-SR, has achieved notable success in symbolic regression tasks, validating the effectiveness of iterative evolution and establishing an evaluation framework for such tasks. However, it assumes static knowledge and generates solutions in a single format, failing to fully exploit the reasoning capabilities of LLMs to derive new knowledge.

Figure 1 presents a conceptual comparison of existing methods. We categorize the complexity of the search space into three levels: *1)* mathematical formula space, *2)* code space, and *3)* natural language space. As the complexity of the search space increases, so does the richness and complexity of the knowledge it encompasses.

The figure contrasts the search efficiency of conventional symbolic regression methods with LLM-based approaches. Prior to the advent of LLMs, both search-based and learning-based methods were confined to the mathematical formula and code spaces, with limited capacity to leverage broader knowledge. With the emergence of LLMs, the search space has expanded to include natural language. Ap-

plications such as FunSearch (Romera-Paredes et al. 2024) and LLM-SR (Shojaee et al. 2024) have demonstrated the efficacy of iterative evolution in symbolic regression tasks. Another approach, EoH (Liu et al. 2024a), also employs iterative evolution but further capitalizes on the linguistic capabilities of LLMs by incorporating natural language heuristics. Our proposed method, CoEvo, advances this further by harnessing the reasoning abilities of LLMs to derive new knowledge and evolve solutions in an open-ended manner.

CoEvo: Continual Evolution of Symbolic Solutions using LLMs

Overview

CoEvo is a framework designed to facilitate a continuous and open-ended search process for symbolic solutions while effectively managing underlying knowledge, as illustrated in Figure 2. The system incorporates an evolutionary search loop that employs idea tree-based solution generation to produce solutions in diverse formats. Additionally, it features a knowledge library that systematically stores and retrieves knowledge fragments throughout the search process.

Idea Tree-based Solution Generation

Solution generation is a crucial component of CoEvo, which is essential during both initialization and offspring production. The framework employs an idea tree-based approach to generate solutions in multiple formats, following a three-step process: *1)* inspiring, *2)* thinking, and *3)* solving. Additionally, the solutions are generated in diverse formats to facilitate exploration across different search spaces.

Our three-step solution generation process closely mirrors human problem-solving, as illustrated in Figure 3. Typically, humans first generate preliminary ideas when presented with a task and subsequently refine them. This methodology aligns with two key insights from LLM research: *1)* the Reason-and-Act framework (Yao et al. 2023), which reflects the iterative cycle of reasoning and action, and *2)* the tree-based multi-phase search process (Wei et al. 2022; Yao et al. 2024), which promotes the exploration of diverse ideas.

Expanding on these insights, we propose an idea tree-based solution generation for task solution generation, as illustrated in Figure 2, part (b).

Idea Tree. The process begins by generating a diverse set of N_0 initial ideas, serving as the root nodes of the tree structure. At each subsequent level k, N_k ideas are developed through direct inference, guided by the evaluator's feedback and existing ideas from the previous level. This iterative refinement enhances the initial concepts and forms a network-like structure for deeper exploration.

Unlike the exhaustive sampling-and-branching process in Tree-of-Thought (Yao et al. 2024), our approach avoids exponential growth in computational resources by adopting a more constrained, network-like structure. Nevertheless, the framework remains fully compatible with such extensions, ensuring scalability for future applications.

Representations of Solutions. To facilitate the generation of solutions in diverse formats, we introduce multiple representations. Prior research has demonstrated the effectiveness of parallel searches across natural language and Python code spaces, particularly in code generation tasks (Wang et al. 2024; Liu et al. 2024a,b). Below, we present examples of several representative formats:

- **Natural Language:** The foundational concept of LLMs is rooted in linguistic principles (Zhao et al. 2024). This domain aligns well with LLMs' capabilities and is central to related research.
- **Mathematical Formulas:** A fundamental representation for expressing mathematical functions and equations. In implementation, we use LaTeX code to represent these formulas.
- **Python Code:** This format is chosen because current LLMs are primarily trained on Python code, enhancing their code-generation capabilities (Roziere et al. 2023; Dubey et al. 2024). Additionally, code representations enable automated task evaluation (Liu et al. 2024a; Romera-Paredes et al. 2024).

Knowledge Library

Figure 4 illustrates the interaction between the knowledge library and the population in generating ideas at different levels during the evolutionary search process. Throughout this process, the library collects, stores, and reuses knowledge to enhance search efficiency. To support these functions, the knowledge library incorporates three key mechanisms: *1)* summarization, *2)* management, and *3)* reuse.

Idea Summarization. Idea summarization occurs when solutions yield improved scores during tree-based search and offspring generation. An elevated score, as determined by the evaluator, signifies that the solution is effective and that the modifications introduced meaningful, knowledge-rich improvements. Consequently, the LLM is prompted to extract and summarize the key idea underlying these changes. The summarized idea is then stored in the knowledge library in a structured definition-description format.

Idea Management. Idea management is a critical component of the knowledge library, ensuring that stored ideas remain organized and retrievable. We configure the knowledge library to maintain a finite number of ideas, as we want to avoid overwhelming the system with excessive knowledge and unleash the power of continual learning. To prevent system overload and promote continual learning, the knowledge library maintains a finite number of ideas. An excessive volume of ideas can hinder search and retrieval efficiency, and redundancy in learned knowledge cannot be guaranteed. To address this, we cluster ideas based on their semantic similarity, computed by cosine similarity between their sentence embeddings, thereby reducing redundancy while preserving key insights.

Idea Reuse. The knowledge library is used to retrieve the knowledge when needed in two modes *Random Reuse* and *Similarity-based Reuse*. When the LLM is generating new solutions, it needs to explore the search space to the largest extent, thus it is provided with random ideas from the knowledge library from each cluster for inspiration. However, when it is conducting tree-based idea search, it needs to explore the ideas that are related to the ideas in the previous

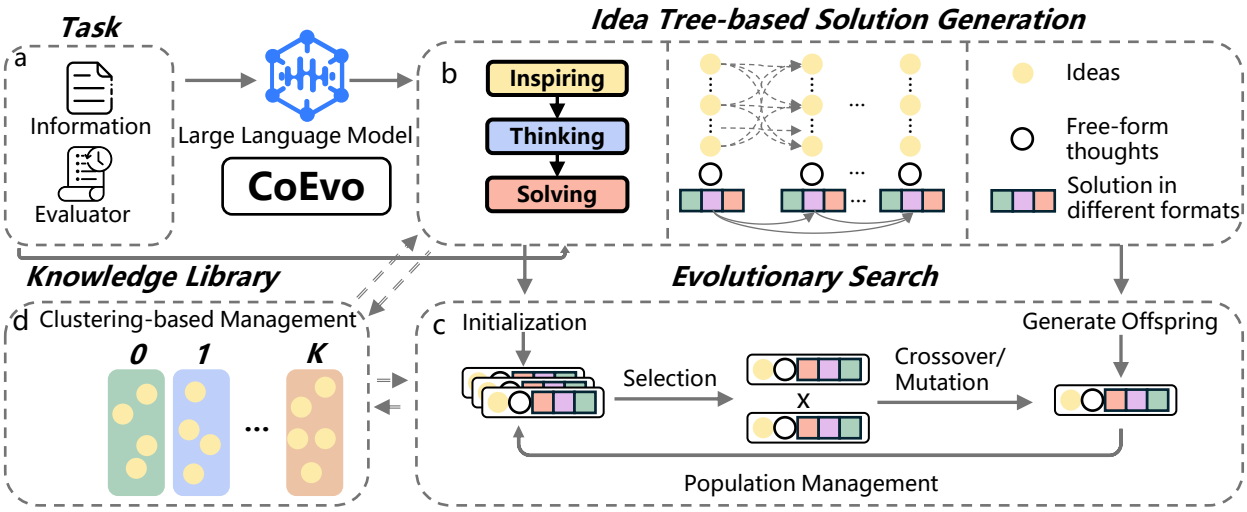


Figure 2: **An overview of CoEvo. (a)** Task of interest. **(b)** Tree-based solution generation for generating of a single solution in different formats. **(c)** Evolutionary search of solutions. **(d)** Knowledge library for storing and retrieving knowledge pieces.

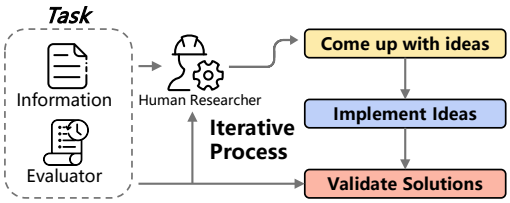


Figure 3: Human thinking process. It is usually an iterative process of idea generation, evaluation, and refinement.

level, thus similarity-based reuse is used. Similarity reuse involves calculate the distances from the current ideas to the ideas in the knowledge library and retrieve the ideas that are most similar to the current ideas.

Evolutionary Search

Our implementation of the evolution process adheres to the standard steps of evolutionary algorithms: initialization, crossover, mutation, and population management.

Initialization. We initiate by randomly generating a set of N solutions as the initial population using the tree-based idea search process in the previous section. Notably, when the initialization starts with an empty knowledge library, the solutions are generated without any prior knowledge. Otherwise the knowledge library is used to inspire the generation of initial solutions as described in Figure 4.

Crossover. We employ two crossover operators—*positive-crossover* and *negative-crossover*. Positive-crossover promotes the generation of solutions similar to parent ideas, whereas negative-crossover fosters the creation of distinctly different solutions, enhancing solution diversity as validated in related research (Wang et al. 2024; Liu et al. 2024d,a).

Mutation. We implement two mutation operators: *positive-mutation* and *negative-mutation*. Positive-mutation introduces small, incremental changes to existing solutions,

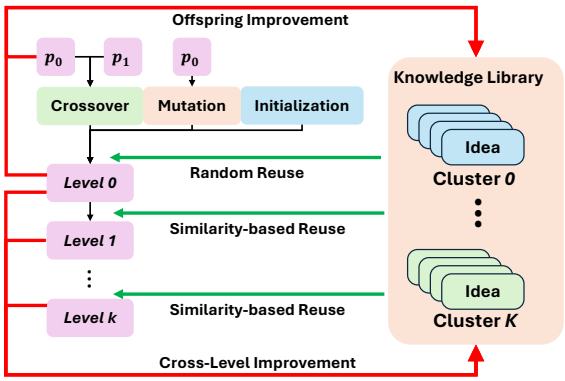


Figure 4: An illustration of the interaction between the knowledge library and the population. The red arrow represents the addition of knowledge to the library, while the green arrow denotes the reuse of knowledge.

while negative-mutation applies more significant alterations.

Population Update. We maintain the top N solutions with the highest scores as the population for the next generation, ensuring a quality-driven evolution.

Experiments

Experimental Setup

To evaluate the effectiveness of our proposed method, we conduct experiments on a subset of scientific problems from the AI Feynman benchmark (Udrescu and Tegmark 2019) as described in LLM-SR(Shojaee et al. 2024). Specifically, we compare our method against leading symbolic regression techniques, including both evolutionary search and deep learning-based methods. We also include LLM-SR, the most recent LLM-based symbolic regression method, as a base-

Category	Method	Oscillation 1		Oscillation 2		E. coli growth		Stress-Strain	
		ID ↓	OOD ↓	ID ↓	OOD ↓	ID ↓	OOD ↓	ID ↓	OOD ↓
Evolutionary Search*	GPlearn[1]	0.0155	0.5567	0.7551	3.188	1.081	1.039	0.1063	0.4091
	PySR (Cranmer 2023b)	0.0009	0.3106	0.0002	0.0098	0.0376	1.0141	0.0331	0.1304
Deep Learning*	NeSymReS (Biggio et al. 2021b)	0.0047	0.5377	0.2488	0.6472	N/A ($d > 3$)		0.7928	0.6377
	E2E (Kamienny et al. 2022)	0.0082	0.3722	0.1401	0.1911	0.6321	1.4467	0.2262	0.5867
	DSR (Petersen et al. 2021)	0.0087	0.2454	0.0580	0.1945	0.9451	2.4291	0.3326	1.108
	uDSR (Landajuela et al. 2022)	0.0003	0.0007	0.0032	0.0015	0.3322	5.4584	0.0502	0.1761
LLM-based	LLM-SR (Mixtral)* (Shojaee et al. 2024)	7.89e-8	0.0002	0.0030	0.0291	0.0026	0.0037	0.0162	0.0946
	LLM-SR (`gpt-3.5-turbo`) (Shojaee et al. 2024)	6.02e-9	0.0004	2.55e-7	3.03e-3	0.0207	0.0547	0.0017	0.0025
	LLM-SR (`gpt-4o-mini`) (Shojaee et al. 2024)	5.14e-9	0.0003	1.79e-7	3.11e-5	0.0214	0.0264	0.0020	0.0020
	CoEvo (Ours, `gpt-3.5-turbo`)	**4.32e-9**	8.71e-5	**1.58e-10**	**1.32e-10**	**1.58e-9**	**1.21e-8**	0.0020	0.0015
	CoEvo (Ours, `gpt-4o-mini`)	1.28e-8	**7.51e-5**	2.98e-7	2.21e-4	0.0019	0.0107	**0.0018**	**9.90e-4**

*: Reported results from LLM-SR (Shojaee et al. 2024).
[1]: https://gplearn.readthedocs.io/en/stable/

Table 1: Comparison of the overall performance of our method with three categories of approaches: evolutionary search, deep learning, and LLM-SR. The normalized mean squared error (NMSE) is reported for both training (in-distribution, ID) and test (out-of-distribution, OOD) data. The best results are highlighted in bold with a gray background, while the second-best results are underlined.

line for LLM-driven approaches. We present the experimental setup in the following subsections.

Benchmark. We adopted the four problems introduced in LLM-SR (Shojaee et al. 2024) for performance evaluation. These problems are Oscillation 1, Oscillation 2, E. coli growth, and Stress-Strain. They were specifically adapted from the original AI Feynman benchmark to prevent rote memorization of solutions. The AI Feynman benchmark (Udrescu and Tegmark 2019), which comprises 120 physics problems, is the current standard for evaluating symbolic regression methods in scientific equation discovery.

Algorithms. We compare our method against a range of symbolic regression techniques, including both evolutionary search, deep learning-based methods, and LLM-based approaches. For evolutionary search, we include `GPlearn`[1] and `PySR` (Cranmer 2023b), which are two widely used symbolic regression libraries. For deep learning-based methods, we consider NeSymReS (Biggio et al. 2021b), E2E (Kamienny et al. 2022), DSR (Petersen et al. 2021), and uDSR (Landajuela et al. 2022). For LLM-based approaches, we include LLM-SR (Shojaee et al. 2024), which is the most recent and advanced method in this category. The evaluation results of non LLM-based methods are reported directly from LLM-SR (Shojaee et al. 2024).

For LLM-SR and our method, we limit the number of iterations to 2, 000 for fair comparison. Moreover, we set the number of generations to 100 and the number of samples to 20 for each generation. The size of the knowledge library is set to 30.

Backbone LLMs. To assess the efficacy of LLM-based methods, we employ two backbone LLMs: `gpt-3.5-turbo` and `gpt-4o-mini`. The former, `gpt-3.5-turbo`, is a widely adopted model known for its efficiency and effectiveness, serving as the default backbone in LLM-SR (Shojaee et al. 2024). It offers a balance of cost-effectiveness and robust performance on symbolic regression tasks. To explore the potential of more advanced models, we also include `gpt-4o-mini`, a newer iteration with enhanced capabilities. This model is anticipated to excel in handling complex tasks. We exclude other models, such as `Claude-3.5` (due to comparable performance to `gpt-3.5-turbo`) and `gpt-o1` (owing to prohibitive costs).

Specifically, `gpt-3.5-turbo` has a knowledge cutoff of September 2021 (OpenAI 2023b), while `gpt-4o-mini` extends to October 2023 (OpenAI 2024).

[1]https://gplearn.readthedocs.io/en/stable/

Overall Performance

The overall performance of all methods on the selected benchmark problems is summarized in Table 1. The normalized mean squared error (NMSE) serves as the performance metric, where lower values indicate better performance.

Superior Performance. CoEvo consistently outperforms all other methods across all tested problems, achieving the lowest NMSE values. This demonstrates its effectiveness in symbolic regression tasks. Notably, CoEvo delivers results several orders of magnitude better on the Oscillation 2 and E. coli growth problems, highlighting its robustness in addressing complex symbolic regression challenges.

Our method exhibits minimal dependence on the choice of LLM, as it achieves comparable performance with both `gpt-3.5-turbo` and `gpt-4o-mini`. Impressively, CoEvo successfully identifies the implicit equation for Oscillation 2, which is not discovered by LLM-SR using `gpt-4o-mini`.

Characteristic of Oscillation 2. Oscillation 2 models the motion of an object governed by a precise physical equation defining its acceleration. Consequently, symbolic regression methods are expected to achieve near-zero NMSE by recovering the underlying analytical expression for acceleration.

However, the design of the evaluator presents a unique characteristic. The evaluator processes data strictly in time order. This sequential, time-series nature of the evaluation data introduces an alternative, highly effective pathway to compute acceleration: applying the `numpy.gradient` function directly to the velocity data yields an accurate numerical differentiation representing acceleration.

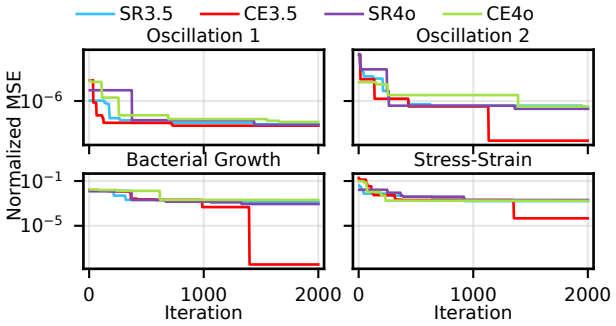


Figure 5: The NMSE values during the search process of our method and LLM-SR. SR: LLM-SR; CE: CoEvo; 3.5: GPT-3.5; 4o: GPT-4o-mini.

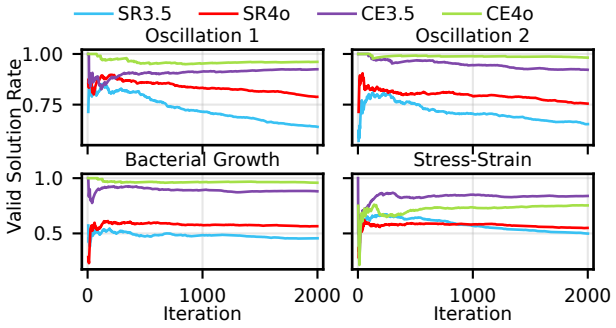


Figure 6: The ratio of the generated solutions that are valid during the search process. SR: LLM-SR; CE: CoEvo; 3.5: GPT-3.5; 4o: GPT-4o-mini.

While other symbolic regression methods focused on recovering the explicit physical equation, only CoEvo, leveraging the capabilities of `gpt-3.5-turbo`, successfully discovered this implicit, data-driven formulation based on velocity differentiation. This finding highlights CoEvo's unique ability to identify non-traditional, contextually optimal solutions that leverage the specific structure of the evaluation process.

Convergence Analysis

In this section, we analyze the convergence behavior of our method compared to LLM-SR by examining the historical NMSE on training data and the ratio of valid solutions during the search process. The historical NMSE reveals the convergence characteristics of the algorithms, while the ratio of valid solutions indicates the efficacy of the exploration strategies employed by each method.

NMSE Convergence. Figure 5 illustrates the historical NMSE across various benchmarks for our method and LLM-SR, using two distinct LLMs. The initial NMSE reduction (within the first 1,000 iterations) is similar across all cases, indicating that the ability of LLMs to generate valid solutions is comparable at the early stages of the search process.

When employing `gpt-3.5-turbo`, CoEvo achieves a significantly lower NMSE than both LLM-SR and its `gpt-4o-mini` counterpart. This result demonstrates that

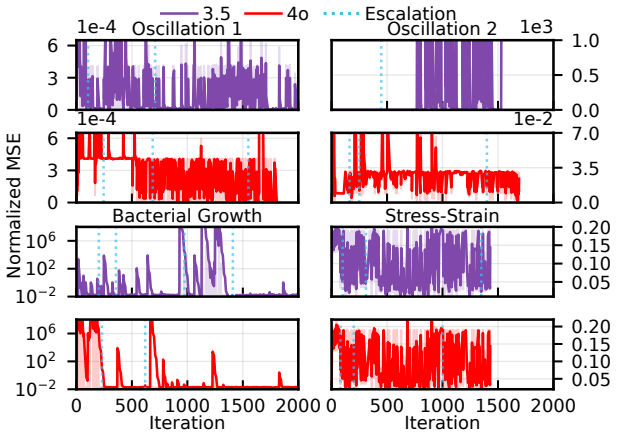


Figure 7: The NMSE of the newly sampled solutions under the influence of the knowledge. 3.5: GPT-3.5; 4o: GPT-4o-mini.

our method refines solutions more effectively over time and enables efficient exploration even with less advanced LLMs.

Valid Solution Ratio. The ratio of valid solutions generated during the search process is illustrated in Figure 6, demonstrating the exploration capabilities of our method and LLM-SR. The valid solution ratio is defined as the proportion of error-free, evaluable solutions among all sampled solutions.

Overall, CoEvo produces significantly more valid solutions than LLM-SR across all benchmarks, indicating its superior effectiveness in exploring the search space. Notably, CoEvo achieves a higher valid ratio with `gpt-3.5-turbo` than LLM-SR does with `gpt-4o-mini`, demonstrating that our method can yield high-quality solutions even with less advanced LLMs.

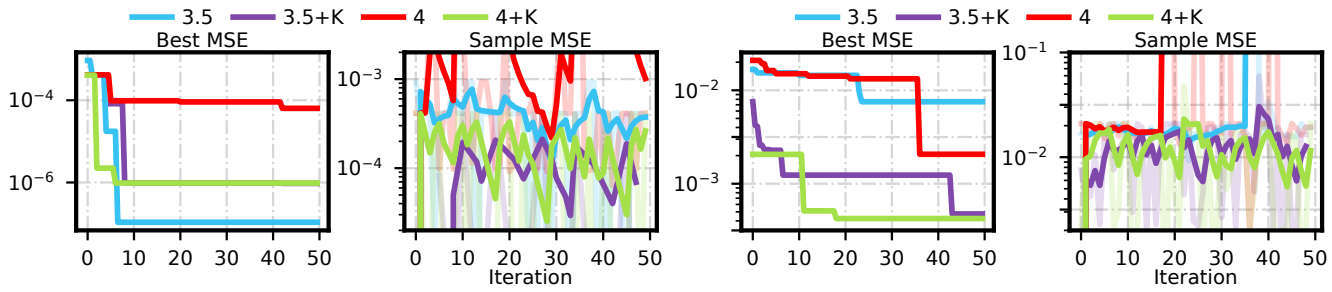
An additional observation for both methods is that, for most benchmarks, the valid solution ratio increases with more powerful LLMs. The sole exception is the Stress-Strain benchmark for CoEvo.

Knowledge Extraction and Application

This section evaluates the impact of knowledge on the quality of solutions generated by CoEvo. First, we examine solution quality across various benchmarks under the influence of dynamic knowledge managed by the knowledge library. Next, we utilize knowledge extracted from different LLMs to generate new solutions and examine its effect on CoEvo's performance. Finally, we visualize the knowledge extracted from Oscillation 2 to identify limitations of the knowledge.

Solution Quality. Figure 7 presents the NMSE of newly sampled solutions influenced by the knowledge library across different benchmarks over 2,000 iterations. The performance escalation from Figure 5 is indicated by blue dotted lines, representing the improvement of the best solution found by CoEvo.

A significant improvement in solution quality is observed in the Oscillation 1 and E. coli growth problems, where the NMSE of newly sampled solutions decreases by 2–3 orders of magnitude compared to the initial solutions. A



(a) Oscillation 1+Knowledge

(b) Bact. growth+Knowledge

Figure 8: Comparison of solution quality for Oscillation 1 and Bact. growth problems using knowledge extracted from `gpt-3.5-turbo` and `gpt-4o-mini`. Best NMSE convergence demonstrates the best NMSE achieved by the newly sampled solutions influenced by knowledge. Individual NMSE distributions show the performance of each solution.

similar improvement occurs in Oscillation 2 when using `gpt-4o-mini` knowledge. In contrast, the Stress-Strain problem is governed by a simple equation with empirically determined parameters, limiting the utility of knowledge; consequently, the NMSE of newly sampled solutions remains comparable to that of the initial solutions.

These improvements arise because the three problems are governed by fixed equations, allowing extracted knowledge to serve as informative hints for solution generation. However, in Oscillation 2, when the implicit function mentioned in the above subsection is identified, newly sampled solutions perform poorly. We further analyze this limitation in the next subsection by examining the underlying knowledge.

Knowledge and Solution Generation. To isolate the influence of the knowledge library, we separately evaluate knowledge extracted from `gpt-3.5-turbo` and `gpt-4o-mini` to generate new solutions. Specifically, we collect knowledge extracted using both LLMs for the Oscillation 1 and E. coli growth problems and use them to generate 50 new solutions for each problem.

Figure 8 illustrates the best NMSE and the individual NMSE of the newly sampled solutions influenced by knowledge from `gpt-3.5-turbo` and `gpt-4o-mini`. For individual NMSE, solutions generated with knowledge consistently outperform those without, exhibiting lower mean errors and reduced variability. For the best NMSE, knowledge-aided solutions converge more quickly, demonstrating that knowledge accelerates the discovery of better solutions.

Investigation on Knowledge of Oscillation 2. Although CoEvo is the only method that identifies the implicit function for Oscillation 2, Figure 7 shows that the NMSE of newly sampled solutions is significantly higher than that of the best solution found by CoEvo. We examine the state of the knowledge library around the discovery of the implicit function and investigate the relationship between the knowledge and solution quality.

Figure 9 illustrates the knowledge library state for Oscillation 2 during the evolutionary search process from iteration 21 (400th sample) to iteration 50 (1000th sample). This interval is selected because it includes both high- and low-quality solutions guided by knowledge, as demonstrated in

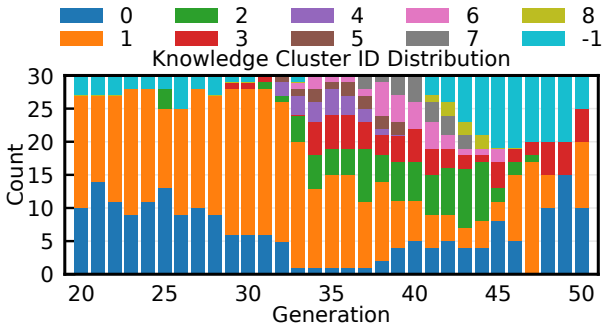


Figure 9: Evolution of the knowledge library and idea distribution for Oscillation 2 (iterations 21-49).

Figure 8. Ideas from all chosen iterations are clustered using the DBSCAN algorithm, with distinct clusters marked by different colors.

The knowledge library evolves dynamically throughout the search process. Before the implicit function is discovered, the library contains fewer categories of useful knowledge, and solutions are concentrated in regions of the search space with low NMSE. After discovery, the library expands to encompass a broader range of useful knowledge, resulting in more diverse solutions with improved NMSE. This occurs because any knowledge incorporating the concept of `numpy.gradient` proves useful for solving the problem.

Conclusion

Symbolic discovery is essential to scientific innovation, yet current methods are limited in navigating representation spaces and evolving knowledge. We introduced CoEvo, which synergizes LLMs with evolutionary search to dynamically generate and refine symbolic solutions alongside a growing knowledge library. Unlike conventional approaches that statically reuse knowledge, CoEvo enables genuine knowledge creation and refinement, pioneering open-ended innovation where solutions and understanding co-evolve to advance scientific discovery.

Acknowledgments

The work described in this paper was supported by the Research Grants Council of the Hong Kong Special Administrative Region, China [GRF Project No. CityU 11212524, 11217325].

References

Abramson, J.; Adler, J.; Dunger, J.; Evans, R.; Green, T.; Pritzel, A.; Ronneberger, O.; Willmore, L.; Ballard, A. J.; Bambrick, J.; et al. 2024. Accurate structure prediction of biomolecular interactions with AlphaFold 3. *Nature*.

Biggio, L.; Bendinelli, T.; Neitz, A.; Lucchi, A.; and Parascandolo, G. 2021a. Neural Symbolic Regression that scales. In *Proceedings of the 38th International Conference on Machine Learning, ICML*, Proceedings of Machine Learning Research, 936–945. PMLR.

Biggio, L.; Bendinelli, T.; Neitz, A.; Lucchi, A.; and Parascandolo, G. 2021b. Neural Symbolic Regression that scales. In *Proceedings of the 38th International Conference on Machine Learning, ICML*, volume 139 of *Proceedings of Machine Learning Research*, 936–945. PMLR.

Boden, M. A. 2004. *The creative mind: Myths and mechanisms*. Routledge.

Brown, S. D.; and Vranesic, Z. G. 2000. *Fundamentals of digital logic with VHDL design*, volume 70125910. McGraw-Hill New York.

Cranmer, M. D. 2023a. Interpretable Machine Learning for Science with PySR and SymbolicRegression.jl. *CoRR*, abs/2305.01582.

Cranmer, M. D. 2023b. Interpretable Machine Learning for Science with PySR and SymbolicRegression.jl. *CoRR*, abs/2305.01582.

Dubey, A.; Jauhri, A.; Pandey, A.; Kadian, A.; Al-Dahle, A.; Letman, A.; Mathur, A.; Schelten, A.; Yang, A.; Fan, A.; et al. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

Faldor, M.; Zhang, J.; Cully, A.; and Clune, J. 2024. OMNI-EPIC: Open-endedness via Models of human Notions of Interestingness with Environments Programmed in Code. arXiv:2405.15568.

Gielen, G.; and Sansen, W. M. 2012. *Symbolic analysis for automated design of analog integrated circuits*, volume 137. Springer Science & Business Media.

Huang, Y.; and Huang, J. 2024. A Survey on Retrieval-Augmented Text Generation for Large Language Models. *CoRR*.

Ifargan, T.; Hafner, L.; Kern, M.; Alcalay, O.; and Kishony, R. 2024. Autonomous LLM-Driven Research — from Data to Human-Verifiable Research Papers. *NEJM AI*, 0(0): AIoa2400555.

Kamienny, P.; d'Ascoli, S.; Lample, G.; and Charton, F. 2022. End-to-end Symbolic Regression with Transformers. In *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS*.

Kronberger, G.; Olivetti de Franca, F.; Desmond, H.; Bartlett, D. J.; and Kammerer, L. 2024. The inefficiency of genetic programming for symbolic regression. In *International Conference on Parallel Problem Solving from Nature*, 273–289. Springer.

Landajuela, M.; Lee, C. S.; Yang, J.; Glatt, R.; Santiago, C. P.; Aravena, I.; Mundhenk, T. N.; Mulcahy, G.; and Petersen, B. K. 2022. A Unified Framework for Deep Symbolic Regression. In *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS*.

Lehman, J.; and Stanley, K. O. 2011. Abandoning objectives: Evolution through the search for novelty alone. *Evolutionary computation*, 19(2): 189–223.

Lenat, D. B. 1983. EURISKO: a program that learns new heuristics and domain concepts: the nature of heuristics III: program design and results. *Artificial intelligence*, 21(1-2): 61–98.

Liu, F.; Tong, X.; Yuan, M.; Lin, X.; Luo, F.; Wang, Z.; Lu, Z.; and Zhang, Q. 2024a. Evolution of Heuristics: Towards Efficient Automatic Algorithm Design Using Large Language Model. In *Forty-first International Conference on Machine Learning, (ICML)*.

Liu, F.; Tong, X.; Yuan, M.; Lin, X.; Luo, F.; Wang, Z.; Lu, Z.; and Zhang, Q. 2024b. Evolution of Heuristics: Towards Efficient Automatic Algorithm Design Using Large Language Model. In *Forty-first International Conference on Machine Learning, ICML*. OpenReview.net.

Liu, F.; Yao, Y.; Guo, P.; Yang, Z.; Zhao, Z.; Lin, X.; Tong, X.; Yuan, M.; Lu, Z.; Wang, Z.; and Zhang, Q. 2024c. A Systematic Survey on Large Language Models for Algorithm Design. *CoRR*, abs/2410.14716.

Liu, X.; Li, P.; Suh, E.; Vorobeychik, Y.; Mao, Z.; Jha, S.; McDaniel, P.; Sun, H.; Li, B.; and Xiao, C. 2024d. AutoDAN-Turbo: A Lifelong Agent for Strategy Self-Exploration to Jailbreak LLMs. *CoRR*.

Newell, A. 1980. Physical Symbol Systems. *Cogn. Sci.*, 4(2): 135–183.

OpenAI. 2023a. GPT-4 Technical Report. *CoRR*, abs/2303.08774.

OpenAI. 2023b. Models - OpenAI API. https://platform.openai.com/docs/models\#gpt-3-5-turbo. Accessed: 2024-12-23.

OpenAI. 2024. Models - OpenAI API. https://platform.openai.com/docs/models\#gpt-4o-mini. Accessed: 2024-12-23.

Petersen, B. K.; Landajuela, M.; Mundhenk, T. N.; Santiago, C. P.; Kim, S.; and Kim, J. T. 2021. Deep symbolic regression: Recovering mathematical expressions from data via risk-seeking policy gradients. In *9th International Conference on Learning Representations, ICLR*. OpenReview.net.

Romera-Paredes, B.; Barekatain, M.; Novikov, A.; Balog, M.; Kumar, M. P.; Dupont, E.; Ruiz, F. J. R.; Ellenberg, J. S.; Wang, P.; Fawzi, O.; Kohli, P.; and Fawzi, A. 2024. Mathematical discoveries from program search with large language models. *Nat.*

Roziere, B.; Gehring, J.; Gloeckle, F.; Sootla, S.; Gat, I.; Tan, X. E.; Adi, Y.; Liu, J.; Sauvestre, R.; Remez, T.; et al. 2023. Code llama: Open foundation models for code. *arXiv preprint arXiv:2308.12950*.

Schmidt, M.; and Lipson, H. 2009. Distilling free-form natural laws from experimental data. *science*, 324(5923): 81–85.

Shojaee, P.; Meidani, K.; Gupta, S.; Farimani, A. B.; and Reddy, C. K. 2024. LLM-SR: Scientific Equation Discovery via Programming with Large Language Models. *CoRR*, abs/2404.18400.

Stanley, K. O.; Lehman, J.; and Soros, L. 2017. Open-endedness: The last grand challenge you've never heard of. *While open-endedness could be a force for discovering intelligence, it could also be a component of AI itself*.

Udrescu, S.; and Tegmark, M. 2019. AI Feynman: a Physics-Inspired Method for Symbolic Regression. *CoRR*, abs/1905.11481.

Wang, E.; Cassano, F.; Wu, C.; Bai, Y.; Song, W.; Nath, V.; Han, Z.; Hendryx, S.; Yue, S.; and Zhang, H. 2024. Planning In Natural Language Improves LLM Search For Code Generation. *CoRR*.

Wei, J.; Wang, X.; Schuurmans, D.; Bosma, M.; Xia, F.; Chi, E.; Le, Q. V.; Zhou, D.; et al. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35: 24824–24837.

Wigner, E. P. 1990. The unreasonable effectiveness of mathematics in the natural sciences. In *Mathematics and science*, 291–306. World Scientific.

Wolf, W. 2002. *Modern VLSI design: system-on-chip design*. Pearson Education.

Yao, S.; Yu, D.; Zhao, J.; Shafran, I.; Griffiths, T.; Cao, Y.; and Narasimhan, K. 2024. Tree of thoughts: Deliberate problem solving with large language models. *Advances in Neural Information Processing Systems*, 36.

Yao, S.; Zhao, J.; Yu, D.; Du, N.; Shafran, I.; Narasimhan, K. R.; and Cao, Y. 2023. ReAct: Synergizing Reasoning and Acting in Language Models. In *The Eleventh International Conference on Learning Representations, ICLR*. OpenReview.net.

Zhao, W. X.; Zhou, K.; Li, J.; Tang, T.; Wang, X.; Hou, Y.; Min, Y.; Zhang, B.; Zhang, J.; Dong, Z.; Du, Y.; Yang, C.; Chen, Y.; Chen, Z.; Jiang, J.; Ren, R.; Li, Y.; Tang, X.; Liu, Z.; Liu, P.; Nie, J.-Y.; and Wen, J.-R. 2024. A Survey of Large Language Models. arXiv:2303.18223.