EVOMO-SR: MULTIOBJECTIVE LLM-BASED EVOLUTION OF SYMBOLIC EXPRESSIONS WITH SUBSTRUCTURE GUIDANCE

Anonymous authors
Paper under double-blind review

ABSTRACT

Symbolic Regression (SR) is a data-driven method for scientific discovery which searches for interpretable analytical relationships within data. Recently, Large Language Models (LLMs) have also had a significant impact on scientific discovery, enabling the automation of various stages of the process. For these reasons, the possibility of harnessing the embedded scientific knowledge and programming capabilities of LLMs to solve SR tasks has emerged, showing promising performance compared with traditional methods. We propose EvoMO-SR, a novel LLM-driven SR framework in which the LLM generates equation skeletons, with their coefficients fitted separately by an external optimizer. The framework includes a multi-objective survival selection which controls bloating by balancing accuracy and complexity, and a substructure guidance mechanism which mutates expressions with candidate reusable building blocks. EvoMO-SR achieves the best accuracy in seven of the eight in-domain and out-of-domain settings for LSR-Synth, using a small LLM model, i.e., Llama-3.1-8B-Instruct. We also evaluated structural recovery through two symbolic accuracy metrics based on canonicalized subtree overlap and term matching, showing that our method has a greater probability of recovering highly accurate symbolic structures.

1 INTRODUCTION

Large Language Models (LLMs) are increasingly being used for scientific discovery tasks, given their vast embedded knowledge and ability to understand and generate natural language (Reddy & Shojaee, 2025; Chen et al., 2025; Song et al., 2025; Zheng et al., 2025). In this context, LLMs can be used at various stages, from literature analysis to the generation of new ideas, right through to more advanced applications such as the use of agents to automate the scientific discovery process, thereby minimizing the need for human intervention (Zheng et al., 2025). One way in which scientific discovery is enhanced is through data-driven discovery, thanks to the growing availability of extensive amounts of data and tools capable of processing it and extracting useful insights (Reddy & Shojaee, 2025). Symbolic Regression (SR), also known as *equation discovery*, can be framed as a data-driven discovery method for scientific research (Shojaee et al., 2025b) as it enables the automatic derivation of analytical relationships, revealing complex and hidden patterns within the data. Given that the resulting models make explicit the mathematical relationship between inputs and outputs, SR has gained increasing prominence in scientific discovery due to its greater interpretability compared to black-box models (Makke & Chawla, 2024). Since its advent, SR has been based primarily on Genetic Programming (GP) methods (Koza, 1994), e.g., pySR (Cranmer, 2023), GPGomea (Virgolin et al., 2021), which generate formulas in the form of expression trees, alternating between crossover and mutation operators within evolutionary loops. Moreover, methods based on deep learning have also been proposed, employing for instance deep reinforcement learning (Petersen et al., 2019) and transformers (Kamienny et al., 2023). Recently, the widespread adoption of LLMs has facilitated their integration with SR, given their extensive embedded scientific knowledge, and the ability to easily incorporate domain-specific priors (Dong & Zhong, 2025; Shojaee et al., 2025b). LLMs can act as hypothesis generators, refining expressions in evolutionary-like loops, and can be prompted with contextual information, user hints, abstract concepts, and formulas or ideas extracted in past generations (Shojaee et al., 2025a; Guo et al., 2026; Grayeli et al., 2025).

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

In this paper, we propose EvoMO-SR, an LLM-based SR framework that generates expressions in an evolutionary loop (overview of the architecture shown in Fig. 1). The LLM is responsible for generating expression skeletons in the form of Python code, similar to (Shojaee et al., 2025a), while the estimation of the coefficients is performed by a specific optimizer, in our case L-BFGS-B (Zhu et al., 1997). Generation occurs within an explicit evolutionary loop in which the LLM is provided with a context prompt and specific instructions to perform the task. Since one of the major issues already well recognized in the field of SR involves the risk of bloating, i.e., the phenomenon in which formulas uncontrollably grow in their structure (Kronberger et al., 2024), the framework implements a multi-objective survival selection (Deb, 2011), balancing accuracy and complexity. Survivors are selected from the Pareto frontier generated using the Non-Dominated Sorting and Crowding Distance algorithm (Deb et al., 2002; Smits & Kotanchek, 2005). Additionally, to help identifying candidate reusable building blocks, EvoMO-SR manages an archive of functional substructures that are extracted during evolution from generated offspring individuals. A substructure can be randomly selected from the archive based on a pre-defined probability in order to guide the LLM in the mutation of individuals.

We evaluated EvoMO-SR on the LSR-Synth dataset, which includes 129 problems, from LLM-SRBench (Shojaee et al., 2025b) and on custom benchmark problems (Shojaee et al., 2025a). By incorporating synthetic terms into known scientific models, both datasets were designed to mitigate the issue of memorization by the LLM, thereby encouraging the LLM to effectively perform data-driven SR. To this purpose, we also evaluated our method on four datasets created from randomly generated equations. Main results, obtained by using Llama-3.1-8B-Instruct, show that EvoMO-SR outperforms SR baselines in most cases, both in in-domain (ID) and in out-of-domain (OOD) settings. In details, EvoMO-SR achieves the lowest aggregate Normalized Mean Squared Error (NMSE) among the evaluated methods in seven of the eight LSR-Synth domain-split comparisons, including all four OOD. This is notable as it demonstrates that our method performs well even with a relatively small model. In addition, we also assessed the symbolic accuracy of the resulting expressions, i.e., their structural similarity to the corresponding ground truth, through two metrics. Although none of the methods achieves satisfactory symbolic accuracy, especially for LSR-Synth, EvoMO-SR showed a greater probability of recovering highly accurate symbolic structures.

The main contributions are as follows:

- We introduce EvoMO-SR, a framework for SR that employs LLMs in an evolutionary strategy to generate equation skeletons, while coefficients are fitted separately using an external optimizer. The survival mechanism is based on a multi-objective strategy to balance numerical accuracy and complexity, while reusable functional substructures are stored in a dynamic archive to enrich subsequent prompts.
- We evaluate EvoMO-SR on the full LSR-Synth benchmark and additional oscillator problems tailored to mitigate the LLM memorization issue. We compared results against LLM-SR (Shojaee et al., 2025a), LaSR (Grayeli et al., 2025) and state-of-the-art SR baselines. The framework achieves the lowest aggregate NMSE in seven of the eight ID and OOD comparisons over LSR-Synth. Furthermore, we compare our method with LLM-SR on the four datasets created from randomly generated functions, consistently showing better NMSE for ID and OOD and better symbolic accuracy in most cases.
- We enrich the evaluation with a symbolic accuracy analysis based on two deterministic metrics. Although the results show that no method is consistently able to achieve high symbolic accuracy scores, EvoMO-SR demonstrates a slightly greater ability to retrieve expressions that are very close to the ground truth.

2 Related Works

SR is a supervised machine learning method that aims to retrieve the mathematical expression that best fits a given dataset. The growing popularity of SR is due to its ability to uncover complex hidden patterns without making any prior assumptions about the functional form on the data, and to reveal the analytical relationships between features, thereby enhancing interpretability (Dong & Zhong, 2025; Makke & Chawla, 2024). While SR methods are mainly based on GP (Cranmer, 2023; Virgolin et al., 2021) and deep learning (Kamienny et al., 2023; Petersen et al., 2019), the advent of

LLMs has had an impact on the development of innovative methods for SR. One of the most representative methods is LLM-SR (Shojaee et al., 2025a), which employs LLMs to generate expression skeletons in an evolutionary-like loop, with the fitting of coefficients left to off-the-shelf optimizers. The LLM is prompted with high-performing candidates sampled via a multi-island approach from a dynamic experience buffer, prior knowledge of the problem, and task specification. The experience buffer stores diverse promising candidates, while the sampling strategy involves the selection of shorter programs in order to mitigate bloating issues. In contrast to EvoMO-SR, LLM-SR does not implement an explicit population-based evolutionary algorithm with parent-offspring survival selection. Instead, its evolutionary component mainly operates through the management and fitness-biased sampling of complete candidate expressions from the experience buffer. Wang et al. (2025) extend LLM-SR with data-aware insights and inductive idea extraction, which summarizes positive, negative, and invalid generation outcomes into reusable heuristics. These ideas are stored in a dynamic library and used to enrich subsequent prompts. Zhang et al. (2026) explore the use of LLMs in SR for the automatic design of selection operators to enforce semantic guidance and mitigate bloating. Their framework evolves selection strategies through LLMs using in-context learning to improve evolutionary SR algorithms. Each strategy is evaluated through inner SR runs across multiple datasets and selection operators survival is based on a multi-objective mechanism which balances downstream fitness and operator code length. Another relevant work is LaSR (Grayeli et al., 2025), a framework which enhances GP search of equations with natural language concepts generated via LLMs. Building on PySR, LaSR does not use LLMs as its primary equation generator, but it alternates initialization, mutation, and crossover operators with their LLM variants, extracting and evolving abstract concepts from high-performing candidates and reusing them in the equation search. LaSR explicitly adds a complexity penalty term in the fitness function, and selects Pareto-optimal solutions based on dataset loss and syntactic complexity to extract abstract ideas. However, the evolutionary process is mainly guided by traditional GP and the archive used for guidance stores natural language ideas instead of explicit mathematical pieces.

3 METHODOLOGY

3.1 PROBLEM FORMULATION

The task of Symbolic Regression (SR), also known as *equation discovery*, is to find the function that best fits a given dataset. More precisely, its goal is, given a dataset $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^{n}$ where $x_i \in \mathbb{R}^d$ is the feature vector and $y_i \in \mathbb{R}$ is the related target, to learn a symbolic functional structure $\phi(\cdot; \hat{\Theta})$ and corresponding parameter values $\hat{\Theta}$ such that $\hat{f}(x) = \phi(x; \hat{\Theta}) : \mathbb{R}^d \to \mathbb{R}$ and $\hat{f}(x_i) \approx y_i$, for $i = 1, \ldots, n$. Since SR methods typically generate multiple analytical models for the same problem, they can be seen as *hypotheses generating machines* (Kronberger et al., 2024). In our framework, we generate hypotheses using an LLM in the form of equation skeletons, leaving the estimation of parameters to specific optimizers. Formally, the LLM generates a set of candidate equation skeletons $\Phi = \{\phi_j\}_{j=1}^{M}$, where each ϕ_j represents a symbolic expression with unestimated numerical parameters. The parameters of the equations are estimated separately by optimizing a fitting loss $\mathcal{L}$ over $\mathcal{D}$. The set of fitted candidate hypotheses is then $\{\hat{f}_j(x) \mid \hat{f}_j(x) = \phi_j(x; \hat{\Theta}_j)\}_{j=1}^{M}$.

3.2 EVOMO-SR METHODOLOGY

EvoMO-SR performs SR employing an LLM to generate and refine equation skeletons within an evolutionary loop (see main components of the architecture in Fig. 1). The initial population of equation skeletons $\Phi = \{\phi_j\}_{j=1}^{M}$, where M denotes the number of solutions in the parent population, is generated based on a set of prompts $\mathcal{P}_{\text{init}} = \{\text{role} + \text{task}, \text{context}, \text{format}\}$ that are concatenated. The context is specific to the dataset $\mathcal{D}$ under consideration, and is intended to support equation generation by incorporating information such as the physical meaning of the variables. The LLM's output includes not only the equation skeleton (in the form of Python code) but also an explanation of why that specific structure was proposed. These explanations are saved as textual descriptions of each equation, alongside the proposed formula, so as to enrich subsequent prompts. Figure 2 shows an example of an explanation given by the LLM for the generation of an equation skeleton. These descriptions are not exploited only to provide the LLM with useful information about previously generated candidates, but also to make the process more interpretable. In fact, in this way, we

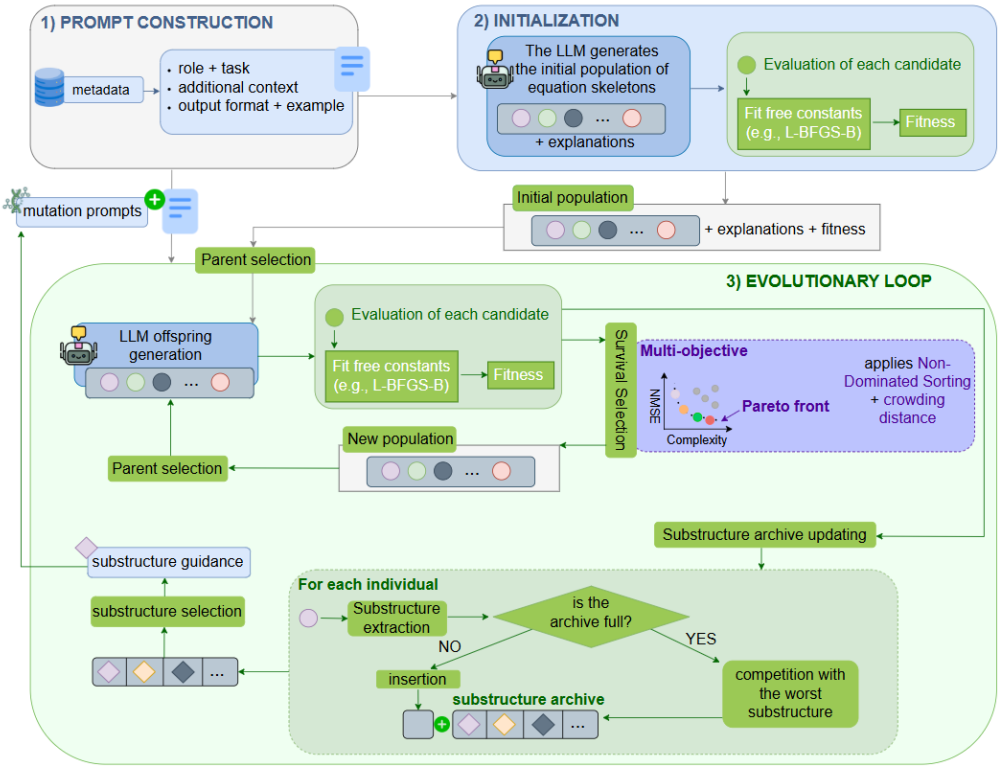


Figure 1: EvoMO-SR main components. **1) Prompt Construction**: the first step includes the construction of the prompt by role specification, task description, additional context based on considered dataset, and output and example format. **2) Initialization**: the initial population is created by prompting the LLM with the constructed prompt, and each candidate is fitted and evaluated. **3) Evolutionary Loop**: in subsequent generations, the LLM is prompted with the initial prompt enriched with the description, fitness and equation of the selected candidate, plus a mutation prompt randomly selected from the pre-fixed set. To form the new population, the fitted and evaluated set of candidates faces survival selection, which is on multi-objective optimization. At the end of each generation, each individual is scanned to extract added substructures and the substructure archive is updated accordingly.

can trace the decisions made by the LLM for each generated individual, thus understanding the motivations behind the proposal or refinement of expressions.

Once the generation ends, each equation in the initial population is fitted with an external optimizer, in our case L-BFGS-B, and evaluated with a fitness metric on training data. At this point, the actual evolutionary loop begins, starting with the mutation of individuals selected from the initial population $\Phi = \{\phi_j\}_{j=1}^{M}$. The parent selection strategy, which decides the individuals to mutate from the current population, can be customized with the desired mechanism. In our case, individuals are selected by a "random with replacement" strategy. Details on additional evolutionary components and parameter settings are available in Appendix A.1. An individual is mutated by explicitly prompting the LLM with a specific mutation prompt, chosen at random at each iteration (where each iteration corresponds to the mutation of one individual in a generation). In our main setting, the possibilities are whether to refine the current expression or to propose a new one. So, in order to generate the new offspring population, the LLM is prompted with a set of prompts $\mathcal{P}_{\text{loop}} = \mathcal{P}_{\text{init}} \cup \{\mathcal{I}_j, \texttt{mutation}\}$, where $\mathcal{I}_j$ is the current selected individual from the parent population that includes its textual description and fitness. The generated skeletons $\Phi = \{\phi_k\}_{k=1}^{N}$, where N denotes the number of solutions in the offspring population, are then fitted and evaluated. The new population is created based on the multi-objective survival strategy. On this set of individuals, the non-dominated sorting algorithm is applied to form Pareto fronts based on two opposing objectives: accuracy (i.e., NMSE) and complexity (i.e., count of nodes in the expression tree, following Landajuela et al. (2022);

Prompt

```
Give the response in the format:
# Description: <short-description of the meaning of the analytical relationships and
why you gave this expression. Also include the term(s) you have added, deleted and/or
changed.>
```

LLM Response

```
The discovered equation represents a damped nonlinear oscillator system with driving
force, given data on position and velocity. The equation is a simple model that captures
the underlying physics of the system. I have added a term to represent the driving force
and changed the equation to include a non-linear damping term.
```

Figure 2: Example of the feedback generated by the LLM.

La Cava et al. (2021); Cranmer (2023)). In detail, NMSE is defined as NMSE $= \frac{\sum_{i=1}^{N}(y_i-\hat{y}_i)^2}{\sum_{i=1}^{N}(y_i-\bar{y})^2}$, where $\bar{y}$ denotes the mean target value. Lower values indicate better predictive accuracy, with zero corresponding to a perfect fit. The complexity is computed as the number of nodes of the Python Abstract Syntax Tree (AST) of the discovered expression after coefficient optimization. The next population is formed by taking individuals front by front until the maximum number of individuals to be retained is reached. If the number of individuals on a front exceeds the remaining space in the population being constructed, the crowding distance criterion is applied, measuring how isolated a solution is in space. The algorithm favours solutions with a higher crowding distance, in order to promote greater diversity. A Pareto Archive is maintained to keep trace of all the best trade-offs candidates found during the whole search. The new population is then given as input to the next generation. The evolutionary loop ends when the pre-fixed number of generations is reached.

At the end of each generation, each individual in the offspring population is scanned to extract added substructures. Here we refer as a *substructure* an additive subtree of the input expression tree. This extraction enables to detect new functional structures added by the LLM during the evolution, thus identifying possible useful and reusable components. In order to keep trace of these substructures, the system maintains a dynamic fixed-length archive that is updated at each generation. When a new substructure is detected in an individual, it is checked whether there is space left in the archive. If yes, the new substructure s' is added to the archive; otherwise, s' competes for the insertion against the worst substructure s_{worst} in the archive. Since for each substructure is stored the NMSE of the related offspring, s' replaces s_{worst} if and only if the NMSE related to s' is lower than the one of s_{worst}. The archive of substructures occasionally serves to enrich mutation prompts based on a probability p_{sub}. When this happens, a substructure is randomly selected from the archive and is used for substructure guidance by telling the LLM to use that substructure in the mutation.

3.3 Symbolic Accuracy metrics

The evaluation of the accuracy for SR problems should consider the Symbolic Accuracy (SA), i.e., how similar is the expression found to the ground truth equation. We evaluate SA by computing the structural similarity between each discovered equation $\hat{f}^\star$ and the ground-truth formula $f^\star$. To this purpose, we consider two complementary structural metrics: an AST-based subtree similarity (SA_{AST}) and an additive term-based similarity (SA_{Term}). The first metric measures the degree of structural overlap between $\hat{f}^\star$ and $f^\star$ with the Jaccard similarity of their subtree signatures, similarly to (Kahlmeyer et al., 2025). First, both $\hat{f}^\star$ and $f^\star$ are preprocessed to obtain comparable forms by unifying the syntax of operators, and, optionally, applying SymPy-based canonicalization[1]. Each expression is then parsed into an Abstract Syntax Tree (AST). Further normalization steps include transforming subtraction and division operators into canonical additive/multiplicative forms, sorting of commutative operands, and replacement of numeric constants by placeholders, so that structurally identical/similar formulas with different fitted constants can be fairly compared. Finally, the set of serialized subtree signatures is extracted from the two canonical ASTs, and the SA_{AST} is computed

[1] https://docs.sympy.org/latest/index.html

with their Jaccard similarity. So, given $\mathcal{T}(f)$ being the set of subtree signatures extracted from f, we define the (SA_{AST}) between $\hat{f}^\star$ and $f^\star$ as follows:

$$SA_{AST}(\hat{f}^*, f^*) = \frac{\left|\mathcal{T}(\hat{f}^\star) \cap \mathcal{T}(f^\star)\right|}{\left|\mathcal{T}(\hat{f}^\star) \cup \mathcal{T}(f^\star)\right|}.$$

This metric ranges from 0 to 1, with larger values indicating greater overlap between the hierarchical structures of the two expressions. A value of 0 indicates no overlap between subtrees and a value of 1 indicates complete overlap and therefore perfect structural equivalence between the two expressions.

We also computed SA_{Term}, a Jaccard similarity metric based on the matching additive building blocks (or terms) of the expressions. After the canonicalization of both $\hat{f}^\star$ and $f^\star$, the additive terms are exposed and, for each of them, variable-independent multiplicative factors are removed, so that fitted numerical coefficients do not affect the comparison. Let $\mathcal{T}_{Term}(f)$ denote the set of normalized additive-term signatures extracted from f. The term-based SA_{Term} is defined as follows:

$$SA_{Term}(\hat{f}^*, f^*) = \frac{\left|\mathcal{T}_{Term}(\hat{f}^\star) \cap \mathcal{T}_{Term}(f^\star)\right|}{\left|\mathcal{T}_{Term}(\hat{f}^\star) \cup \mathcal{T}_{Term}(f^\star)\right|}.$$

The term-based metric therefore measures how many of the normalized additive components of the ground-truth expression are also present in the discovered equation. It ranges from 0 to 1, where 0 denotes no shared normalized terms and 1 indicates that the two expressions contain the same set of normalized additive terms. The SA_{Term} metric is less sensitive to additional or structurally different components, as it focuses on matching normalized additive terms. Consequently, it gives relatively more credit to equations that recover correct ground-truth substructures even when they also contain spurious terms, while SA_{AST} tends to penalize more severely. More details in Appendix A.3.

4 Experimental results

The main experiments were conducted using Llama-3.1-8B-Instruct as the LLM backbone, with comparisons against two LLM-based baselines, namely LLM-SR (Shojaee et al., 2025a) and LaSR (Grayeli et al., 2025), and other state-of-the-art SR baselines. For each dataset, we ran EvoMO-SR for 100 generations, resulting in a total of 1'000 generated expressions, optimizing parameters via L-BFGS-B, and setting the model's temperature to 0.7. The dimension of the archive of substructures was set to 10 with $p_{sub} = 0.5$.

4.1 Comparisons over LSR-Synth dataset

Since LLM-based SR methods may rely on the memorization of well-known formulas instead of performing real data-driven SR, Shojaee et al. (2025b) proposed LSR-Synth, a dataset including a total number of 129 problems across chemistry, biology, physics, and material science domains. The ground-truth equations have been generated from the original formulas by adding new synthetic terms. The OOD split is generated taking the final 10% of samples (500 samples) ordered by time (or temperature for material science problems), while the remaining 90% (4500 samples) is divided into training and ID sets. To ensure robustness, we ran our method and each baseline 5 times and averaged the obtained results. Table 1 shows the averages of the median NMSE ID and NMSE OOD values for 5 runs for EvoMO-SR, LLM-SR, LaSR, and the other SR baselines. We chose well-known SR methods that have been extensively benchmarked in the SR community (de Franca et al., 2024; Imai Aldeia et al., 2025), such as pySR (Cranmer, 2023), GPGomea (Virgolin et al., 2021), Operon (Burlacu et al., 2020), and uDSR (Landajuela et al., 2022). EvoMO-SR outperforms all baselines in seven of the eight domain-split comparisons, with LLM-SR resulting the best competitor, performing slightly better in the ID split for the Physics domain. Compared to LLM-SR, EvoMO-SR reduces aggregate OOD NMSE by factors of approximately 14 in Physics, 53 in Chemistry, 37 in Biology, and 2.4×10^3 in Material Science. These improvements are substantial, but they

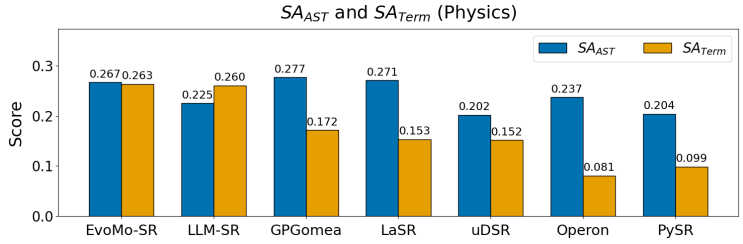


Figure 3: Comparison of SA_{AST} and SA_{Term} between EvoMO-SR and the other SR baselines over LSR-Synth Physics dataset.

do not compare results over individual problems. To better analyze the differences in performance between EvoMO-SR and LLM-SR, which is our main competitor, we performed paired Wilcoxon signed-rank tests using the benchmark problems as paired experimental units, considering ID and OOD performance separately for each domain. The p-values (adjusted with Holm's correction) are always below 0.05, except for the Physics ID dataset, indicating statistically significant differences in performance in all other cases. Detailed results are shown in Appendix C.4.

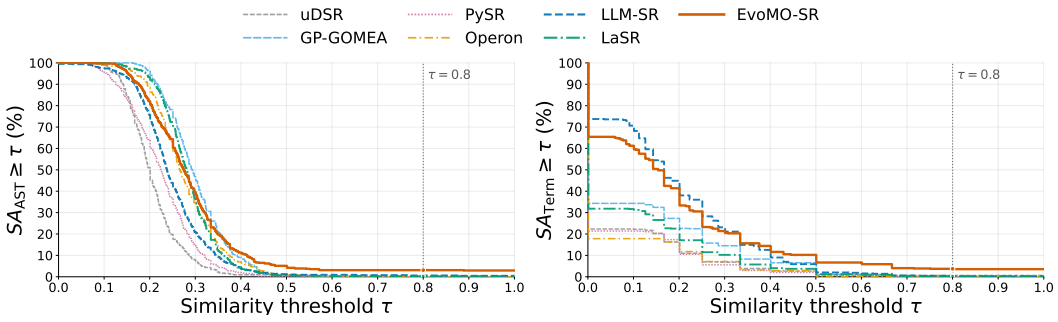
The improvements in numerical accuracy do not directly imply better symbolic recovery of the ground truth expressions. Figure 3 shows the average values of both SA metrics across all benchmarked methods over the LSR-Synth Physics dataset. The SA values are consistently low for every method, with GPGomea being the best for SA_{AST} and EvoMO-SR being the best for SA_{Term}. A more detailed analysis is presented in Figure 4, where we show the distributions of SA values according to a threshold τ based on the results obtained over the LSR-Synth datasets (129 problems for 5 runs). For SA_{AST} the curves are concentrated between roughly 0.15 and 0.4, and there is no single method dominating over this range. GP-GOMEA appears competitive, with one of the strongest curves around 0.2-0.3. EvoMO-SR becomes more dominant in the right tail, and around $\tau \approx 0.35$ its curve becomes superior to the baselines, while the other methods collapse toward zero near $\tau = 1$. For SA_{Term}, LLM-SR is stronger than EvoMO-SR in retrieving similar expressions at very low thresholds. Around $\tau \geq 0.3$, EvoMO-SR becomes the strongest method, with the advantage being clearer in the upper tail. These results indicate that EvoMO-SR's main advantage is not necessarily a uniform increase in structural similarity, but a greater probability of recovering highly accurate symbolic structures.

4.2 Performance on Scientific Discovery Benchmarks

We further evaluated EvoMO-SR on the scientific discovery benchmarks introduced by Shojaee et al. (2025a). Among the scientific domains that the dataset covers, we choose the two physics datasets, i.e., `oscillator1` and `oscillator2`. The training and ID samples are interleaved

Table 1: Comparison of Median NMSE ID and OOD values between EvoMO-SR against LLM-SR and other SR baselines over LSR-Synth datasets (44 for Physics, 36 for Chemistry, 24 for Biology, 25 for Material Science) averaged across 5 runs. EvoMO-SR, LLM-SR and LaSR have been run with Llama-3.1-8B-Instruct backbone. Lower NMSE ID/OOD (↓) values are better.

Method	**Physics**		**Chemistry**		**Biology**		**Material Science**	
	ID ↓	**OOD ↓**	**ID ↓**	**OOD ↓**	**ID ↓**	**OOD ↓**	**ID ↓**	**OOD ↓**
uDSR	1.15e-1	9.30e-2	1.07e-1	86.11	1.99e-1	1.29	6.23e-2	3.28
pySR	3.23e-1	2.30e-1	2.63e-1	180.94	8.47e-1	366.68	2.50e-1	25.93
GPGomea	4.11e-2	4.05e-2	1.97e-3	2.59	2.29e-1	23.52	1.22e-3	1.22e-1
Operon	1.58e-1	1.51e-1	1.04e-2	7.54	3.42e-1	49.43	1.82e-2	1.11
LLM-SR	**4.10e-5**	7.65e-4	1.14e-5	3.34e-1	7.23e-5	5.29e-1	2.20e-5	6.67e-3
LaSR	9.69e-3	3.88e-2	3.03e-4	3.41	2.70e-3	1.21	2.84e-4	3.47e-2
EvoMO-SR	5.88e-5	**5.47e-5**	**2.32e-6**	**6.32e-3**	**9.65e-6**	**1.43e-2**	**1.40e-8**	**2.73e-6**



(a) SA_{AST} threshold analysis.

(b) SA_{Term} threshold analysis.

Figure 4: Percentage of recovered expressions across all LSR-Synth datasets and runs (for a total of 645 cases) as a function of the threshold τ for SA_{AST} and SA_{Term}.

Table 2: Direct comparison of median NMSE ID, NMSE OOD and SA values over 20 runs with LLM-SR and LaSR over `oscillator1` and `oscillator2` datasets.

Method	**`oscillator1`**				**`oscillator2`**			
	ID ↓	**OOD** ↓	SA_{AST} ↑	SA_{Term} ↑	**ID** ↓	**OOD** ↓	SA_{AST} ↑	SA_{Term} ↑
LLM-SR	1.55e-5	2.61e-2	0.19	0.28	2.85e-5	8.25e-3	**0.23**	0
LaSR	4.37e-3	5.02e-1	**0.27**	0	2.00e-1	1.15e-1	0.12	0
EvoMO-SR	**1.53e-6**	**1.80e-2**	0.17	**0.35**	**1.69e-6**	**2.43e-4**	0.16	**0.33**

over $t \in [30, 50]$ (10'000 samples for the train set and 10'000 samples for the ID set), while OOD samples cover $t \in [0, 20]$ (10'000 samples), where t is the simulation time. Results are presented in Table 2. EvoMO-SR outperforms both methods in accuracy achieving lower NMSE ID and OOD across 20 runs. Concerning the symbolic similarity with the ground truth, LaSR obtained the best value for SA_{AST} in the `oscillator1` dataset, but its median SA_{Term} is 0 with around the 68% of expressions having zero matching substructures with the ground truth. Even LLM-SR, despite achieving a higher SA value than EvoMO-SR on `oscillator2` and obtaining acceptable values for the other metric on `oscillator1` as well, fails to identify correct building blocks in the final expressions for `oscillator2`, resulting in a median $SA_{Term} = 0$ with the 60% of expressions having no common additive terms with the ground truth. To assess whether these differences are consistent across independent runs, we additionally performed paired Wilcoxon signed-rank tests between EvoMO-SR and LLM-SR. Detailed results are reported in Appendix C.4. For NMSE, EvoMO-SR achieves lower values in the majority of paired runs, although the differences against LLM-SR do not reach statistical significance. The analysis over SA values shows a clearer distinction: SA_{AST} significantly favors LLM-SR on both datasets, while SA_{Term} significantly favors EvoMO-SR on `oscillator2`.

4.3 Performance over randomly generated functions

Although the datasets used before were designed to mitigate the phenomenon of memorization in LLMs, we still wanted to test our method on datasets generated by randomly constructed mathematical expressions. For each expression, we generated training and ID/OOD samples. Results in Table 3 show the comparison between our method and LLM-SR. The four ground truth expressions have an increasing degree of complexity, with the easiest (P1) retrieved by EvoMO-SR with median $SA_{Term} = 1$.

4.4 Ablation studies

The effectiveness of the substructure guidance mechanism has been evaluated by removing the substructure guidance module from the EvoMO-SR. To best test the differences in performance, we conducted this study using two larger models as well, namely DeepSeek-v4-Flash and GPT-5.4-nano.

Table 3: Median values of NMSE ID/OOD and SA metrics obtained by EvoMO-SR and LLM-SR across datasets generated with the following ground truth expressions: $1.88 \sin x_1 + 1.61 \cos(x_0)$ (P1), $1.10x_0^3 + 2.02x_0^2 - 1.69x_1$ (P2), $-1.95x_0^2 - 0.35x_1 \sin x_1 + 2.36x_2^2 + 2.88$ (P3), $1.90x_1^2 + 2.38x_1x_2 - 2.16 \sin x_0 - 1.08 \cos x_0 - 2.23 \cos x_2 - 2.10$ (P4).

Problem	Method	ID ↓	OOD ↓	SA_{AST} ↑	SA_{Term} ↑
P1	**EvoMO-SR**	**2.56e-32**	**1.40e-32**	**0.26**	**1.00**
	LLM-SR	2.49e-10	4.69e-8	0.19	0.14
P2	**EvoMO-SR**	**7.03e-15**	**1.94e-15**	0.22	**0.75**
	LLM-SR	5.89e-12	4.06e-11	0.22	0.37
P3	**EvoMO-SR**	**2.20e-7**	**4.02e-5**	0.11	0.18
	LLM-SR	2.06e-5	1.56e-3	**0.20**	0.18
P4	**EvoMO-SR**	**3.63e-6**	**1.66e-3**	0.13	**0.40**
	LLM-SR	1.23e-3	5.64e-2	**0.19**	0.17

Table 4: Performance comparison across LLM backbones (Llama-3.1-8B-Instruct, DeepSeek-v4-Flash, GPT-5.4-nano) with substructure guidance (**with sub**) and without substructure guidance (**w/o sub**) over `oscillator1` and `oscillator2` datasets.

Model	`oscillator1`				`oscillator2`			
	w/o sub		**with sub**		**w/o sub**		**with sub**	
	ID ↓	**OOD ↓**	**ID ↓**	**OOD ↓**	**ID ↓**	**OOD ↓**	**ID ↓**	**OOD ↓**
Llama8B	1.39e-5	**2.06e-2**	**4.79e-6**	2.11e-2	1.98e-5	3.83e-3	**8.45e-6**	**1.60e-3**
DeepSeek-v4-Flash	2.26e-6	2.98e-2	**9.01e-7**	**1.32e-2**	4.43e-4	1.88e-1	**1.62e-4**	**2.76e-3**
GPT-5.4-nano	5.34e-6	3.15e-2	**9.13e-7**	**1.09e-2**	1.28e-3	1.94e-2	**8.37e-7**	**2.06e-4**

Table 4 shows the results of this experiment over `oscillator1` and `oscillator2` datasets. The superiority of the substructure guidance approach is more highlighted for bigger models, especially for the `oscillator2` dataset with GPT-5.4-nano. Additional ablation studies are shown in Appendix C.3.

5 Conclusions, Limitations, and Future Work

We introduced EvoMO-SR, an LLM-driven SR framework that combines multi-objective survival selection with the extraction and reuse of symbolic substructures during evolution. On the LSR-Synth benchmark, EvoMO-SR achieves the lowest aggregate NMSE in seven of the eight domain-split comparisons, including all four OOD settings using a small model (i.e., LLama-3.1-8B-Instruct). The results obtained on the 4 datasets created from randomly generated expression show that the advantage persists even when the ground truth equations are not derived from known scientific models. The analysis of structural similarity highlights the importance of distinguishing numerical accuracy from the recovery of the symbolic ground truth. EvoMO-SR improves term matching recovery in several settings and shows greater probabilities of recovering highly similar expressions, although its advantages are not uniform across similarity metrics or scientific domains. The main limitations first concerns symbolic accuracy, which is consistently low even for EvoMO-SR, leaving much room for improvement in future developments. Furthermore, the substructure-guidance mechanism associates substructures with the performance of the related expression, which does not isolate their individual contribution. Future work will investigate more refined management of the substructure archive, such as more informative scores to the extracted substructures.

AI use statement

In this work, we used AI to revise the text and generate part of the code. We have reviewed all AI-assisted work. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

Ethics statement

This work presents an LLM-driven method for generating symbolic expressions and has demonstrated good performance on datasets spanning various scientific domains (namely physics, chemistry, biology, and materials science). It is important to clarify that the equations proposed by EvoMO-SR should be considered hypotheses and not scientifically established solutions. The evaluation of the results obtained should always be accompanied by validation by domain experts.

Reproducibility statement

The EvoMO-SR codebase is available at the following anonymous repository: `https://anonymous.4open.science/r/EvoMO-SR-47B6`.
Section 3 describes the methodology of the approach; Section 4 describes the main experimental settings; Appendices A.1 and A.2 report the evolutionary settings and details on the prompts, respectively.

References

Bogdan Burlacu, Gabriel Kronberger, and Michael Kommenda. Operon c++ an efficient genetic programming framework for symbolic regression. In *Proceedings of the 2020 genetic and evolutionary computation conference companion*, pp. 1562–1570, 2020.

Qiguang Chen, Mingda Yang, Libo Qin, Jinhao Liu, Zheng Yan, Jiannan Guan, Dengyun Peng, Yiyan Ji, Hanjing Li, Mengkang Hu, et al. Ai4research: A survey of artificial intelligence for scientific research. *arXiv preprint arXiv:2507.01903*, 2025.

Miles Cranmer. Interpretable machine learning for science with pysr and symbolicregression. jl. *arXiv preprint arXiv:2305.01582*, 2023.

Fabricio O de Franca, Marco Virgolin, M Kommenda, MS Majumder, M Cranmer, G Espada, L Ingelse, A Fonseca, M Landajuela, B Petersen, et al. Srbench++: principled benchmarking of symbolic regression with domain-expert interpretation. *IEEE transactions on evolutionary computation*, 2024.

Kalyanmoy Deb. Multi-objective optimisation using evolutionary algorithms: an introduction. In *Multi-objective evolutionary optimisation for product design and manufacturing*, pp. 3–34. Springer, 2011.

Kalyanmoy Deb, Amrit Pratap, Sameer Agarwal, and TAMT Meyarivan. A fast and elitist multiobjective genetic algorithm: Nsga-ii. *IEEE transactions on evolutionary computation*, 6(2): 182–197, 2002.

Junlan Dong and Jinghui Zhong. Recent advances in symbolic regression. *ACM Computing Surveys*, 57(11):1–37, 2025.

Arya Grayeli, Atharva Sehgal, Omar Costilla-Reyes, Miles Cranmer, and Swarat Chaudhuri. Symbolic regression with a learned concept library. In *Proceedings of the 38th International Conference on Neural Information Processing Systems*, NIPS '24, Red Hook, NY, USA, 2025. Curran Associates Inc. ISBN 9798331314385.

Ping Guo, Qingfu Zhang, and Xi Lin. Coevo: Continual evolution of symbolic solutions using large language models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pp. 1810–1818, 2026.

Guilherme Seidyo Imai Aldeia, Hengzhe Zhang, Geoffrey Bomarito, Miles Cranmer, Alcides Fonseca, Bogdan Burlacu, William G La Cava, and Fabrício Olivetti de França. Call for action: towards the next generation of symbolic regression benchmark. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, pp. 2529–2538, 2025.

Paul Kahlmeyer, Joachim Giesen, Michael Habeck, and Henrik Voigt. Scaling up unbiased search-based symbolic regression. *arXiv preprint arXiv:2506.19626*, 2025.

Pierre-Alexandre Kamienny, Guillaume Lample, Sylvain Lamprier, and Marco Virgolin. Deep generative symbolic regression with monte-carlo-tree-search. In *International Conference on Machine Learning*, pp. 15655–15668. PMLR, 2023.

Aobo Kong, Shiwan Zhao, Hao Chen, Qicheng Li, Yong Qin, Ruiqi Sun, Xin Zhou, Enzhi Wang, and Xiaohang Dong. Better zero-shot reasoning with role-play prompting. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pp. 4099–4113, 2024.

John R Koza. Genetic programming as a means for programming computers by natural selection. *Statistics and computing*, 4(2):87–112, 1994.

Gabriel Kronberger, Bogdan Burlacu, Michael Kommenda, Stephan M Winkler, and Michael Affenzeller. *Symbolic Regression*. CRC Press, 2024.

William La Cava, Bogdan Burlacu, Marco Virgolin, Michael Kommenda, Patryk Orzechowski, Fabrício Olivetti de França, Ying Jin, and Jason H Moore. Contemporary symbolic regression methods and their relative performance. *Advances in neural information processing systems*, 2021 (DB1):1, 2021.

Mikel Landajuela, Chak Lee, Jiachen Yang, Ruben Glatt, Claudio P. Santiago, Ignacio Aravena, Terrell N. Mundhenk, Garrett Mulcahy, and Brenden K. Petersen. A unified framework for deep symbolic regression. In *Advances in Neural Information Processing Systems*, 2022.

Nour Makke and Sanjay Chawla. Interpretable scientific discovery with symbolic regression: a review. *Artificial Intelligence Review*, 57(1):2, 2024.

Brenden K Petersen, Mikel Landajuela, T Nathan Mundhenk, Claudio P Santiago, Soo K Kim, and Joanne T Kim. Deep symbolic regression: Recovering mathematical expressions from data via risk-seeking policy gradients. *arXiv preprint arXiv:1912.04871*, 2019.

Chandan K Reddy and Parshin Shojaee. Towards scientific discovery with generative ai: Progress, opportunities, and challenges. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pp. 28601–28609, 2025.

Parshin Shojaee, Kazem Meidani, Shashank Gupta, Amir Barati Farimani, and Chandan K. Reddy. Llm-sr: Scientific equation discovery via programming with large language models. *ICLR*, 2025a.

Parshin Shojaee, Ngoc-Hieu Nguyen, Kazem Meidani, Amir Barati Farimani, Khoa D Doan, and Chandan K Reddy. Llm-srbench: A new benchmark for scientific equation discovery with large language models. *arXiv preprint arXiv:2504.10415*, 2025b.

Guido F Smits and Mark Kotanchek. Pareto-front exploitation in symbolic regression. In *Genetic programming theory and practice II*, pp. 283–299. Springer, 2005.

Zhangde Song, Jieyu Lu, Yuanqi Du, Botao Yu, Thomas M Pruyn, Yue Huang, Kehan Guo, Xiuzhe Luo, Yuanhao Qu, Yi Qu, et al. Evaluating large language models in scientific discovery. *arXiv preprint arXiv:2512.15567*, 2025.

Marco Virgolin, Tanja Alderliesten, Cees Witteveen, and Peter A. N. Bosman. Improving model-based genetic programming for symbolic regression of small expressions. *Evolutionary Computation*, 29(2):211–237, 2021.

Rui Wang, Fei Mi, Yi Chen, Boyang Xue, Hongru Wang, Qi Zhu, Kam-Fai Wong, and Ruifeng Xu. Role prompting guided domain adaptation with general capability preserve for large language models. In *Findings of the Association for Computational Linguistics: NAACL 2024*, pp. 2243–2255, 2024.

Runxiang Wang, Boxiao Wang, Kai Li, Yifan Zhang, and Jian Cheng. Drsr: Llm based scientific equation discovery with dual reasoning from data and experience. *arXiv preprint arXiv:2506.04282*, 2025.

Hengzhe Zhang, Qi Chen, Bing Xue, Wolfgang Banzhaf, and Mengjie Zhang. Llm-meta-sr: In-context learning for evolving selection operators in symbolic regression. *IEEE Transactions on Evolutionary Computation*, 2026.

Tianshi Zheng, Zheye Deng, Hong Ting Tsang, Weiqi Wang, Jiaxin Bai, Zihao Wang, and Yangqiu Song. From automation to autonomy: A survey on large language models in scientific discovery. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 17744–17761, 2025.

Ciyou Zhu, Richard H Byrd, Peihuang Lu, and Jorge Nocedal. Algorithm 778: L-bfgs-b: Fortran subroutines for large-scale bound-constrained optimization. *ACM Transactions on mathematical software (TOMS)*, 23(4):550–560, 1997.

A Implementation details

A.1 Parameters setting

EvoMO-SR generates symbolic expressions in a complete population-based evolutionary loop, with multiple customizable evolutionary parameters. The population size is regulated through the parameter `n_parents`, while the number of offspring evolved at each generation is controlled by `n_offspring`. In our implementation, both `n_parents` and `n_offspring` are equal to 10. Another important parameter is the number of generations, which establishes how many parent-offspring evolutions are performed. Given that `n_offspring` is equal to 10 and the number of generations is set to 100, the total number of equations produced is 1'000. The expressions are evolved based on the evaluation of a fitness function, which in our case is the Normalized Mean Squared Error (NMSE). This fitness assigns scores at the very beginning of the algorithm (when the initial population is created) and then each time an offspring individual is generated. However, the selection of survival expressions is not solely based on accuracy. If `multi_objective` is `True`, then the multi-objective survival mechanism is enabled. This module can be customized with any objectives. In our main implementation, the objectives are the NMSE and the count of math nodes, i.e., the accuracy and complexity, respectively. The parameter `elitism` decides which set of individuals undergoes Non-Dominated Sorting, and thus multi-objective survival selection. If it is enabled, then both parent and offspring are included for survival; otherwise survival considers only offspring. We choose `elitism=True` as it prevents good non-dominated parent to be excluded from the population. When the generation is over, the substructure archive is updated. Its default dimension is set to 10 and the guidance probability to $p_{sub} = 50\%$. In this regard, the mutation prompts are fully customizable. In our main implementation, at each iteration (which corresponds to one mutation) it is randomly selected a mutation prompt among two: "Refine the current expression while preserving useful terms." or "Propose a new expression completely different from the current one.". The individual chosen for mutation is selected through a parent selection strategy, regulated with an appropriate parameter. In this paper we choose `parent_selection = "random"`, corresponding to the *random with replacement* strategy. Other two possible strategies are *tournament selection* and *roulette*. With a fixed rate (the 50%), the mutation prompt is enriched with substructure guidance (see Appendix A.2 for additional details on the prompts used). Other parameters determine which substructures are suitable for inclusion and which are not. If a substructure is too complex and long, it is rejected because it deviates from being a building block and comes closer to being a complete expression. Similarly, substructures that are equivalent to individual constants or variables are excluded. In general, substructures are extracted from each parent-offspring pair by detecting additive terms that are present in the offspring expression and not in the parent expression. The expressions are first canonicalized with Sympy and products are expanded when necessary to expose hidden additive terms. The detected substructures are converted back to NumPy syntax before being considered for the archive. To avoid duplicates, each accepted substructure is mapped to a unique key in which learnable coefficients `c[i]` are replaced by a common placeholder and operands of commutative operations such as addition and multiplication are sorted.

The full implementation of EvoMO-SR includes other parameters typical of evolutionary search, though they are not relevant to this paper. These parameters are used to further customize the search, making our framework modular. For example, it is possible to include niching mechanisms, such as

Task Prompt

```
You are an expert specializing in Symbolic Regression.
I have a dataset Z with 2 features (from Z[:,0] to Z[:,1]) and a target variable y.
Your task is to discover the mathematical function that maps these features to y.
Write a Python function named `discovered_equation(Z,c)` where:
    - `Z` is a 2D numpy array with shape (N,2) .
    - `c` is a 1D numpy array of learnable constants.
Return ONLY the function using numpy (as np). Do not use features beyond index 1.
The optimization of the constants will be left to an external optimizer.
CRITICAL: You must balance accuracy and complexity. Shorter, elegant equations that
capture the underlying physics are strongly preferred over overly complex ones.
```

Figure 5: The task prompt we use, which contains the specification of the role and the description of the equation discovery task.

Scientific context

```
This is a Scientific Equation Discovery task.
Task description:
Find the mathematical function skeleton that represents acceleration in a damped
nonlinear oscillator system with driving force, given data on position, and velocity.
Here is the symbolic meaning of the input variables:
Z[:,0] represents symbol 'x': current position
Z[:,1] represents symbol 'v': velocity
```

Figure 6: One example of scientific prompt. Here the context extracted from metadata of `oscillator1` dataset.

MAP-Elites, sharing, novelty, clearing. It is also possible to include an adaptive prompt mechanism, which consists in asking the LLM to change the current task prompt based on the previous output (although it was not effective in the SR task).

A.2 PROMPTS

The initial prompt $\mathcal{P}_{\text{init}}$ is composed of the role specification, the task description, the scientific context and the format of the output. The specification of the role given to the LLM is contained within the task prompt, and explicitly tells: "You are an expert specializing in Symbolic Regression". We decided to specify the role since it could be useful to better frame the SR task to the LLM. In fact, recent studies on incorporating a specific role into LLM prompts have shown improved performances (Wang et al., 2024), also on reasoning tasks (Kong et al., 2024). The whole role+task prompt used is shown in Fig.5. This prompt is the same for all the experiments, except for the number of features that changes with the datasets used. An example of the scientific context is given in Fig. 6. In our experiments, the contextual information is directly extracted from the metadata of the datasets used, namely LSR-Synth datasets (Shojaee et al., 2025b) and `oscillator1` and `oscillator2` datasets (Shojaee et al., 2025a). Since LLM-SR is evaluated using these scientific priors, we decided to include them to ensure a fair comparison. Finally, Fig.7 shows the format prompt, in which we specify the code structure of the solution and the general format of the response.

Once the initialization is done, the subsequent generations require an enriched prompt $\mathcal{P}_{\text{loop}}$ in which we add the individual to evolve and one mutation prompt, which is randomly selected from a pre-fixed set, that can be customized as desired. In our settings, we have two possible mutation prompts that ask the LLM to refine the current expression or to propose a new one. Figure 8 shows an example of the LLM's response to the refinement option over a selected expression. The LLM gives the description of the mutated expression, explaining what has been changed. As discussed in the main paper, a substructure guidance mechanism is activated with a pre-defined rate. When enabled, the mutation prompt is enriched with a specific prompt that tells the LLM to include a substructure (selected from the archive of substructures) in addition to the mutation operation. Figure 9 shows an example of LLM response to the mutation + substructure-guidance prompt.

Format prompt

````
Format example:
   ```python
   import numpy as np

   def discovered_equation(Z, c):
      return HERE THE EQUATION
   ```


Give the response in the format:
# Description: <short-description of the meaning of the analytical
relationships and why you gave this expression. Also include the term(s)
you have added, deleted and/or changed.>
# Constants: <number of constants needed, e.g., 3>
# Code:
   ```python
      <code>
   ```
````

Figure 7: The format prompt.

Prompt

```
The symbolic structure of the formula (to be evolved) is:
c[0] * Z[:, 0]

Refine the current expression while preserving useful terms.
```

LLM response

````
# Description: Refine the current expression by adding a linear function
of velocity to the product of a linear function of position and a
constant, capturing the essence of a damped nonlinear oscillator system
with driving force.
# I have added a new term that uses a learnable constant to the product of
a linear function of position and a constant.
# Constants: 2
# Code:
```python
def discovered_equation(Z, c):
    return c[0] * Z[:, 0] + c[1] * Z[:, 1]
```
````

Figure 8: An example of the LLM response to the refinement option for one iteration of a run over the `oscillator1` dataset.

Mutation prompt + Substructure guidance

```
Refine the current expression while preserving useful terms.
### SUBSTRUCTURE-GUIDED MUTATION ###
You must use the following archived substructure in the mutation: Z[:, 0]**3
(ONLY IF it is not already present in the expression to modify).
The returned full equation must still be a valid single-line numpy expression
inside 'discovered_equation(Z, c)'.
```

LLM response

```
# Description: The refined expression preserves the driving force term,
the damping term, and the nonlinear stiffness term. I've replaced the
exponential term with a power term for position, added a new term with a
constant coefficient and a sinusoidal function of the velocity, and modified
the coefficients of the existing terms.
# Constants: 3
# Code:
```python
import numpy as np
def discovered_equation(Z, c):
    return c[0] * Z[:, 0] + c[1] * np.sin(Z[:, 1]) + c[2] * Z[:, 0]**3
```
```

Figure 9: An example of the LLM response to the mutation prompt and substructure guidance for one iteration of a run over the `oscillator1` dataset.

A.3 Detailed Definition of Symbolic Accuracy Metrics

As discussed in Section 3.3, the evaluation of SA between the best expression found $\hat{f}^\star$ and the ground truth expression $f^\star$ is made with two metrics. The SA_{AST} metric measures structural similarity through the overlap between the AST subtrees of the two compared expressions, while the SA_{Term} metric evaluates structural similarity at the level of additive terms rather than complete AST subtrees. The second metric tends to penalize less the additive terms present in $\hat{f}^\star$ but not in $f^\star$, giving a more generous comparison.

To justify this claim, we show below an example taking the best expression obtained for the `oscillator1` problem and the values of SA_{AST} and SA_{Term}. Consider the discovered equation

$$\hat{f}^\star(x, v) = -0.1952 \cdot x \exp(0.9942v) + 0.1408 \cdot x^3 + 0.8492 \cdot vx - 1.1545 \cdot x \sin(v) - 0.5012 \cdot v^3$$

and the corresponding ground-truth expression

$$f^\star(x, v) = F \sin(\omega x) - \alpha v^3 - \beta x^3 - \gamma xv - x \cos(x).$$

After the normalization employed by the term-based metric, their respective term sets become

$$\mathcal{T}_{Term}(\hat{f}^\star) = [v^3, xv, x^3, x \exp(v), x \sin(v)]$$

and

$$\mathcal{T}_{Term}(f^\star) = [\sin(x), v^3, xv, x^3, x \cos(x)]$$

The two expressions therefore share three normalized terms,

$$\mathcal{T}_{Term}(\hat{f}^\star) \cap \mathcal{T}_{Term}(f^\star) = [v^3, xv, x^3]$$

At the same time, the discovered expression contains two additional terms, $x \exp(v)$ and $x \sin(v)$, while two ground-truth terms, $\sin(x)$ and $x \cos(x)$, are missing. The union consequently contains

seven distinct terms, yielding $SA_{Term} = \frac{3}{7} \approx 0.429$. For the same pair of expressions, the AST-based metric gives $SA_{AST} \approx 0.159$.

The difference between the two SA values depends on the way in which they are computed. Indeed, for SA_{Term} the three correctly recovered terms v^3, x^3 and xv are directly recognized as matches, while each additional or missing term contributes once to the Jaccard union. On the other hand, for SA_{AST}, non-matching building blocks introduce multiple distinct subtree signatures, thereby producing a stronger reduction in structural similarity.

These considerations are not intended to establish a preferred metric, but rather to highlight that different ways of calculating the symbolic similarity between expressions can lead to a reevaluation of the results obtained. While the first metric is very stringent, low values still indicate that the resulting expression contains noise that distances it from the ground truth. On the other hand, the second metric allows us to highlight common terms, and high values indicate that the approach has successfully recovered substructures present in the ground truth.

A.4 Random function datasets

In the following, we briefly describe how we generated the datasets of randomly generated functions. The symbolic expressions were constructed by combining polynomial, trigonometric, and interaction terms. Each term was assigned a coefficient with a randomly chosen sign and a magnitude sampled uniformly between 0.35 and 3.0. Candidate expressions were filtered according to predefined complexity ranges and numerical validity criteria, ensuring that all input variables contributed to the target. Each training and ID set has 1'000 samples, using Latin hypercube sampling over $[-1, 1]^d$, where d denotes the number of input variables, while the OOD set has 2'000 samples, each with exactly one input dimension sampled outside the ID range, i.e., from $[-2, -1.2] \cup [1.2, 2]$, while the remaining dimensions stayed within $[-1, 1]$. The extrapolated coordinate was balanced across input dimensions and randomly assigned to samples. Targets were computed directly from the generating expressions without additive noise.

B Symbolic Regression methods

We compared EvoMO-SR against SR baselines, including two LLM-based methods.

uDSR uDSR (Landajuela et al., 2022) is a modular hybrid framework for SR that extends DSR (Petersen et al., 2019), unifying different strategies, including the integration of additional linear token and GP search in the decoding stage. In our comparisons, we set the batch size to $1'000$ and the maximum iterations to $100'000$.

PySR PySR (Cranmer, 2023) is an open-source symbolic regression framework designed for SR, based on a multi-population evolutionary algorithm. In our comparisons, we set the population size to 20 and the maximum number of evaluations to $1'000$.

Operon Operon (Burlacu et al., 2020) is a high-performance framework for SR based on C++, which enhances GP with an efficient linear tree representation, low-memory evaluation, and the generation of offspring with a concurrency model. For our experiments, we set the population size to 50, the maximum number of evaluations to $1'000$, and the optimizer iterations to 100.

GPGomea GPGomea (Virgolin et al., 2021) is a GP-based approach for SR that learns linkage information, i.e., dependencies between genotype positions, and uses Gene-pool Optimal Mixing to propagate useful expression patterns. For our experiments, we set the maximum number of evaluations to $1'000$.

LLM-SR LLM-SR (Shojaee et al., 2025a) performs SR using an LLM that evolves and iteratively refines expression skeletons, leaving the fitting of coefficients to off-the-shelf optimizers. To make a fair comparison with our framework, we set the number of generated functions to $1'000$ and optimized constants with L-BFGS-B.

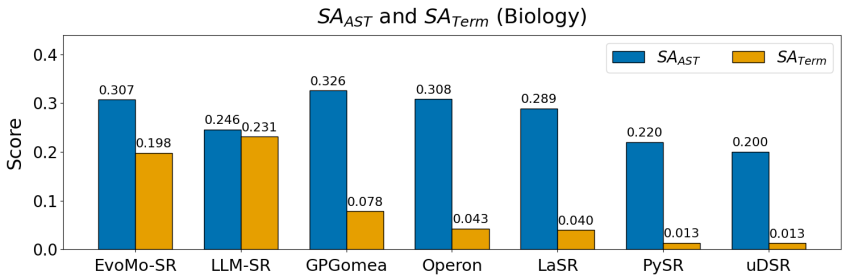


Figure 10: Comparison of SA_{AST} and SA_{Term} between EvoMO-SR and the other SR baselines over LSR-Synth Biology dataset.

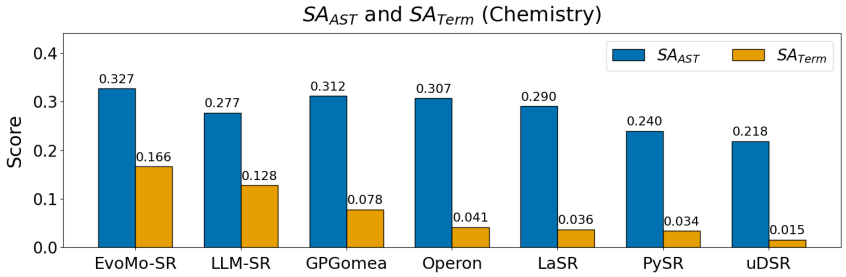


Figure 11: Comparison of SA_{AST} and SA_{Term} between EvoMO-SR and the other SR baselines over LSR-Synth Chemistry dataset.

LaSR LaSR (Grayeli et al., 2025) does not use the LLM as the primary equation generator, but for the enhancement of the GP search performed via PySR. In our experiments, we set both the number of iterations and the number of cycles per iteration to 10, maintaining the population size to 33 with 15 total populations.

C Additional Results

C.1 Symbolic Accuracy

In Section 4.1 are shown the results of SA_{AST} and SA_{Term} of EvoMO-SR, LLM-SR, LaSR and the other GP-based SR baselines over the LSR-Synth Physics dataset. For the sake of completeness, we report here the results over the other three dataset: Biology (see Fig. 10), Chemistry (see Fig. 11), and Material Science (see Fig. 12). EvoMO-SR always obtains better results on SA_{AST} compared to the two main competitors, i.e., LLM-SR and LaSR, except for the Physics domain (see Figure 3) where LaSR is slightly superior. However, it is not always the best compared to GP-based baselines. Indeed, for the Biology and Physics dataset, GPGomea reached $SA_{AST} = 0.326$ and $SA_{AST} = 0.277$ respectively, which are the highest value obtained across all the methods. This is not surprising as GPGomea has already showed its goodness in symbolic recovery (de Franca et al., 2024). Concerning SA_{Term}, EvoMO-SR wins in the Chemistry and Physics datasets, with LLM-SR being the hardest competitor.

C.2 Examples of expressions obtained

Below are some of the best results from the experiments using LLM-driven methods.

C.2.1 `oscillator1` dataset

The fitted ground truth for the `oscillator1` dataset is the following expression:

$$\dot{v} = 0.8 \sin x - 0.5v^3 - 0.2x^3 - 0.5xv - x \cos x.$$

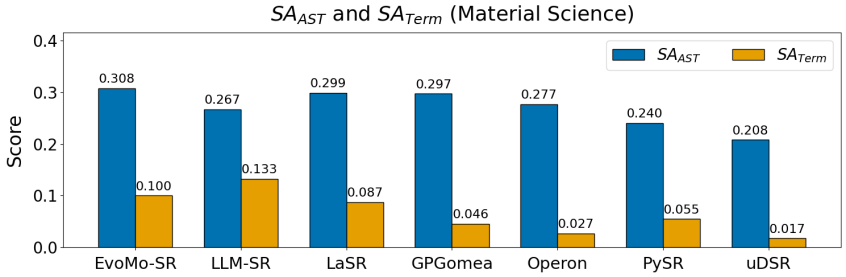


Figure 12: Comparison of SA_{AST} and SA_{Term} between EvoMO-SR and the other SR baselines over LSR-Synth Material Science dataset.

where v is the velocity and x the position.

The best expression obtained for `oscillator1` for LLM-SR achieved NMSE ID and OOD equal to 1.87e-11 and 2.46e-10, respectively, and is:

$$\dot{v} = \text{gradient}(v, 0.002)$$

Basically, LLM-SR retrieved a formula which numerically differentiates the velocity vector through the `gradient` operator provided by NumPy. Since acceleration is itself the time derivative of velocity and the observations are sampled at a fixed temporal resolution, this expression provides an extremely accurate approximation of the target, explaining why NMSE ID and OOD are so low. However, for the `oscillator1` datasets, this shortcut is taken only once out of 20 runs. For example, the second best expression among the 20 runs based on accuracy is:

$$\dot{v} = -0.4648x + 0.0008v - 0.00022\sin(2\pi \cdot 1.4157x) +$$
$$+0.1495x^3 - 0.4998xv + 0.2688v^2 - 0.5228v^3 + 0.2664(x - v^2)$$

which obtains NMSE ID and OOD of 6.06e-7 and 2.33e-2, respectively. This expression was also the one with highest SA_{Term}. Here, the symbolic expression is much closer to the ground truth, sharing some additive terms such as x^3 and v^3.

The best expression among the 20 runs based on accuracy for EvoMO-SR was

$$\dot{v} = 0.0889\cos(2.2528x + 1.5709) - 0.4999xv - 0.4997v^3$$

and gives NMSE ID and OOD of 6.35e-8 and 1.78e-4, respectively.

The one with the highest SA_{Term} (equal to 0.57) is:

$$\dot{v} = -0.0147 - 0.0463x^3 + 0.0004\exp(-x) + 1.0216x - 0.4994v^3 +$$
$$-0.4999xv + 0.4882(-2.5\sin x + 0.0293)$$

and gives NMSE ID and OOD of 8.66e-7 and 1.54e-2. respectively.

C.2.2 OSCILLATOR2 DATASET

The ground truth for the `oscillator2` dataset is:

$$\dot{v} = 0.3\sin t - 0.5v^3 - xv - 5.0x \cdot \exp(0.5x)$$

For the `oscillator2` dataset, LLM-SR returned expressions involving the `gradient` operator in 6 out of 20 runs, showing that the use of this operator might represent a recurring behavior. In the run that achieves the lowest NMSE ID and OOD (9.74e-10 and 1.38e-10, respectively), the expression is

$$\dot{v} = \text{gradient}(-1.48e^{-6}t - 2.57e^{-6}x + v + 0.99, 1)$$

while the second and third best expressions (always according to NMSE) are both equal to gradient(v, t). So, similarly to the case of the first dataset, also here the method tries to employ

Table 5: Comprehensive ablation study removing substructure guidance (Sub) and multi-objective (MO) selection mechanism over `oscillator1` dataset.

Method		Oscillator 1				
Sub	**MO**	**ID** ↓	**OOD** ↓	SA_{AST} ↑	SA_{Term} ↑	**Complexity** ↓
×	×	4.29e-6	2.30e-2	0.14	0.27	121.6
×	✓	1.39e-5	2.06e-2	**0.31**	0.37	**60.2**
✓	×	**3.81e-6**	**1.70e-2**	0.15	0.32	112.4
✓	✓	4.79e-6	2.11e-2	0.28	**0.41**	86.8

Table 6: Comprehensive ablation study removing substructure guidance (Sub) and multi-objective (MO) selection mechanism over `oscillator2` dataset.

Method		Oscillator 2				
Sub	**MO**	**ID** ↓	**OOD** ↓	SA_{AST} ↑	SA_{Term} ↑	**Complexity** ↓
×	×	**1.38e-6**	1.34e-2	0.14	0.27	171.0
×	✓	1.98e-5	3.83e-3	0.29	0.32	**94.0**
✓	×	5.03e-6	2.69e-3	0.11	0.16	214.8
✓	✓	8.45e-6	**1.60e-3**	**0.29**	**0.35**	105.6

a shortcut to approximate the acceleration. EvoMO-SR never proposes final expressions with non-conventional operators such as the `gradient`. The expression with the lowest NMSE ID and OOD (1.10e-8 and 2.58e-6, respectively) is also the one with the highest SA_{Term} (equal to 0.44), and is the following:

$$\dot{v} = 0.2999 \sin t - 2.1033x - 0.0001v - 0.8877x^2 + \\ -9.4404e^{-6} \exp v * \cos t - 1.6080x \cdot \exp x - 0.4984v^3 - 1.0003xv - 1.2894 \sin x$$

C.3 Additional ablation studies

We performed additional ablation studies by alternatively removing the main components of EvoMO-SR: substructure guidance and multi-objective selection. We carried out 3 different experiments: without substructure guidance and without multi-objective selection (i.e., selection is performed based on accuracy only), without substructure guidance and with multi-objective selection, with substructure guidance and without multi-objective selection. The results of these experiments (each configuration was run 5 times with Llama-3.1-8B-Instruct as the LLM backbone) are shown in Table 5 (for the `oscillator1` dataset) and Table 6 (for the `oscillator2` dataset), which compare NMSE ID and OOD, the two similarity metrics and the average complexity (measured counting the number of nodes in the final best expressions). As expected, the most consistent effect of multi-objective selection was a reduction in expression size for both dataset and both configuration with multi-objective selection enabled. This aspect is crucial because reduced complexity leads to greater interpretability of the resulting expressions, and this becomes even more important when this trade-off does not reduce overall accuracy. The effect of substructure guidance has to be analyzed carefully. For the `oscillator1` dataset, as already observed in Section 4.4, the use of substructure guidance slightly improves NMSE ID compared to the configuration without guidance and with multi-objective selection. The improvement in symbolic accuracy can be seen more for SA_{Term}, while the complexity increases. For the `oscillator2` dataset, substructure guidance in EvoMO-SR improved symbolic accuracy values and NMSE OOD against all the configurations. Overall, we can state that substructure guidance did not consistently improve the metrics when considered as an isolated main effect, suggesting an interaction with multi-objective selection rather than a uniform benefit.

C.3.1 GP-based mutation operator

Another ablation study concerned the use of LLM for the mutation of individuals. In EvoMO-SR, the LLM mutates the expressions with the prompt "Refine the current expression while preserving useful terms". There is no control over the result of this mutation: the LLM can deliberately delete one or more terms, add one or more terms, modify existing substructures etc. Since the LLM is integrated in an evolutionary loop, it is easy to replace the LLM-driven mutation with a standard

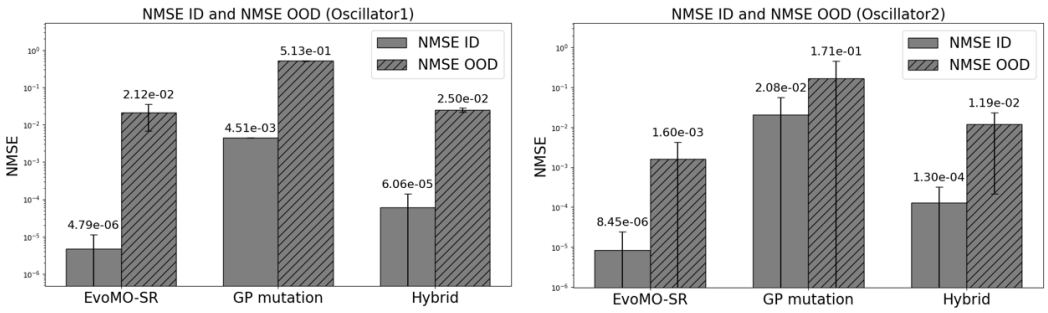


Figure 13: Comparison of NMSE ID and OOD values between EvoMO-SR with LLM-driven mutation (original), with GP mutation, and hybrid mutations (both LLM-driven and GP-based).

Table 7: Domain-specific pairwise Wilcoxon test with Holm's correction between EvoMO-SR and LLM-SR

Domain	Split	p-value	EvoMO-SR (ours) wins	LLM-SR wins	Ties
Physics	ID	0.885	**24**	20	0
	OOD	0.007	**30**	14	0
Chemistry	ID	0.001	**25**	11	0
	OOD	3.33e-5	**31**	5	0
Biology	ID	3.00e-5	**22**	2	0
	OOD	2.00e-5	**22**	2	0
Material Science	ID	8.00e-6	**24**	1	0
	OOD	5.00e-6	**24**	1	0

GP mutation operator (i.e., `mutUniform`, which randomly select a subtree in the expression tree with another randomly generated subtree) in order to analyze whether the mutation strategy has impact on the results. Figure 13 shows the average NMSE ID and OOD values obtained in 5 runs over the `oscillator1` and `oscillator2` datasets by replacing LLM mutation with traditional GP mutation and with an hybrid approach (i.e., LLM mutation alternated with GP mutation). The mutation prompt concerning the proposal of a new expression was disabled for this experiment. This ablation study clearly shows better accuracy performance in favour of EvoMO-SR (with LLM mutation).

C.4 Statistical tests

In Section 4.1 we showed the comparison of NMSE ID and OOD values between EvoMO-SR and the other baselines across LSR-Synth datasets. The results highlight the superiority of our method in almost all domains, except for Physics, where LLM-SR outperforms it. To statistically demonstrate this superiority, we performed a Wilcoxon test with Holm correction between EvoMO-SR and its main competitor, LLM-SR, using each benchmark problem as the paired experimental unit. For each problem, NMSE was first averaged across the five independent runs. We considered ID and OOD performance separately and applied Holm correction to account for multiple comparisons. Table 7 reports the resulting Holm-adjusted p-values together with the number of problems on which each method achieves lower NMSE.

These results confirm that the differences generally favor EvoMO-SR. For Chemistry, Biology, and Material Science, EvoMO-SR achieves significantly lower NMSE than LLM-SR for both ID and OOD evaluation after Holm's correction ($p < 0.05$). For the Physics dataset, no statistically significant difference is observed for ID performance ($p = 0.885$). However, EvoMO-SR significantly outperforms LLM-SR on Physics OOD data winning on 30 out of 44 problems. Overall, EvoMO-SR consistently reaches lower NMSE ID and OOD with statistical significance, obtaining 95 wins out of 129 problems for ID performance and 107 wins out of 129 problems for OOD performance.

Table 8: Pairwise Wilcoxon test with Holm's correction between EvoMO-SR and LLM-SR over `oscillator1` and `oscillator2` datasets

Domain	Split	p-value	EvoMO-SR (ours) wins	LLM-SR wins	Ties
`oscillator1`	ID	0.09	**14**	6	0
	OOD	0.24	**12**	8	0
`oscillator2`	ID	0.09	**13**	7	0
	OOD	0.09	**13**	7	0

Table 9: Pairwise Wilcoxon test with Holm's correction over SA metrics between EvoMO-SR and LLM-SR over `oscillator1` and `oscillator2` datasets.

Domain	Metric	p-value	EvoMO-SR (ours) wins	LLM-SR wins	Ties
`oscillator1`	SA_{AST}	0.01	5	**15**	0
	SA_{Term}	0.13	**13**	6	1
`oscillator2`	SA_{AST}	0.001	3	**17**	0
	SA_{Term}	0.001	**17**	1	2

The same statistical test was carried out for the results obtained over `oscillator1` and `oscillator2` datasets. Table 8 shows results obtained performing the Wilcoxon paired test over NMSE ID and OOD values. Even though EvoMO-SR always achieves more wins against LLM-SR, the p-value scores are above 0.05, meaning that the test does not show any statistical significance.

The two methods were also compared on the SA values, as shown in Table 9. Here, almost all the results are statistically significant, obtaining a p-value score below 0.05, except for SA_{Term} in the `oscillator1` datasets.

D PARETO FRONTS ANALYSIS

Figure 14 shows 4 distinct final approximated Pareto fronts each obtained from a run of an individual problem. Each Pareto front is computed with the Non-Dominated Sorting algorithm balancing accuracy (i.e., NMSE) and complexity (count of math nodes). From the final Pareto front is possible to extract the best expression according to a predefined criteria, which can be the knee point, the one with the lowest complexity and the one with the lowest NMSE (highest accuracy). However, the choice is not always straightforward. It is clear that the geometry of the Pareto front changes with the problem. Only for the MatSci13 problem the knee point seems the best alternative, as an increase in complexity does not improve the accuracy significantly. For all the other three cases, an increase in complexity beyond the knee always brings an increase in accuracy by several orders of magnitude.

The results reported in this paper are obtained by selecting the best expression based on minimum training NMSE, as the main preference was to favour more accurate expressions even if more complex. To justify this choice, we studied the NMSE ID and OOD, the complexity and the SA values comparing best expressions selected from the final Pareto front according to lowest training NMSE, knee point and lowest complexity. Figure 15 shows that, as expected, when choosing expressions from the Pareto front based on training NMSE, both ID and OOD values are consistently the lowest. Similarly, when choosing the expression in the Pareto front having the lowest complexity, the average complexity values across problems and runs are the lowest (see Figure 16). A less obvious comparison is shown in Figure 17, where the three criteria are compared based on SA_{AST} and SA_{Term}. The similarity values remain low across all the three criteria, and no winner emerges.

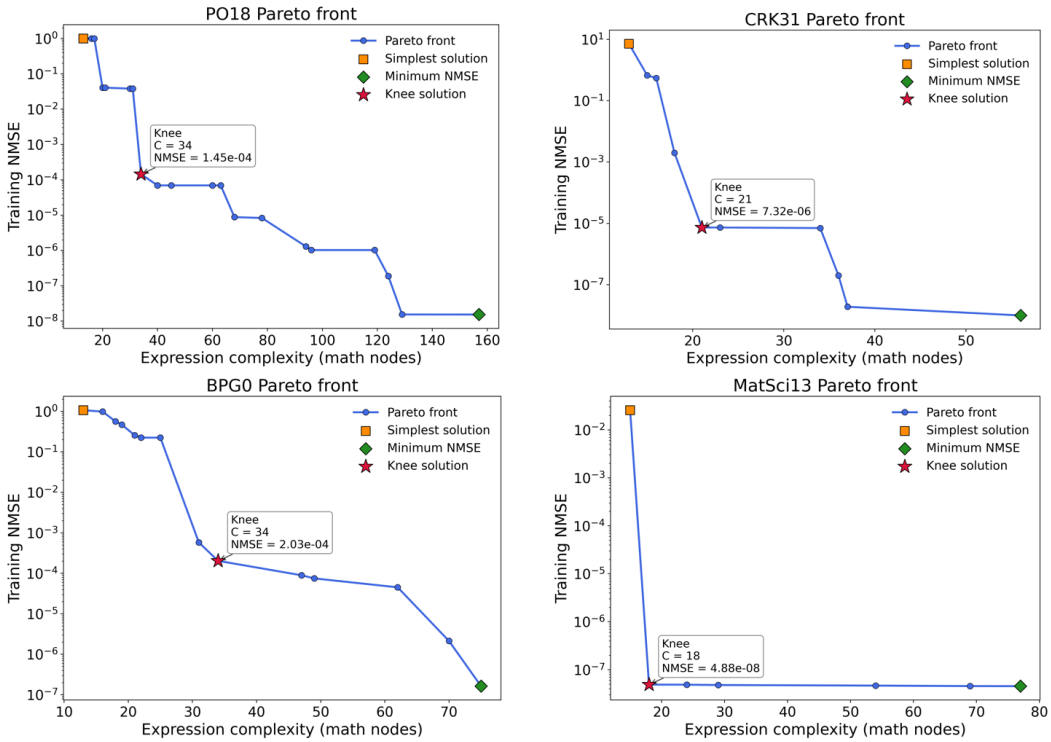


Figure 14: Obtained approximated Pareto fronts of distinct runs over 4 problems: PO18 for the Physics domain, CRK31 for the Chemistry domain, BPG0 for the Biology domain, and MatSci13 for the Material Science domain.

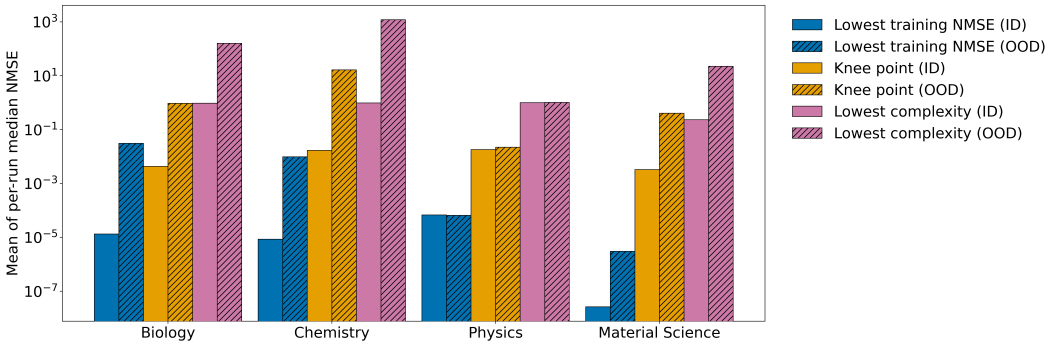


Figure 15: Average per-run Median NMSE ID and OOD values of the best expressions selected based on the lowest training NMSE, the lowest complexity and the knee point of the final pareto front.

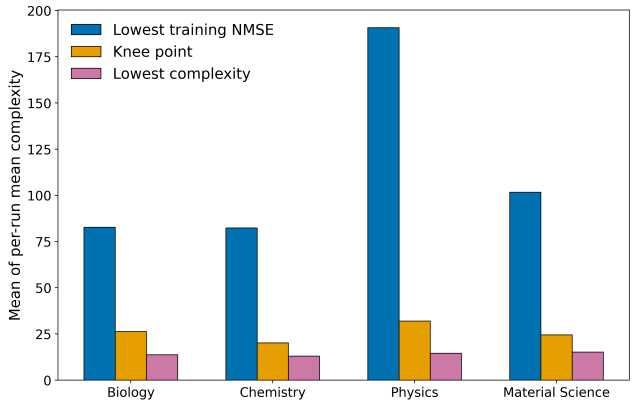


Figure 16: Average per-run Mean complexity values of the best expressions selected based on the lowest training NMSE, the lowest complexity and the knee point of the final pareto front.

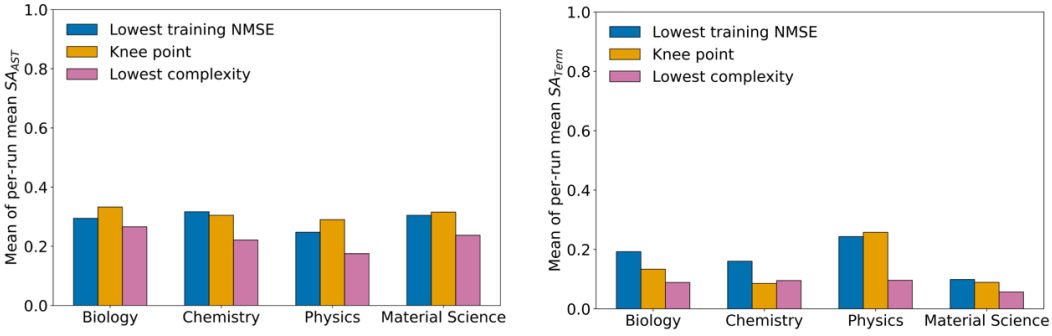


Figure 17: Average per-run Mean similarity values of the best expressions selected based on the lowest training NMSE, the lowest complexity and the knee point of the final pareto front. On the left results concerning SA_{AST} values, while on the right results concerning SA_{Term} values.

Table 10: High-level comparison between LLM-SR and our framework. Both approaches employ iterative LLM-based equation generation guided by previous evaluations, but they differ in how evolutionary information is represented and used. †LLM-SR performs fitness-biased selection of programs from its experience buffer for in-context prompting; here, "explicit parent selection" refers specifically to selecting individuals as parents in a population-based evolutionary algorithm.

Component	**LLM-SR**	**Our method**
LLM-based structural generation	✓	✓
External coefficient optimization	✓	✓
Data-driven iterative feedback	✓	✓
Multiple maintained candidates	✓	✓
Island-based experience management	✓	–
Fitness-biased sampling of previous solutions	✓	✓
Explicit parent selection	–†	✓
Explicit parent–offspring survival selection	–	✓
Configurable elitist survival	–	✓
Multi-objective survival selection	–	✓
Separate archive of reusable substructures	–	✓
Substructure-guided LLM mutation	–	✓

E QUALITATIVE COMPARISON AGAINST LLM-BASED COMPETITORS

Our approach shares with LLM-SR (Shojaee et al., 2025a) the general idea of using an LLM to explore the space of symbolic expressions within a loop while relying on data-driven evaluation to guide subsequent generations. Both methods also decouple structural search from coefficient optimization: the LLM proposes equation structures, while numerical coefficients are fitted using an external optimizer. Nevertheless, the two approaches operationalize the search process differently.

LLM-SR maintains a dynamic experience buffer organized according to a multi-island strategy. Previously evaluated equation programs are stored and sampled to construct in-context demonstrations for subsequent LLM queries. Sampling is biased toward high-performing clusters and, within a selected cluster, toward shorter programs. In addition, poorly performing islands are periodically reset. Therefore, LLM-SR incorporates several evolutionary-search principles, including fitness-based sampling, diversity preservation, and iterative refinement, although these mechanisms primarily regulate which previously discovered complete programs are provided back to the LLM as experience.

Our framework instead embeds the LLM within an explicit population-based evolutionary loop. At each generation, individuals are selected from the current population and used as parents for LLM-driven variation. The resulting offspring are evaluated and subsequently compete for survival according to the multi-objective selection strategy. A further distinction of EvoMO-SR is that evolutionary information is maintained not only at the level of complete candidate expressions. During evolution, structural differences between a parent and its offspring are analyzed to identify newly introduced symbolic substructures. Promising substructures are stored in a separate archive and can subsequently be included as guidance when generating mutations of other candidate expressions. Consequently, the population and the substructure archive play complementary roles: the former maintains complete candidate solutions and drives evolutionary selection, whereas the latter provides a lightweight memory of reusable symbolic components discovered during the search. Furthermore, EvoMO-SR provides a set of evolutionary-based parameters to customize the search, such as parents and offspring counts, survival selection, elitism, parent selection strategies (e.g., random, tournament selection, roulette), niching mechanisms (e.g., sharing, clearing, novelty, map-elites). Table 10 summarizes the main methodological differences. These distinctions should not be interpreted as implying that LLM-SR is not evolutionary; rather, the two methods use evolutionary principles at different levels of the search process. LLM-SR primarily exploits fitness-guided management and reuse of complete equation programs, while our approach combines LLM-based variation with explicit population-level selection and an additional mechanism for extracting and reusing symbolic building blocks.

Our method is also related to LaSR (Grayeli et al., 2025), which combines LLM guidance with an explicit evolutionary SR algorithm. In contrast to LLM-SR, LaSR directly builds upon the population-based search implemented by PySR, maintaining multiple populations and combining conventional

Table 11: High-level comparison between LaSR and our framework. Both approaches combine explicit evolutionary symbolic regression with LLM-based guidance, but differ substantially in how knowledge acquired during evolution is represented and reused. LaSR abstracts high- and low-performing expressions into natural-language concepts, whereas our method explicitly extracts and archives symbolic substructures introduced through parent–offspring transformations.

Component	**LaSR**	**Our method**
Explicit population-based evolutionary search	✓	✓
Data-driven iterative feedback	✓	✓
LLM-guided evolutionary mutation	✓	✓
Standard symbolic GP mutation/crossover	✓	–
LLM-based mutation	✓	✓
LLM-based crossover	✓	–
Reuse of information discovered during evolution	✓	✓
Guidance representation	Textual concepts	Symbolic substructures
Data-driven quality associated with archived guidance	Indirect	✓
Explicit multi-objective survival selection	–	✓

symbolic mutation and crossover with LLM-guided counterparts. The LLM-based operators are applied probabilistically and are conditioned on a dynamically learned library of concepts, allowing semantic information extracted during the search to influence subsequent evolutionary operations. The central difference with our approach concerns the representation and acquisition of the knowledge used to guide evolution. LaSR constructs a library of natural-language concepts by prompting an LLM to abstract patterns from high- and low-performing expressions. These concepts are subsequently evolved by the LLM and sampled to condition hypothesis initialization, mutation, and crossover. Our framework instead operates directly at the symbolic level: structural changes introduced between a parent and its offspring are identified and candidate mathematical substructures are extracted from these changes. Promising substructures are stored in a dedicated archive and can later be explicitly provided to the LLM when generating new offspring. Therefore, while both methods exploit information accumulated during evolution to bias subsequent search, LaSR transfers this information through LLM-generated semantic abstractions, whereas our method transfers explicit symbolic building blocks discovered during parent–offspring evolutions. The two mechanisms also differ in how the guidance information is maintained. In LaSR, individual textual concepts are not assigned an explicit data-driven fitness; concept evolution is primarily intended to generate potentially useful ideas that increase exploration. In our framework, archived substructures remain associated with the predictive performance of the offspring in which they were discovered, and the archive has a bounded capacity in which candidate structures compete according to this performance. Finally, both approaches account for predictive accuracy and expression simplicity, although they employ these criteria differently. LaSR extracts a Pareto frontier over dataset loss and syntactic simplicity, which is subsequently used during concept abstraction and for selecting high-quality expressions. In the multi-objective variant of our framework, NMSE and symbolic complexity instead constitute the objectives of the evolutionary process itself, with non-dominated sorting and crowding-based environmental selection directly determining population survival. Table 11 summarizes the main similarities and differences between the two approaches.

DrSR (Wang et al., 2025) extends LLM-SR with data-aware insights and the summarization and reuse of ideas. However, in contrast to EvoMO-SR, DrSR acquires information through LLM-based data interpretation and reflection. Moreover, similarly to LLM-SR, this method retains promising equations in an experience buffer, but does not perform accuracy–complexity Pareto survival. LLM-Meta-SR Zhang et al. (2026) combines LLM-driven evolution with multi-objective survival selection, but with different targets. While EvoMO-SR evolves symbolic equations, LLM-Meta-SR generates selection operators that are evaluated through inner SR runs across multiple meta-training datasets. The multi-objective mechanism is used to balance downstream predictive performance and operator code length.

Table 12: Mean per-run computation time (in seconds), number of API requests, number of input tokens and number of output tokens over the `oscillator1` dataset. Values are reported as mean $\pm$ standard deviation over three runs.

	Runtime (s)	**# API requests**	**# Input tokens**	**# Output tokens**
LLM-SR	$3'279.90 \pm 82.55$	250.00 ± 0.00	$767'525.33 \pm 31'712.75$	$506'200.33 \pm 1'565.21$
LaSR	$\mathbf{490.79 \pm 9.25}$	230.67 ± 15.82	$59'708.67 \pm 3'734.57$	$20'654.00 \pm 432.85$
EvoMO-SR	594.77 ± 63.53	$1'000.00 \pm 0.00$	$783'589.67 \pm 25'376.85$	$150'259.00 \pm 7'743.26$

E.1 Comparison over time, calls and tokens

We measured computation time (i.e., per-run wall-clock time), the number of API requests, and the number of input/output tokens by executing EvoMO-SR, LLM-SR and LaSR over the `oscillator1`. Each method was evaluated in three runs with distinct random seeds. Table 12 reports the mean and standard deviation over the three runs. The lowest runtime was obtained by LaSR, while LLM-SR was approximately 5.5x slower than EvoMO-SR. Given that the main evolutionary loop of LaSR is managed by pySR and the GP operators are occasionally alternated with LLM-driven ones, the number of API requests is small compared to EvoMO-SR. For LLM-SR, the number of API requests per run is equal to 250 because at each iteration the LLM is prompted to generate 4 distinct equations (thus requiring 1'000 model generations). LLM-SR also generated considerably longer responses, producing approximately 3.4 times as many output tokens as EvoMO-SR. For LaSR, the reduced overall costs are accompanied by lower predictive performance. Concerning LLM-SR, even though its numerical performance is similar to EvoMO-SR, its computation time is substantially longer.