# Graph-based Symbolic Regression with Invariance and Constraint Encoding

**Ziyu Xiang**[1] **Kenna Ashen**[1] **Xiaofeng Qian**[1] **Xiaoning Qian**[1,2]
[1]Texas A&M University, College Station, TX 77840, USA
[2]Brookhaven National Laboratory, Upton, NY 11973, USA
{zxiang,kashen13,feng,xqian}@tamu.edu; xqian1@bnl.gov

## Abstract

Symbolic regression (SR) seeks interpretable analytical expressions that uncover the governing relationships within data, providing mechanistic insight beyond 'black-box' models. However, existing SR methods often suffer from two key limitations: (1) *redundant representations* that fail to capture mathematical equivalences and higher-order operand relations, breaking permutation invariance and hindering efficient learning; and (2) *sparse rewards* caused by incomplete incorporation of constraints that can only be evaluated on full expressions, such as constant fitting or physical-law verification. To address these challenges, we propose a unified framework, **Graph-based Symbolic Regression (GSR)**, which compresses the search space through the permutation-invariant representations, expression graphs (EGs), that intrinsically encode expression equivalences via a term-rewriting system (TRS) and a directed acyclic graph (DAG) structure. GSR mitigates reward sparsity by employing a hybrid neural-guided Monte Carlo tree search (hnMCTS) on EGs, where constraint-informed neural guidance enables the direct incorporation of expression-level constraint priors, and an adaptive $\epsilon$-UCB policy balances exploration and exploitation. Theoretical analyses establish the uniqueness of our proposed EG representation and the convergence of the hnMCTS algorithm. Experiments on synthetic and real-world scientific datasets demonstrate the efficiency and accuracy of GSR in discovering underlying expressions and adhering to physical laws, offering practical solutions for scientific discovery.

## 1 Introduction

Symbolic regression (SR) [1, 2] is an approach to unveil the inherent dependencies governing the system under study in a symbolic form. SR explores the space of closed-form mathematical expressions to find the most ideal analytical expressions that capture the underlying variable relationships from observed data. Compared to black-box models (e.g., neural networks) that lack transparency or direct mechanistic understanding, SR strikes a balance between prediction accuracy and interpretability, highlighting its potential in advancing AI for scientific discovery [3].

However, solving SR problems is inherently NP-hard [4], due to the combinatorial nature of possible mathematical expressions [5]. To improve search efficiency, recent SR methods have leveraged modern supervised deep learning (DL) and reinforcement learning (RL) techniques [1, 6, 7, 8]. Despite these advancements, little attention has been given to capturing the inherent invariances of expressions and effectively incorporating constraints in SR, leading to suboptimal learning strategies.

Prevailing works in SR [9, 8, 10] adopt semantic or expression tree (ET) representations, which fail to account for the equivalence of mathematically identical expressions, breaking permutation invariance with redundant states that hinder fast convergence. Additionally, the limited structural information in ET may not fully leverage the effectiveness of DL-based encoding, limiting its expressive power.

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

Furthermore, existing search-based methods [11, 12] primarily enforce constraints evaluated at the operation level by incorporating priors during the sampling stage to avoid invalid expressions. However, constraints that depend on complete expressions, such as those involving constant fitting and physical law violation checking, are typically used as *post hoc* reward penalties instead of direct prior constraint incorporation in search. Such current practice leads to a sparse search space, making SR less effective for complex, constrained scientific research applications.

To address these challenges, we propose a Graph-based Symbolic Regression(GSR), which encodes permutation invariance and constraint priors using a Relational Graph Convolutional Network (R-GCN) on expression graphs (EGs). EGs inherently capture equivalences of expressions via a term rewriting system (TRS) and its directed acyclic graph (DAG) structure, reducing redundant representations. To fully exploit EGs, we introduce a hybrid neural-guided Monte Carlo tree search (hnMCTS), which balances exploration and exploitation by combining vanilla MCTS with neural-guided MCTS through an $\epsilon$-UCB policy, where R-GCN leverages node and edge features of EGs to encode expression-level constraint priors, allowing them to be incorporated directly during the sampling process for the optimal search efficiency. Evaluating on widely recognized SR benchmark datasets and real-world applications demonstrates that GSR effectively compresses the search space, mitigates reward sparsity challenges, and reduces the reliance on costly constant fitting through its structured encoding. These results highlight the efficiency, accuracy, and robustness of GSR in solving complex, physics-informed SR problems for scientific discovery.

## 2 Preliminary and Related Work

### 2.1 Symbolic Regression (SR)

Given a dataset $\mathcal{D} = \{(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), \ldots, (\boldsymbol{x}_N, y_N)\}$, SR aims to find a mathematical expression $f$ to map the input variable vector $\boldsymbol{x}_i$ to the corresponding output $y_i$ with the minimum error $L(\cdot, \cdot)$ over all data points in $\mathcal{D}$:

$$\min_{f} \sum_{i=1}^{N} L(f(\boldsymbol{x}_i), y_i), \tag{1}$$

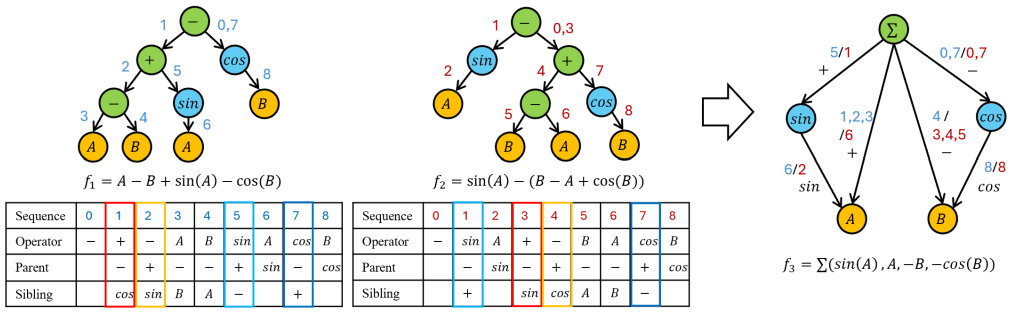
where $f$ belongs to the mathematical expression space constructed using a predefined grammar (e.g., arithmetic operations, functions, constants, and variables) as a dictionary set $\Phi = \{\phi_0, \phi_1, \ldots, \phi_n\}$. In our method, $\phi$ denotes the operations or variables $\boldsymbol{x}_i$. We also include operator $const$ for inserting constants and $trans$ for a transplantation strategy in $\Phi$ by default, detailed in Appendix C.1.

In this work, we develop GSR based on hnMCTS by leveraging vanilla MCTS [8, 13] for lightweight exploration and neural-guided MCTS [14] for enhanced exploitation. The proposed framework not only optimizes the exploration-exploitation trade-off but also seamlessly integrates with graph-based expression representation, enabling more structured and efficient search.

### 2.2 Expression Representation

SR can be formulated as a Markov Decision Process (MDP), where a new operator or variable is iteratively sampled based on the current expression state. The expression tree (ET) is the most commonly used state representation [2], as shown in Fig. 1. In ET, operators are modeled as nodes, and edges represent hierarchical relationships between operators and operands, with trees growing from outer functions to inner ones. This structure reflects the sequential construction of expressions through iterative sampling steps.

However, ET representation suffers from a major limitation: it fails to account for expression equivalences and permutation invariance in operator generation sequences. As shown in Fig. 1, two mathematically equivalent expressions with the same set of nodes can have different generation sequences, resulting in distinct parent and sibling relationships and tree structures. This redundancy inflates the state space, reducing learning efficiency, particularly for RL-based approaches (e.g., PG and MCTS) and sequential encoding methods (e.g., RNNs), which struggle to converge due to unnecessary state diversity. Furthermore, ET encodes only basic operator types and hierarchical relationships, lacking higher-order features such as arity and operation order, making it hard to distinguish argument positions(e.g., base or exponent for power). These limitations weaken the predictive power of DL-based encoders.




| Sequence | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|---|
| Operator | − | + | − | $A$ | $B$ | $sin$ | $A$ | $cos$ | $B$ |
| Parent | | − | + | − | − | + | $sin$ | − | $cos$ |
| Sibling | | $cos$ | $sin$ | $B$ | $A$ | − | | + | |

| Sequence | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|---|
| Operator | − | $sin$ | $A$ | + | − | $B$ | $A$ | $cos$ | $B$ |
| Parent | | − | $sin$ | − | + | − | − | + | $cos$ |
| Sibling | | + | | $sin$ | $cos$ | $A$ | $B$ | − | |

Figure 1: An illustrative example of the permutation invariance problem in SR. (I) $f_1$ and $f_2$: two equivalent expressions represented by ETs generated in different sequences as in the tables, yielding two distinct representations due to the different parent and sibling relationships highlighted by the colored boxes. This breaks the permutation invariance for sequential encoding methods. (II) $f_3$: another equivalent expression represented in our proposed expression graph (EG), generated by both $f_1$ sequence (blue numbers) and $f_2$ sequence (red numbers), yielding the same representation, preserving the permutation invariance for the Graph Neural Network (GNN) encoder.

Most existing SR works [7, 8, 15, 16] rely on either semantic or ET-based representations, with little focus on improving expression representation. The authors of AI-Feynman [11, 17] addressed this by pre-training neural networks to capture modularities and symmetries, decomposing SR problems into smaller sub-problems. However, this approach suffers from efficiency bottlenecks due to the need for extensive pre-training and brute-force search, which becomes computationally infeasible for complex systems. Though there are some works related to graph-based representation of expressions, such as [4] which establishes the NP-hard nature of symbolic regression by linking symbol graph to degree-constrained Steiner arborescence problem, or Expression Directed Acyclic Graphs (DAGs) [18] which is proposed to capture distributive equivalence and improve SR scalability, they fail to model the general equivalences, capture high-order dependencies, and preserve permutation invariance, as key insights in our paper. Consequently, an effective expression representation that reduces the state's redundancy and facilitates richer structural encoding remains an open problem in SR.

### 2.3 Constraint Incorporation

Constraints are ubiquitous in SR problems, which confine exploration to valid regions and guide SR for meaningful and robust solutions. Failure to incorporate constraints properly can result in sparse rewards and invalid solutions, particularly in real-world applications governed by fundamental physical laws. We classify constraints into two categories:

**Operation-level Constraints:** Constraints that can be evaluated based on a few operations during sampling, such as basic mathematical rules, void operations, maximum complexity (*i.e.*, the number of operations and variables), and hand-crafted priors for *a priori* known physics constraints. These constraints are typically incorporated as priors in the sampling stage by zeroing out probabilities of actions that would generate invalid expressions, preventing infeasible candidates from being explored.

**Expression-level Constraints:** Constraints that require a complete expression for evaluation, such as hidden mathematical constraints (e.g., $A \div ?$ becomes invalid if "?" is later sampled as $f(B) + C$ and $f(B) + C = 0$), constants fitting constraints, and physics-based penalty functions (e.g., enforcing $f(r \to 0) = \infty$ as discussed in Section 5.2). These constraints are typically incorporated as *post hoc* reward penalties, such as assigning zero rewards to invalid expressions after they are fully generated.

Existing SR approaches [11, 12, 19] handle constraints by integrating operation-level constraints as hand-crafted priors during the operation sampling stage and expression-level constraints via reward penalty functions. However, the absence of expression-level constraint priors during sampling, especially in random roll-outs in MCTS and random crossover in GP, leads to inefficient exploration and sparse rewards, making it difficult for SR models to converge. While unit consistency can be modeled as an operation-level constraint [19], the constant fitting process required for unit consistency remains an expression-level constraint, necessitating computationally expensive evaluations for every sample. This results in low efficiency and limits the scalability of SR methods. Hence, a more efficient way to incorporate expression-level constraint priors into the sampling stage remains an open challenge in SR. Additional related work on SR methods is reviewed in Appendix B.

## 3 Methods

To resolve the inefficient search limitations in symbolic regression (SR) as discussed in Section 2, we propose Graph-based Symbolic Regression (GSR) with hybrid neural-guided Monte Carlo tree search (hnMCTS) to address both redundant search space and incomplete constraint incorporation, with an improved convergence and exploitation and exploration trade-off.

### 3.1 Expression Graph Representation

**Expression Graph.** The redundant search space originates from the failure to capture the expression equivalencies and preserve the permutation invariance. Consequently, we propose the expression graphs (EGs), denoted as $\mathcal{G}$, which are unique expression representations aggregating their equivalent expressions, regardless of the generating sequence they went through during the sampling.

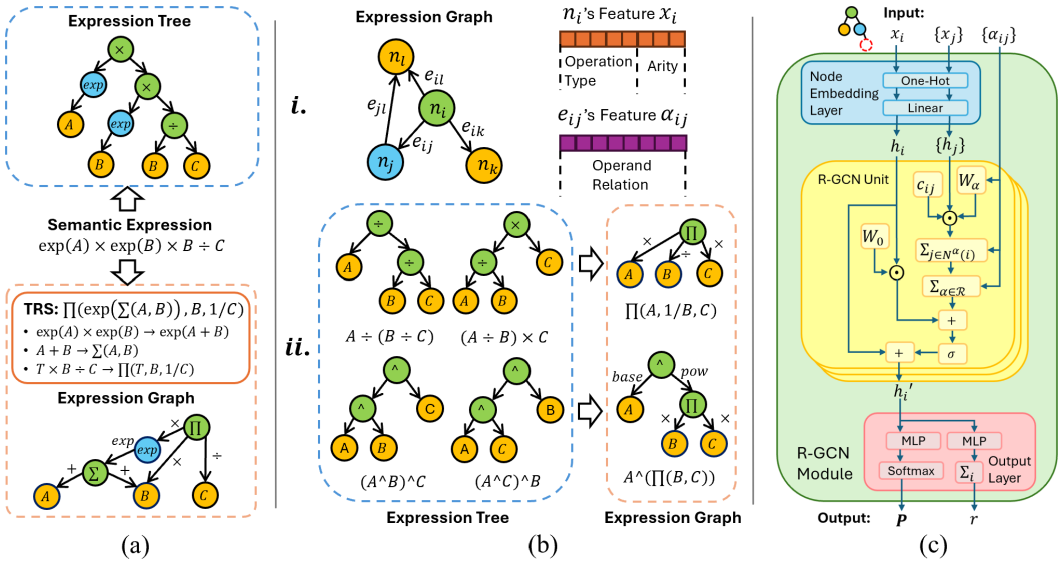



Figure 2: (a) Comparisons between the semantic expression representation, the ET representation (blue block), and the EG representation (orange block, where bullet points are the TRS rewrite rules). (b) Examples of (i) node and edge features of an EG, and (ii) EGs to unify the binary operations and align the commutative operands at the same level for permutation invariant representations. (c) An example of R-GCN encoder architecture for our EG representation, where node $j$ is the neighbor of node $i$, $N^{\alpha}(i)$ represents all neighbors of $i$ with the operand relation $\alpha$, $\odot$ represents the element-wise multiplication, $W_0$ and $W_{\alpha}$ are self-loop weight and edge-dependent weight respectively, $c_{ij}$ denotes the coefficient for $W_{\alpha}$, and $\sigma$ denotes the activation function.

As shown in Fig. 2 (a), the construction of an EG involves two procedures: (1) implementing the **term rewriting system (TRS)** with rewrite rules for equivalences, detailed in Appendix C.2, which captures the expression equivalences through recursively normalizing the current sampled candidate expression into its canonical form; and (2) converting the canonical form of the semantic expression into a **directed acyclic graph (DAG)-based structure**, which intrinsically encodes the permutation invariance through unification of the binary operations (e.g. $(+,-) \to \sum$ and $(\times,\div) \to \prod$) and aligning commutative operands at the same level, as shown in Fig. 1 and 2 (b). Note that for each EG, the node set $\mathcal{V} = \{n_i\}$ represents sampled operators, and the edge set $\mathcal{E} = \{e_{ij}\}$ encodes relationships between operators and operands. Compared with ET, EG with the DAG structure extended its capability for richer structure information (e.g., additional node features of arity and edge features of operand relation $\alpha$) to differentiate operands' roles (e.g., base or power for $\wedge$, $+$ or $-$ for $\sum$); and for improved scalability in handling complicated and highly dimensional datasets by node-sharing of the same variable node. Theorem 3.1 establishes the uniqueness of the EG representation, and Appendix C.2 provides an extended discussion of the EG representation.

**Theorem 3.1.** *Let $T(\Phi)$ be the set of expressions over $\Phi$, and $R$ be the TRS defined in Appendix C.2. Terms $t_1, t_2 \in T(\Phi)$ are equivalent ($t_1 \sim_R t_2$) if they rewrite to a common term under $\leftrightarrow_R^*$ (reflexive-*

*transitive-symmetric closure of R). $EG : T(\Phi) \to \mathcal{G}$ is a function normalized via bottom-up $R$ application. The EG representation is unique if for all $t_1 \sim_R t_2$, $EG(t_1) \cong EG(t_2)$ (isomorphism).*

*Proof Sketch.* For uniqueness, we show that $R$ is a terminating and confluent TRS, implying unique normal forms (irreducible terms under $\to_R$). The EG then encodes this normal form as a minimal DAG with shared substructures, preserving uniqueness. Appendix C.2 details the complete proof.

**Graph Neural Network Encoder.** Considering the DAG structure of EG and discrete edge features capturing the operands' relations, we propose a Relational Graph Convolutional Network (R-GCN) [20], a novel graph neural network (GNN) architecture [21, 22], as the foundation encoder in GSR. Its edge type-dependent message passing not only preserves the permutation invariance for EG during sampling; more importantly, the unique message passing depending on different operand relations also creates a more accurate and expressive representation than traditional encoders with ET. Figure 2(c) shows an example of the R-GCN architecture, and Appendix C.3 provides the explanations for the detailed message passing updates. Given an EG $\mathcal{G}$ as input, R-GCN encodes the expression state as the EG representation, formally expressed as:

$$\text{R-GCN}(\mathcal{G}\{\mathcal{V}, \mathcal{E}\}) = (\pi_\theta|_{\mathbf{P}}(\mathcal{G}), \pi_\theta|_r(\mathcal{G})) = (\mathbf{P}, r), \tag{2}$$

where $\mathbf{P} \in \mathbb{R}^{|\mathcal{V}|\times|\Phi|}$ is a prior probability matrix, with each row corresponding to the prior probability distribution for each $v \in \mathcal{V}$, and each column in the distribution representing the prior probability for each operator $\phi \in \Phi$ to be added to the node. $r \in \mathbb{R}^+$ is the predicted reward value of the state $\mathcal{G}$. $\pi_\theta|_{\mathbf{P}}$ and $\pi_\theta|_r$ are trainable models predicting $\mathbf{P}$ and $r$ given $\mathcal{G}$, with parameters $\theta$ of R-GCN. By self-learning with hnMCTS, EG representations encode expression-level constraint priors into $\mathbf{P}$ and $r$, providing structure-aware and constraint-informed neural guidance during the sampling.

### 3.2 Hybrid Neural-guided Monte Carlo Tree Search

To facilitate the EG-encoded constraint prior with MCTS [8], we focus on neural-guided MCTS [14, 23], which enables self-learning of R-GCN on EGs from MCTS. However, the pure neural-guided MCTS suffers the initial training bottlenecks due to the biased estimates of the under-trained R-GCN module. Consequently, we propose a new **hybrid neural-guided MCTS (hnMCTS)**, which adopts an adaptive **$\epsilon$-UCB policy** to dynamically combine the vanilla MCTS with neural-guided MCTS, to provide a warm-start training, faster convergence, and the balanced exploitation and exploration.

**MDP settings.** We model the expression generation process as a finite-horizon MDP with the sampling trajectory $\tau = \{s_0, a_0, s_1, a_1, \ldots, s_t, a_t, \ldots, a_{H-1}, s_H\}$, where $H$ is the maximum complexity. At step $t$, we define the state $s_t$ to be the current EG, $\mathcal{G}_t$, with the sampling node; action $a_t$ to be the sampled operator or variable $\phi \in \Phi$ added to $s_t$. When $\tau$ is complete (attaining the maximum complexity $H$ or finishing the expression in a closed form), we obtain the finalized expression $f_\tau$. Then we get a reward $R(\tau)$ through evaluating $\tau$ with the metrics in Appendix C.7.

Figure 3 illustrates the workflow of hnMCTS. To sample $\tau$, at each step $t$ in $\tau$, we perform one batch of hnMCTS for each state $s_t$ to update MCTS policy $\pi_M^t$ so that we can sample $a_t \sim \pi_M^t(s_t)$. During each iteration of the batch, hnMCTS samples a trajectory $\tau_t = \{s_t^0, a_t^0, s_{t+1}^1, a_{t+1}^1, \ldots, s_{t+i}^i, a_{t+i}^i, \ldots, a_{H-1}^{H-t-1}, s_H^{H-t}\}$ from the root state $s_t$ as the search, where the superscript denotes the search step in $\tau_t$ and subscript denotes the current complexity.

**Steps in hnMCTS.** As shown in Fig. 3, hnMCTS will switch between the vanilla and the neural-guided MCTS at the beginning of each $\tau_t$ by the $\epsilon$-UCB policy, with the random variable $\epsilon \in [0, 1]$ and the given threshold $c_\epsilon \in (0, 1]$. Then, it will take four steps based on the selected policy.

*Selection:* Starting from the root state $s_t$, hnMCTS selects the next step iteratively before arriving at an expandable node or terminal node following the maximum Upper Confidence Bounds (UCB) [24] or the maximum predictor-based UCB (PUCB) policy based on the selection of the $\epsilon$-UCB policy. Define $b$ as the next possible action, $C_U$ and $C_P$ as exploration rate controlling hyperparameters, $Q(s, a)$ as the expected action value, $P(s, a)$ as prior probability, and $N(s, a)$ as the total number of

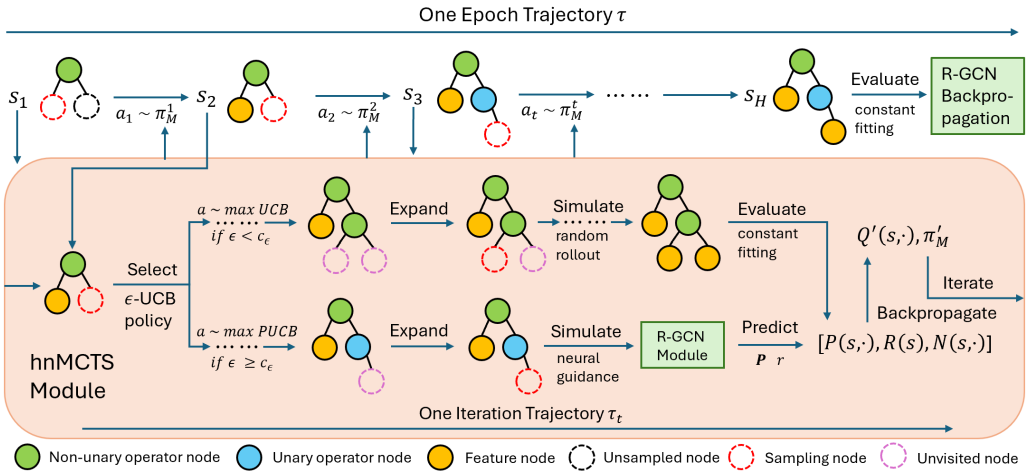



Figure 3: An exemplar workflow of hnMCTS with R-GCN on EG for neural-guided MCTS.

times for taking action $a$ from state $s$, we have the $\epsilon$-UCB policy:

$$\epsilon\text{-UCB policy}: a \sim \begin{cases} \arg\max_a \{\text{UCB}(s,a) = Q(s,a)/N(s,a) + C_U \sqrt{\frac{ln \sum_b N(s,b)}{N(s,a)}}\} & \text{if } \epsilon < c_\epsilon \\ \arg\max_a \{\text{PUCB}(s,a) = Q(s,a) + C_P P(s,a) \frac{\sqrt{\sum_b N(s,b)}}{1+N(s,a)}\} & \text{if } \epsilon \geq c_\epsilon \end{cases} \tag{3}$$

*Expansion:* If hnMCTS traverses to a visited node with unvisited children, we call it an expandable node. We will select one of its unvisited children for the following simulations.

*Simulation:* After traversing to an unvisited node, based on the selection of the $\epsilon$-UCB policy, hnMCTS will either have the uniform prior and calculate $R(s)$ through random rollout as the vanilla MCTS; or obtain $(\mathbf{P}(s,\cdot), R(s)) = (\pi_\theta|_{\mathbf{P}}(s), \pi_\theta|_r(s))$ through the R-GCN encoding on EGs as the neural-guided MCTS, where $\mathbf{P}(s,\cdot)$ is the prior distribution $\mathbf{P}$ and $R(s)$ is the expected reward $r$ of state $s$ according to (2), which incorporates expression-level constraint priors directly into sampling process in PUCB(3) by the self-learn mechanism according to (4).

*Backpropagation:* After obtaining $R(s)$ of the expandable node, hnMCTS will increase $N(s,a)$ by one and update $Q(s,a) = (1/N(s,a)) \sum_{s'|s,a} R(s')$ for all those $(s,a)$ pairs that have been traversed through. Here, $s'$ represents the next state of $s$ by taking action $a$.

After one batch of hnMCTS, we can update $\pi_M^t = N(s_t,\cdot)/\sum_b N(s_t,b)$ to step next in $\tau$. When $\tau$ is completed, we subsequently update the parameters $\theta$ of R-GCN by minimizing the loss function:

$$l = \sum_t ((R(\tau) - \pi_\theta|_r(s_t))^2 - \pi_M(s_t) \log \pi_\theta|_{\mathbf{P}}(s_t)). \tag{4}$$

Appendix C.4 outlines the proposed hnMCTS algorithm. With the dynamically controlled $\epsilon$-UCB policy, hnMCTS provides: (1) expression-level constraints incorporated proactively via R-GCN priors $(\mathbf{P}, r)$ within PUCB (3), densifying rewards without evaluations on full expressions; (2) a smooth warm-start for training, initially relying on random roll-outs to explore broadly, and progressively shifting to neural-guided MCTS as R-GCN training stabilizes; (3) the reduced reliance on costly constant fitting of evaluations via R-GCN predictions, significantly reducing computational overhead; (4) an improved convergence established in Theorem 3.2, and a balanced exploration-exploitation trade-off by controlling $c_\epsilon$ for the $\epsilon$-UCB policy as explained in Appendix C.5.2.

**Theorem 3.2.** *Given bounded rewards $R(s,a) \in (0,1]$, a finite action space $\mathcal{A}$, $C_U > 0$, $C_P > 0$, unbiased simulations, and $0 < c_\epsilon \leq 1$, hnMCTS with the $\epsilon$-UCB policy ensures $\lim_{T\to\infty} N_T(s,a) = \infty$ for all $a \in \mathcal{A}$, $N_T(s,a) = O(\ln T)$ for $a \neq a^*$(the optimal action), and $N_T(s,a^*) \sim T$.*

*Proof Sketch.* The $\epsilon$-UCB policy with $0 < c_\epsilon \leq 1$ alternates between the vanilla (UCB) and the neural-guided (PUCB) MCTS, ensuring infinite exploration of all actions due to UCB's exploration term, even if the neural prior $P(s,a^*) = 0$. Suboptimal actions have logarithmic visit counts $(E[N_T(s,a)] = O(\ln T))$ while the optimal action's visits grow linearly $(E[N_T(s,a^*)] \sim T)$, driven by $Q$-values from unbiased simulations. This guarantees policy convergence $(\pi_M^T(s,a^*) \to 1)$ with a relaxed condition compared to neural-guided MCTS. Appendix C.5.1 provides the complete proof.

# 4 Experiments

## 4.1 Synthetic Dataset Benchmarking

**Benchmarking Datasets:** we have evaluated our GSR on diverse benchmark datasets, including 1) the black-box dataset consisting of 120 tasks without solution expressions from PMLB [25], 2) Feynman dataset [11] consists of 119 physics-informed problems with ground-truth solutions, 3) Nguyen's SR benchmark dataset [26] and Nguyen constant dataset [7]. For the black-box dataset and the Feynman dataset, we follow the experimental settings of state-of-the-art symbolic regression benchmarking SRBench [27], which includes 21 baseline models for the black-box dataset benchmarking and 14 baseline models for the Feynman dataset benchmarking. Additionally, we compared with the recent transformer-based MCTS method, Transformer-based Planning for Symbolic Regression (TPSR) [10], in the black-box dataset; and a neural-guided GP method, A Unified Framework for Deep Symbolic Regression (uDSR) [9], in the Feynman dataset; as well as a transformer-based GP method, Deep Generative Symbolic Regression (DGSR) [6], for both datasets. We summarize our additional experimental settings in Appendix D.1

**Benchmarking Performances:** Figure 4 summarizes the benchmarking results on both the black-box and Feynman datasets. On the complex black-box problems, GSR consistently achieves higher $R^2$ scores than most baselines (Fig. 4(a)), indicating stronger predictive accuracy in settings where expression structures are not explicitly provided. On the Feynman dataset, GSR recovers a larger number of ground-truth expressions (Fig. 4(d)), demonstrating improved ability to identify physically meaningful symbolic forms rather than merely fitting numerical outputs.

In terms of model complexity (Fig. 4(b)), GSR attains relatively compact expressions while maintaining competitive accuracy. Regarding computational efficiency, GSR exhibits training times comparable to the pretrained transformer-based TPSR method [10] (Fig. 4(c)), despite not relying on any pretraining or large auxiliary datasets, suggesting that the canonical expression graph representation effectively reduces redundant structures during search.

Overall, these results indicate that encoding invariance and expression-level constraints directly into the search process improves the trade-off between accuracy, interpretability, expression simplicity, and computational cost, particularly on more challenging benchmarks. Additional results on the Nguyen benchmark are reported in Appendix D.2.

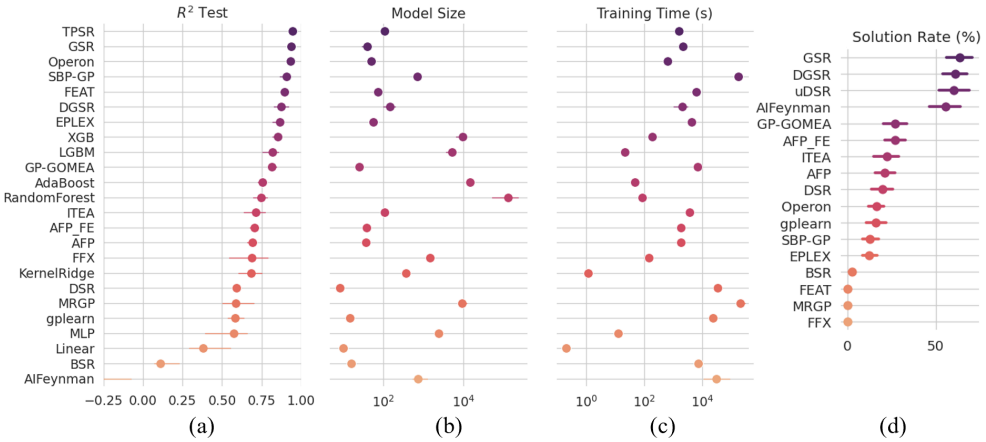



Figure 4: Performance comparisons of 24 algorithms regarding: (a) $R^2$ test score, (b) model size (solution complexity), and (c) training time on the black-box dataset; Performance comparisons of 17 algorithms regarding: (d) solution rate (the ratio of ground-truth equivalent solutions in mathematics to the total equations in the dataset) on the Feynman dataset. The dots represent the median value, and the shadow line represents the 95% confidence interval.

## 4.2 Ablation Study

**Ablation Settings:** To evaluate the individual contributions of invariance and expression-level constraint encoding in GSR, we conduct ablation studies on six challenging symbolic regression

| Model | GSR | MCTS-EG | hnMCTS-ET | MCTS |
|---|---|---|---|---|
| Invariance Encoding | ✓ | ✓ | × | × |
| Expression-level Constraint Encoding | ✓ | × | ✓ | × |
| **Solution Rate** (%) ↑ | **98.0** | 90.3 | <u>92.7</u> | 85.8 |
| **Number of Evaluations** ↓ | **89,128** | <u>107,651</u> | 120,056 | 162,783 |
| **Training time** ($s$) ↓ | **1302.9** | 2089.2 | <u>1759.3</u> | 2782.6 |
| **Sparse Rate** (%) ↓ | **11.4** | 21.2 | <u>16.5</u> | 25.3 |

Table 1: Comparisons between GSR and three ablation models regarding the average solution rate, number of evaluations, the training time for the first solution, and the sparse rate (ratio of invalid generated expressions to the total expressions generated in an epoch) at 50% of the total training epochs on six tasks (Nguyen-12 and five Nguyen benchmark with constant tasks).

tasks drawn from the Nguyen benchmark with constants [7] and the Nguyen-12 task from the original Nguyen benchmark [26], under consistent experimental settings described in Appendix D.2. In addition to GSR, we compare the following ablation models: 1) MCTS-EG, employing the EG representation for invariance encoding but using standard MCTS without constraint encoding; 2) hnMCTS-ET, using hnMCTS for constraint encoding but relying on ET representation without invariance encoding; 3) MCTS, a baseline using vanilla MCTS with ET, lacking both invariance and constraint encoding.

**Performance Analysis:** Table 1 reports the results of the ablation study. Relative to vanilla MCTS, hnMCTS-ET, which incorporates expression-level constraint encoding, achieves higher solution rates, improved training efficiency, and reduced reward sparsity. These gains indicate that constraint-guided sampling, together with the expressive capacity of the R-GCN encoder, provides more informative guidance during search, particularly for tasks involving constants and complex constraint structures.

In contrast, MCTS-EG requires substantially fewer function evaluations than vanilla MCTS, with a 33.9% reduction in the effective search space, demonstrating that equivalence-aware expression graph representations reduce redundant exploration through state-space compression. However, its overall training time remains higher than hnMCTS-ET, as constant fitting must still be performed explicitly during search, limiting the computational savings from reduced evaluations.

By integrating both expression graph–based invariance encoding and constraint-guided neural search, GSR achieves further improvements in solution rate, evaluation efficiency, and reward sparsity mitigation. This result highlights the complementary roles of invariance-aware state compression and expression-level constraint guidance, enabled by edge-aware message passing of the R-GCN over structurally enriched expression graphs. Additional analyses of reward sparsity and supporting visualizations are provided in Appendix D.4.

# 5 Application

To demonstrate GSR's scalability and constraint-aware capability in highly complex, irregular, and physics-constrained real-world settings, in this section, we apply it to a materials science application focused on predicting interatomic potential energy for atomistic simulations. This problem is central to understanding material behavior and accelerating the design and discovery of novel materials. While traditional first-principles methods such as density functional theory (DFT) [28, 29], offer high accuracy, their use is often limited by the substantial computational cost and extensive memory requirements. To address this, surrogate ML models are increasingly explored, with symbolic regression showing strong potential for balancing efficiency and interpretability. Appendix E.1 details the problem backgrounds, and Appendix E.2 describes the practical usage of GSR in this domain.

## 5.1 Problem Setting

We adopt a dataset from first-principles molecular dynamics simulation of 32 copper atoms from [16], which is described in detail in Appendix E.4. We aim to discover the interatomic potential energy function $f(\cdot)$ that effectively maps the atoms' pairwise distances $\boldsymbol{r}$ to a total formation energy $E$. In this study, we consider four methods to learn the interatomic potential energy function: 1) a GNN-based black-box method, CGCNN [30]; 2) a genetic programming-based SR method from [16]; 3) our GSR in this paper; and 4) the vanilla MCTS method. GP1 and GP2 in Table 2 are two fitted

expressions by genetic programming reported in [16]. Appendix E.4 summarizes our experimental settings for baseline models.

### 5.2 Physics Constraints

To ensure the resulting $f(\cdot)$ is physically meaningful, we consider the following additional physics constraints besides the universal constraints listed in Section 2.3:

**(1) Scalar Output** : The output of $f(\boldsymbol{r})$ has to be a single scalar to match $E$ for valid evaluation.

**(2) Unit Consistency** : A scalar $const$ should be multiplied before any polynomial terms and in any number in the exponential terms to ensure the unit calculation aligns with the physical meaning.

**(3) Electrostatic Repulsion** : The energy approaches infinity at an infinitesimal distance (i.e. $f(r \to 0) = \infty$) due to Coulomb's law.

We include explanations for all constraints in Appendix E.3. Note that CGCNN only satisfies the scalar output (1) constraint; GP1 and GP2 satisfy both the scalar output (1) and unit consistency (2) constraints; and MCTS and GSR satisfy all the constraints.

### 5.3 Performance and Transferbility

**Test Results:** Table 2 presents our comparison results for the formation energy prediction, where the Mean Absolute Error (MAE) is computed based on linear regression results of the model expressions regarding the formation energy $E$, divided by the number of atoms (32) in the crystal. Figure 5 depicts curves of the interatomic potential energy functions $f(r)$ according to the learned analytical expressions in Table 2. It illustrates the electrostatic repulsion that the energy goes to infinity $f(r) \to \infty$ at short interatomic distance ($r \to 0$), with more details described in Appendix E.6.

Comparing training and testing MAEs in Table 2, GSR achieves the best testing MAE while preserving all physical constraints (the $1/r^m$ terms reflect electrostatic repulsion at an infinitesimal distance). MCTS, though satisfying all the constraints as GSR, its largest MAE and highest complexity highlight the impact of sparse rewards to vanilla MCTS during random rollouts. In contrast, GSR guides MCTS with encoded physics constraints that improve the likelihood of generating valid expressions during simulations. Although CGCNN has the lowest training MAE, its larger testing MAE indicates the potential overfitting risk, showing less robustness than SR methods such as GSR and GP2 in data-limited scenarios. GP1's low complexity comes at the cost of significantly poor performance,

| **Model** | **Constraints** | **Expression** $f(r)$ (eV/atom) | **Complexity** $\downarrow$ | **MAE** (meV/atom) $\downarrow$ | |
|---|---|---|---|---|---|
| | | | | **Train** / **Test** | **Transfer** |
| CGCNN | (2)× (3)× | - | - | **2.08** / 3.09 | 44.89 |
| GP1 | (2)✓ (3)× | $\sum(r^{10.21-5.47r} - 0.21^r)s(r)$ $+0.97(\sum 0.33^r s(r))^{-1}$ | 21 | 3.68 / 3.53 | 43.32 |
| GP2 | (2)✓ (3)× | $7.33\sum r^{3.98-3.94r}s(r) + (27.32 - \sum$ $(11.13 + 0.03r^{11.74-2.93r})s(r))(\sum s(r))^{-1}$ | 28 | 2.57 / 2.70 | 41.63 |
| MCTS | (2)✓ (3)✓ | $\sum(13.70r^3 + 27.18r^2 - 13.70re^r)e^{-0.98r^2}$ $+\sum 2.98r^{-1}e^{-12.87r}$ | 31 | 3.91 / 3.65 | 36.31 |
| GSR | (2)✓ (3)✓ | $\sum 6.04 \times 10^{-11}(r^3e^{3r} - r^{12})$ $+\sum 121.41(r^{-7} + 3r^{-8} - r^{-6})$ | 24 | 2.41 / **2.63** | **34.15** |

Table 2: Analytical interatomic potential energy functions $f(r)$ fitted to DFT total energy using GP1, GP2, MCTS, and GSR compared with the CGCNN model. Here, $f(r)$ is in the unit of $eV/atom$, and $s(r)$ is the smoothing function introduced in Eq (7) of [16]. "Constraints" denotes the satisfied constraints defined in Section 5.2. "Complexity" is computed by the sum of nodes in the expression graph or the expression tree. "Train/Test" represents the training and testing MAE of the copper dataset. "Transfer" represents the MAE of expressions directly tested on the newly generated dataset as described in Appendix E.7. Typically, DFT achieves an MAE of $10 \sim 20$ meV/atom for energy predictions for reference.

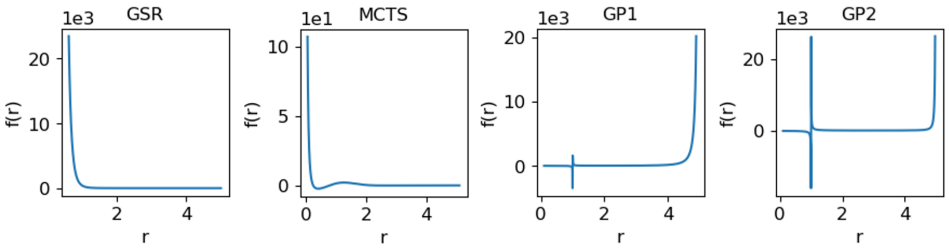



Figure 5: Interatomic potential energy functions $f(r)$ learned by GSR, MCTS, GP1 and GP2, as a function of interatomic distance $r$ ranging between 0 to 5Å. The $x$-axis is the interatomic distance $r$ and the $y$-axis is the interatomic potential energy. Only MCTS and GSR produce physically meaningful results, adhering to the strong electrostatic repulsion at short interatomic distances, while both GP1 and GP2 fail. A finite-valued model at short interatomic distances, like GP1 and GP2, allows atoms to overlap, which can either cause downstream dynamics simulations to crash or lead to meaningless results violating physics laws.

while GP2, despite the comparable predictive power to GSR, has higher complexity that makes the underlying dependencies less straightforward. Besides, none of CGCNN, GP1, or GP2 adheres to the electrostatic repulsion constraint, resulting in less physically meaningful solutions at short interatomic distances. Overall, GSR demonstrates strong expressive power with lower complexity than GP methods and reduced overfitting compared to neural networks, while consistently maintaining physical meaning. Plots for Table 2 results are included in Appendix E.5.

**Transfer Results:** To further assess the transferability of the interatomic potential energy model, we create a new dataset of 100 samples by performing DFT-based molecular dynamics simulations where a volumetric compression of approximately 50% is applied to the unit cell of the original copper dataset. As a result, the new dataset consists of samples with shorter interatomic distances (bond lengths) between neighboring atoms, as described in Appendix E.7. This allows us to assess the transferability of our physics-constrained analytical model compared to other conventional machine learning models. The "Transfer" MAE in Table 2 is obtained through zero-shot learning of the solution expressions directly tested on the newly generated dataset, where GSR and MCTS show significantly lower MAEs. As electrostatic repulsion has a direct impact at shorter interatomic distances, GSR and MCTS that integrate the physical constraints provide better generalization to shorter bond lengths on the new dataset. These results again verify that the incorporated physics constraints can prevent overfitting and foster the discovery of generalizable knowledge.

## 6 Conclusion and Limitation

In this work, we identified two critical challenges in current Symbolic Regression (SR) approaches: (1) *redundant expression representations* that fail to capture expression equivalences and preserve the permutation invariance, and (2) *the absence of expression-level constraint priors* incorporated during the sampling stage, leading to low search efficiency, reward sparsity, and limited applicability in real-world settings. To address these challenges, we proposed Graph-based Symbolic Regression (GSR) – a method built on Relational Graph Convolutional Network (R-GCN) operating over expression graphs (EGs); and a hybrid neural-guided Monte Carlo Tree Search (hnMCTS) framework with an adaptive $\epsilon$-UCB policy. EGs provide invariance-aware representations and higher-order structural features, enabling R-GCN to encode constraint priors more effectively. hnMCTS with R-GCN encoding facilitates a balanced exploration-exploitation trade-off, reduces reward sparsity, and enhances search efficiency.

Comprehensive benchmarks and ablation studies on synthetic datasets confirm that GSR achieves superior performance with a compressed search space. Furthermore, a real-world application in materials science, involving complex physics-based constraints, demonstrates GSR's capability to produce interpretable and physically consistent solutions, underscoring its practical value.

Limitations remain for the GSR, where the term rewriting system (TRS) is a handcrafted rule set that captures most of the basic algebraic equivalences but not higher-order symmetries or domain-specific invariances. Looking forward, extending GSR with automated or learned TRS rules and pretrained symmetry-aware graph encoders could further enhance generalization and cross-domain applicability.

## Acknowledgment

Z. X. and X.N.Q. acknowledge the support from U.S. National Science Foundation (NSF) grants DMREF-2119103, SHF-2215573, and IIS-2212419. First-principles calculations by K.A. were supported by the Center for Reconfigurable Electronic Materials Inspired by Nonlinear Dynamics (reMIND), an Energy Frontier Research Center funded by the Department of Energy under award DE-SC0023353. X.F.Q. acknowledges the support from NSF under grant CMMI-2226908. Portions of this research were conducted with the advanced computing resources provided by Texas A&M High Performance Research Computing.

## References

[1] Dimitrios Angelis, Filippos Sofos, and Theodoros E Karakasidis. Artificial intelligence in physical sciences: Symbolic regression trends and perspectives. *Archives of Computational Methods in Engineering*, 30(6):3845–3865, 2023.

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

[3] Yiqun Wang, Nicholas Wagner, and James M Rondinelli. Symbolic regression in materials science. *MRS Communications*, 9(3):793–805, 2019.

[4] Jinglu Song, Qiang Lu, Bozhou Tian, Jingwen Zhang, Jake Luo, and Zhiguang Wang. Prove symbolic regression is NP-hard by symbol graph. *CoRR*, abs/2404.13820, 2024.

[5] Marco Virgolin and Solon P Pissis. Symbolic regression is NP-hard. *Transactions on Machine Learning Research*, 2022.

[6] Samuel Holt, Zhaozhi Qian, and Mihaela van der Schaar. Deep generative symbolic regression. In *The Eleventh International Conference on Learning Representations*, 2023.

[7] Brenden K Petersen, Mikel Landajuela Larma, Terrell N Mundhenk, Claudio Prata Santiago, Soo Kyung Kim, and Joanne Taery Kim. Deep symbolic regression: Recovering mathematical expressions from data via risk-seeking policy gradients. In *International Conference on Learning Representations*, 2020.

[8] Fangzheng Sun, Yang Liu, Jian-Xun Wang, and Hao Sun. Symbolic physics learner: Discovering governing equations via Monte Carlo Tree Search. In *The Eleventh International Conference on Learning Representations*, 2023.

[9] Mikel Landajuela, Chak Shing Lee, Jiachen Yang, Ruben Glatt, Claudio P Santiago, Ignacio Aravena, Terrell Mundhenk, Garrett Mulcahy, and Brenden K Petersen. A unified framework for deep symbolic regression. *Advances in Neural Information Processing Systems*, 35:33985–33998, 2022.

[10] Parshin Shojaee, Kazem Meidani, Amir Barati Farimani, and Chandan Reddy. Transformer-based planning for symbolic regression. *Advances in Neural Information Processing Systems*, 36:45907–45919, 2023.

[11] Silviu-Marian Udrescu and Max Tegmark. AI Feynman: A physics-inspired method for symbolic regression. *Science Advances*, 6(16):eaay2631, 2020.

[12] Liron Simon Keren, Alex Liberzon, and Teddy Lazebnik. A computational framework for physics-informed symbolic regression with straightforward integration of domain knowledge. *Scientific Reports*, 13(1):1249, 2023.

[13] 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*, pages 15655–15668. PMLR, 2023.

[14] David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, et al. Mastering the game of GO without human knowledge. *nature*, 550(7676):354–359, 2017.

[15] Terrell N. Mundhenk, Mikel Landajuela, Ruben Glatt, Claudio P. Santiago, Daniel faissol, and Brenden K. Petersen. Symbolic regression via deep reinforcement learning enhanced genetic programming seeding. In A. Beygelzimer, Y. Dauphin, P. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, 2021.

[16] Alberto Hernandez, Adarsh Balasubramanian, Fenglin Yuan, Simon AM Mason, and Tim Mueller. Fast, accurate, and transferable many-body interatomic potentials by symbolic regression. *npj Computational Materials*, 5(1):112, 2019.

[17] Silviu-Marian Udrescu, Andrew Tan, Jiahai Feng, Orisvaldo Neto, Tailin Wu, and Max Tegmark. AI Feynman 2.0: Pareto-optimal symbolic regression exploiting graph modularity. *Advances in Neural Information Processing Systems*, 33:4860–4871, 2020.

[18] Paul Kahlmeyer, Joachim Giesen, Michael Habeck, and Henrik Voigt. Scaling up unbiased searchbased symbolic regression. In *Proceedings of the Thirty-Third International Joint Conference on Artificial Intelligence, IJCAI-24*, pages 4264–4272, 2024.

[19] Wassim Tenachi, Rodrigo Ibata, and Foivos I Diakogiannis. Deep symbolic regression for physics guided by units constraints: toward the automated discovery of physical laws. *The Astrophysical Journal*, 959(2):99, 2023.

[20] Michael Schlichtkrull, Thomas N Kipf, Peter Bloem, Rianne Van Den Berg, Ivan Titov, and Max Welling. Modeling relational data with graph convolutional networks. In *The semantic web: 15th international conference, ESWC 2018, Heraklion, Crete, Greece, June 3–7, 2018, proceedings 15*, pages 593–607. Springer, 2018.

[21] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and S Yu Philip. A comprehensive survey on graph neural networks. *IEEE transactions on neural networks and learning systems*, 32(1):4–24, 2020.

[22] Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations*, 2017.

[23] Surag Nair. A simple Alpha(Go) Zero tutorial, 2017.

[24] Peter Auer. Using confidence bounds for exploitation-exploration trade-offs. *Journal of Machine Learning Research*, 3(Nov):397–422, 2002.

[25] Randal S Olson, William La Cava, Patryk Orzechowski, Ryan J Urbanowicz, and Jason H Moore. PMLB: a large benchmark suite for machine learning evaluation and comparison. *BioData mining*, 10:1–13, 2017.

[26] Nguyen Quang Uy, Nguyen Xuan Hoai, Michael O'Neill, Robert I McKay, and Edgar Galván-López. Semantically-based crossover in genetic programming: application to real-valued symbolic regression. *Genetic Programming and Evolvable Machines*, 12:91–119, 2011.

[27] 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.

[28] Pierre Hohenberg and Walter Kohn. Inhomogeneous electron gas. *Physical review*, 136(3B):B864, 1964.

[29] W. Kohn and L. J. Sham. Self-consistent equations including exchange and correlation effects. *Phys. Rev.*, 140:A1133–A1138, 1965.

[30] Tian Xie and Jeffrey C Grossman. Crystal graph convolutional neural networks for an accurate and interpretable prediction of material properties. *Physical Review Letters*, 120(14):145301, 2018.

[31] G. Kresse and J. Furthmüller. Efficient iterative schemes for *ab initio* total-energy calculations using a plane-wave basis set. *Phys. Rev. B*, 54:11169–11186, 1996.

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

[33] Michael Schmidt and Hod Lipson. Distilling free-form natural laws from experimental data. *Science*, 324(5923):81–85, 2009.

[34] Runhai Ouyang, Stefano Curtarolo, Emre Ahmetcik, Matthias Scheffler, and Luca M Ghiringhelli. SISSO: A compressed-sensing method for identifying the best low-dimensional descriptor in an immensity of offered candidates. *Physical Review Materials*, 2(8):083802, 2018.

[35] Steven L Brunton, Joshua L Proctor, and J Nathan Kutz. Discovering governing equations from data by sparse identification of nonlinear dynamical systems. *Proceedings of the national academy of sciences*, 113(15):3932–3937, 2016.

[36] Mojtaba Valipour, Bowen You, Maysum Panju, and Ali Ghodsi. SymbolicGPT: A generative transformer model for symbolic regression. *arXiv preprint arXiv:2106.14131*, 2021.

[37] Luca Biggio, Tommaso Bendinelli, Alexander Neitz, Aurelien Lucchi, and Giambattista Parascandolo. Neural symbolic regression that scales. In *International Conference on Machine Learning*, pages 936–945. Pmlr, 2021.

[38] Maciej Świechowski, Konrad Godlewski, Bartosz Sawicki, and Jacek Mańdziuk. Monte Carlo Tree Search: A review of recent modifications and applications. *Artificial Intelligence Review*, 56(3):2497–2562, 2023.

[39] Bruno Buchberger. Bruno Buchberger's PhD thesis 1965: An algorithm for finding the basis elements of the residue class ring of a zero dimensional polynomial ideal. *Journal of symbolic computation*, 41(3-4):475–511, 2006.

[40] Gérard Huet. Confluent reductions: Abstract properties and applications to term rewriting systems: Abstract properties and applications to term rewriting systems. *Journal of the ACM (JACM)*, 27(4):797–821, 1980.

[41] Peter Auer, Nicolo Cesa-Bianchi, and Paul Fischer. Finite-time analysis of the multiarmed bandit problem. *Machine learning*, 47:235–256, 2002.

[42] Yoshitomo Matsubara, Naoya Chiba, Ryo Igarashi, and Yoshitaka Ushiku. Rethinking symbolic regression datasets and benchmarks for scientific discovery. *Journal of Data-centric Machine Learning Research*, 2024.

[43] Bowen Deng, Peichen Zhong, KyuJung Jun, Janosh Riebesell, Kevin Han, Christopher J Bartel, and Gerbrand Ceder. CHGNet as a pretrained universal neural network potential for charge-informed atomistic modelling. *Nature Machine Intelligence*, 5(9):1031–1041, 2023.

[44] Songling Liu, Yonghao Sun, Zhaoyuan Liu, and Weihua Wang. Tetrahedral stacking of Copper in its incubation period of crystallization: A study of large-scale molecular dynamics simulation. *The Journal of Physical Chemistry C*, 127(44):21816–21821, 2023.

[45] G Makov and Mike C Payne. Periodic boundary conditions in *ab initio* calculations. *Physical Review B*, 51(7):4014, 1995.

[46] P. E. Blöchl. Projector augmented-wave method. *Phys. Rev. B*, 50:17953–17979, 1994.

[47] John P. Perdew, Kieron Burke, and Matthias Ernzerhof. Generalized gradient approximation made simple. *Phys. Rev. Lett.*, 77:3865–3868, 1996.

[48] Hendrik J. Monkhorst and James D. Pack. Special points for Brillouin-zone integrations. *Phys. Rev. B*, 13:5188–5192, 1976.

[49] Chad A McCoy, Marcus D Knudson, and Seth Root. Absolute measurement of the Hugoniot and sound velocity of liquid Copper at multimegabar pressures. *Physical Review B*, 96(17):174109, 2017.

[50] DE Fratanduono, RF Smith, SJ Ali, DG Braun, A Fernandez-Pañella, S Zhang, RG Kraus, F Coppari, JM McNaney, MC Marshall, et al. Probing the solid phase of noble metal Copper at terapascal conditions. *Physical Review Letters*, 124(1):015701, 2020.

## NeurIPS Paper Checklist

1. **Claims**

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

   Answer: [Yes]

   Justification: As detailed in Section 3 and 4, our main claims made in the abstract and introduction on our GSR's effectiveness and efficiency accurately reflect the paper's contributions and scope.

   Guidelines:

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

2. **Limitations**

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

   Answer: [Yes]

   Justification: Section Conclusions and Future Work briefly discussed our future research direction on addressing challenges in integrating more expression symmetries and a pretrained GNN encoder, which hopefully has stronger trustworthy guarantees on learned SR models.

   Guidelines:

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

3. **Theory assumptions and proofs**

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

Answer: [Yes]

Justification: We provide a complete proof of Theorem 3.1 in Appendix C.2 and a complete proof of Theorem 3.2 in Appendix C.5.1.

Guidelines:

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

4. **Experimental result reproducibility**

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

Answer: [Yes]

Justification: We have made all efforts to clearly describe our experimental settings in Section 4 and 5, as well as in Appendix D.1, C.6 and E.4. Multiple SR experimental runs have been conducted in Section 4 and 5, and their results have been explicitly reported.

Guidelines:

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

some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

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

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

   Answer: [Yes]

   Justification: Both the code and data will be released in an open-access repository after the paper is publicly available.

   Guidelines:

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

6. **Experimental setting/details**

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

   Answer: [Yes]

   Justification: We have made all efforts to clearly describe our experimental settings in Section 4 and 5, as well as in Appendix D.1, C.6 and E.4.

   Guidelines:

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

7. **Experiment statistical significance**

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

   Answer: [Yes]

   Justification: We have conducted SR experiments on the standard benchmarking dataset several times and reported their 95% confidence interval with the median value in Section 4.

   Guidelines:

   - The answer NA means that the paper does not include experiments.

- The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

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

   Answer: [Yes]

   Justification: The computing platform together with runtime is described in the Appendix A.

   Guidelines:

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

9. **Code of ethics**

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

   Answer: [Yes]

   Justification: We confirm that the research conducted and presented in the paper conforms, in every respect, with the NeurIPS Code of Ethics.

   Guidelines:

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

10. **Broader impacts**

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

    Answer: [Yes]

    Justification: We have provided a brief statement on potential societal impacts, especially related to materials science, in Section 5.

Guidelines:

- The answer NA means that there is no societal impact of the work performed.
- If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

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

    Answer: [NA]

    Justification: We do not foresee any potential risk of releasing related data and code as we focus on fundamental research in AI for Science and follow the commonly adopted practice in the field.

    Guidelines:

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

12. **Licenses for existing assets**

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

    Answer: [Yes]

    Justification: We have cited the references for all the original code, data, and models used in our paper.

    Guidelines:

    - The answer NA means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.

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

13. **New assets**

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

    Answer: [Yes]

    Justification: We have clearly described all the details of our GSR and the dataset we have generated via DFT in Sections 3, 5, and Appendix E.7.

    Guidelines:

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

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

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

    Answer: [NA]

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

    Guidelines:

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

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

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

    Answer: [NA]

    Justification: The presented work does not involve the risks listed above.

Guidelines:

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

16. **Declaration of LLM usage**

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

    Answer: [NA]

    Justification: The presented work does not involve important or original core methods originating from LLMs.

    Guidelines:

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

# Appendix

## A Computing Resources

We have performed our experiments on a platform with one CPU, Intel Xeon 6248R (Cascade Lake), 3.0GHz, 24-core, and one GPU, NVIDIA A100 40GB GPU accelerator. For the reported Synthetic Dataset Benchmarking in Section 4 in the main text, our graph-based symbolic regression (GSR) can sample 60 expressions per second on average. For the materials science application in Section 5, GSR on average samples 20 expressions per second.

First-principles DFT calculations of the test dataset were performed on a compute node consisting of two Intel Xeon 6248R (Cascade Lake) CPUs with a total of 48 cores and 384GB DDR4 memory. The initial supercell structure of 32-atom fcc copper was relaxed using VASP [31] with the conjugate gradient method in four steps within a total of 18 seconds. For each temperature, VASP-based molecular dynamics simulations in the NVT ensemble were run for 6,000 steps for a total of 900 minutes.

## B Additional Literature Review on Symbolic Regression Methods

Symbolic regression (SR) has a rich history with a diverse range of methods. Early ones primarily rely on Genetic Programming (GP) [16, 32, 33], which searches for candidate expressions through evolutionary processes such as selection, mutation, and crossover. Similar as brute-force methods like SISSO [34] and basis-dependent methods like SINDy [35], GP-based methods often suffer from poor scalability and high sensitivity to hyperparameters [7].

Modern SR methods integrate deep learning (DL) and reinforcement learning (RL) to improve heuristic search efficiency. DSR [7] employs a recurrent neural network (RNN) with risk-seeking policy gradients (PG), performing well on simple tasks but incurring high computational costs due to repeated RNN inference for every sample and limited exploration in complex problems. Neural-guided genetic programming methods [9, 15] address these issues by combining RL with GP, improving both efficiency and exploration. Deep generative models [6, 10, 36, 37] offer fast inference but lack adaptability to out-of-distribution datasets due to their static pre-training. In contrast, Monte Carlo Tree Search (MCTS)-based methods [8, 13, 38] have better exploration-exploitation balance, achieving state-of-the-art performance.

## C Methodological Detail

### C.1 Transplantation and Constant Optimization

**Transplantation:** Inspired by the cross-over mechanism in Genetic Programming (GP), we introduce the concept of function modularity to break down complex problems into smaller sub-problems [11, 8]. Leveraging a divide-and-conquer heuristic search strategy, we store promising expressions with lower complexity than the current sampling average in a budget set. These stored expressions can then be directly inserted into newly generated expressions as a single operator, which we refer to as the "transplantation" operator, denoted by $trans$. The $trans$ operator functions as a variable that GSR can choose during exploration. When selected, GSR randomly picks a subset of expressions from the budget and uses them to generate new expressions, selecting the best one to calculate the action value of the $trans$ operator. By default, GSR enables the $trans$ operator after half of the total epochs, with the budget set to store the top $n$ models that have a complexity lower than half of the maximum complexity.

**Constants Optimization:** Constants play a crucial role in symbolic regression, not only for maintaining unit consistency in real-world scientific problems but also for optimizing regression accuracy. In our approach, constants are introduced into expressions via the $const$ operator, functioning as a variable operator. While enabling constant optimization enhances the accuracy of the regression, it significantly increases the computational cost of evaluating an expression. To mitigate this, all constants within an expression are optimized only once per evaluation using a non-linear regression method, specifically the BFGS optimizer, as described in [7].

### C.2 EG Representation Detail

**EG as State.** To leverage the expression graph (EG) representation $\mathcal{G}$ as the input for GSR encoding, we can directly feed $\mathcal{G}$ into a Relational Graph Convolutional Network (R-GCN) due to the R-GCN's inherent aggregation and edge-dependent message-passing mechanisms, which naturally preserve the permutation invariance encoded in EG and exploit the discrete edge type in EGs. However, to use EG as a state in Monte Carlo Tree Search (MCTS), we first convert $\mathcal{G}$ into a canonical form by sorting the nodes. Next, we modify the upper triangle of the sorted adjacency matrix by replacing its elements with the corresponding edge features indexed by numbers, where "0" denotes a disconnected edge. The diagonal elements are replaced by the node features. Finally, we output this modified sorted adjacency matrix in its Voigt form, ensuring the representation of states while preserving symmetries.

| Rewrite Rule | Level | Equivalence |
|---|---|---|
| $a+b \to \sum(a,b)$<br>$a-b \to \sum(a,-b)$<br>$a \times b \to \prod(a,b)$<br>$a \div b \to \prod(a,1/b)$ | Operation | Commutative/<br>Associative equivalence. |
| $a \times (b+c) \to \sum(a \times b, a \times c)$ | Operation | Distributive equivalence. |
| $(a^b)^c \to a^{\prod(b,c)}$ | Operation | Power equivalence. |
| $\exp(a) \times \exp(b) \to \exp(a+b)$<br>$\log(a)+\log(b) \to \log(a \times b)$ | Operation | Exponential/<br>Logarithmic equivalence. |
| $a \div a \to 1$<br>$a \times 1 \to a$<br>$a+0 \to a$ | Expression | Identity equivalence. |
| $\sin^2(a)+\cos^2(a) \to 1$ | Expression | Trigonometric equivalence. |
| $a^2-b^2 \to (a-b)(a+b)$<br>$(a \pm b)^2 \to a^2 \pm 2ab + b^2$ | Expression | Square equivalence. |

Table 3: An exemplar of the TRS. Operation-level equivalences are judged through local operations without specific operand requirements, while expression-level equivalences require specific operands to satisfy the equivalence.

**Term Rewriting System (TRS).** Table 3 presents an example of the term rewriting system (TRS) with the rewrite rules and the corresponding equivalences for our expression graph (EG) representation. Equivalences are categorized into operation and expression-level ones, depending on whether they can be evaluated using local operations without specific operand requirements. Operation-level equivalences, such as the commutative property $a+b=b+a$, are identified solely based on the operator (e.g., addition or multiplication) and are encoded in EG by unifying operators into a single node type (e.g., $\sum$ or $\prod$) with unordered edge connections, preserving permutation invariance. In contrast, expression-level equivalences, such as the trigonometric identity $\sin^2(a)+\cos^2(a)=1$, require specific operands to be satisfied (e.g., $\sin^2(a)+\cos^2(b) \neq 1$) and are encoded only after the relevant operands are sampled, allowing EG to transform the expression into a simplified node structure. EG leverages node and edge features to directly encode operation-level equivalences during operator sampling, while expression-level equivalences are deferred until the complete expression is formed, enabling verification and transformation. Due to their specific operand dependency, expression-level equivalences occur less frequently than operation-level ones, resulting in operation-level equivalences dominating the reduction of redundant representations during the search process.

**Proof of Theorem 3.1.** Uniqueness requires $R$ convergent (confluent + terminating), ensuring unique normal forms $nf(t)$, and EG encodes the normal form uniquely.

- **Termination**: Finite expressions yield finite steps; $R$ satisfies Termination as rules are non-recursive (no loops recreating left-hand sides).

- **Confluence**: If $t \to^* u$ and $t \to^* v$, then there exists $w$ such that $u \to^* w$ and $v \to^* w$. $R$ is confluent, as rules are orthogonal (left-linear, non-overlapping). Arithmetic rules (commutative/associative/distributive/identity) mirror polynomial rewriting to sum-of-products form, confluent with multisets for commutative operands [39]. Trigonometric/exponential rules do not overlap with arithmetic rules, preserving confluence via modularity [40]. By Newman's Lemma, termination and local confluence imply global confluence, so $R$ has a unique $nf(t)$.
- **EG Encodes Uniquely**: Builds minimal DAG from $nf(t)$ (Appendix C.2): flattens chains (multi-operand nodes with multisets), expands distributivity, simplifies identities/trigonometric. Fixed variables ensure uniqueness up to isomorphism; node sharing for minimality, multisets for order-independence.

Consequently, $t_1 \sim_R t_2 \implies nf(t_1) = nf(t_2) \implies EG(t_1) \cong EG(t_2)$. It also holds for nested cases (e.g., $(a \times (b + c)) + d$) via recursive normalization.

### C.3 R-GCN Message Passing Detail

We adopt the Relational Graph Convolutional Network (R-GCN) [20] for GSR, and modify it with the residual design for stable training with the following message passing equation for each R-GCN unit as shown in Fig. 2:

$$h'_i = h_i + \sigma(\sum_{\alpha} \sum_{j} c_{ij} W_\alpha h_j + W_0 h_i), \tag{5}$$

where node $j$ is the neighbor of node $i$, $N^\alpha(i)$ represents all neighbors of $i$ with the corresponding relation $\alpha$, $W_0$ and $W_\alpha$ are self-loop weight and relation weight respectively, $c_{ij}$ denotes the normalizer for $W_\alpha$, and $\sigma$ is the softplus activation function. To output the prior distribution $\mathbf{P}$, we use a multilayer perceptron (MLP) with softmax activations on the node messages, while a separate MLP with the softplus activation is employed to produce the global readout on node messages for predicted rewards.

The relation weight $W_\alpha$ is edge-feature dependent, which means that for different edge types, R-GCN will go through distinct message passing updates, leading to unique expression representation for different operands' relations and more accurate and expressive encoding for expressions.

### C.4 Hybrid neural-guided MCTS (hnMCTS)

**Algorithm 1** summarizes the pseudo-code for our GSR. The core of the algorithm is the repeated sampling of expression trajectories $\tau$ until either the desired solution is found or the maximum number of iterations is reached. At each trajectory step $t$, a batch of hnMCTS with index $k$ and a maximum batch size $B$ is executed, following the four steps for hnMCTS iterations in **Algorithm 2**.

**Algorithm 1** Graph-based Symbolic Regression

**Input:** operator dictionary $\Phi$, Batch size $B$, maximum complexity $H$, $\epsilon$-UCB policy threshold $c_\epsilon$
**repeat**
  **for** $t = 1$ **to** $H$ **do**
    **for** $k = 1$ **to** $B$ **do**
      Update $Q(s,a), N(s,a)$ from `hnMCTS`$(s_t, Q(s,a), N(s,a), \Phi, H, c_\epsilon)$
    **end for**
    Update $\pi_M^t(s_t) = N(s_t, \cdot)/\sum_b N(s_t, b)$
    Sample $a_t \sim \pi_M^t(s_t)$, $a_t \in \Phi$ , grow $s_t$ with $a_t$ to $s_{t+1}$
    **if** $s_{t+1}$ is complete **then**
      **break**
    **end if**
  **end for**
  Calculate $R(\tau)$, record $\pi_\theta(s)$ and $\pi_M(s)$
  Calculate the loss $l = \sum_t ((R(\tau) - \pi_\theta|_r(s_t))^2 - \pi_M(s_t) \log \pi_\theta|_{\mathbf{P}}(s_t))$
  Update $\pi_\theta$ according to $l$
**until** Optimal $f$ is found
**Output:** solution expression $f$

**Algorithm 2** hnMCTS

**Input:** root state $s_t$, value function $Q(s,a)$, visit count $N(s,a)$, operator dictionary $\Phi$, maximum complexity $H$, $\epsilon$-UCB policy threshold $c_\epsilon$
Sample $\epsilon \in [0,1)$
**for** $i = 0$ **to** $H - t - 1$ **do**
  **if** $\epsilon < c_\epsilon$ **then**
    Select $a^i_{t+i} \sim \arg\max_a \text{UCB}(s^i_{t+i}, a^i_{t+i})$
  **else**
    Select $a^i_{t+i} \sim \arg\max_a \text{PUCB}(s^i_{t+i}, a^i_{t+i})$
  **end if**
  Grow $s^i_{t+i}$ with $a^i_{t+i}$, step to $s^{i+1}_{t+i+1}$, and update $N(s^i_{t+i}, a^i_{t+i}) \leftarrow N(s^i_{t+i}, a^i_{t+i}) + 1$
  **if** $s^{i+1}_{t+i+1}$ has unvisited children **then**
    **break**
  **end if**
**end for**
$i \leftarrow i + 1$
**if** $s^i_{t+i}$ is expandable **then**
  **if** $\epsilon < c_\epsilon$ **then**
    **repeat**
      Grow $s^i_{t+i}$ with a random $a^i_{t+i}$, step to $s^{i+1}_{t+i+1}$ and update $i \leftarrow i + 1$
    **until** $s^i_{t+i}$ is complete
    Evaluate $R(s^i_{t+i})$ with constants fitting and record $P(s_t, \cdot) = 1/|\Phi|$
  **else**
    $R(s_t) = \pi_\theta|_r(s)$, $P(s_t, \cdot) = \pi_\theta|_{\mathbf{P}}(s)$
  **end if**
**else**
  Evaluate $R(s^i_{t+i})$ with constants fitting
**end if**
Backpropagate $Q(s,a) = (1/N(s,a)) \sum_{s'|s,a} R(s')$ for all $(s,a)$ pairs traversed through
**Output:** value function $Q(s,a)$, visit count $N(s,a)$

### C.5 Proof of Convergence and Exploration-Exploitation Trade-Off

This appendix provides detailed proofs for the convergence (Theorem 3.2) and exploration-exploitation trade-off of the hybrid neural-guided Monte Carlo Tree Search (hnMCTS) with an $\epsilon$-UCB policy, as outlined in the main text. We first prove convergence for the edge cases ($c_\epsilon = 1$, $c_\epsilon = 0$) and the general case ($0 < c_\epsilon < 1$), showing that hnMCTS achieves a better convergence rate than pure neural-guided MCTS. We then analyze the exploration-exploitation trade-off, highlighting how the $\epsilon$-UCB policy balances these objectives.

#### C.5.1 Convergence

We consider a state $s$ with a finite action space $\mathcal{A}$, where $|\mathcal{A}| = K$. Rewards are bounded, $R(s,a) \in [0,1)$, and the optimal action is $a^* = \arg\max_{a \in \mathcal{A}} Q^*(s,a)$, with $Q^*(s,a)$ as the true action value. The hnMCTS policy is defined as:

$$\pi^T_M(s,a) = \frac{N_T(s,a)}{\sum_{b \in \mathcal{A}} N_T(s,b)},$$

where $N_T(s,a)$ is the visit count for action $a$ after $T$ iterations. Convergence requires:

$$\lim_{T \to \infty} \pi^T_M(s,a) = \pi^*_M(s,a) = \begin{cases} 1 & \text{if } a = a^*, \\ 0 & \text{if } a \neq a^*, \end{cases}$$

assuming a unique optimal action for simplicity. For multiple optimal actions, the policy distributes probability equally among them.

The $\epsilon$-UCB policy selects vanilla MCTS with probability $c_\epsilon$ and neural-guided MCTS with probability $1 - c_\epsilon$, as defined in Equation (1) of the main text. Naive MCTS uses the UCB selection policy:

$$UCB(s,a) = Q_T(s,a) + C_U \sqrt{\frac{\ln \sum_b N_T(s,b)}{N_T(s,a)}},$$

where $Q_T(s,a) = \frac{1}{N_T(s,a)} \sum_{t=1}^{N_T(s,a)} R_t(s,a)$, and $C_U > 0$. Neural-guided MCTS uses the PUCB selection policy:

$$PUCB(s,a) = Q_T(s,a) + C_P P(s,a) \sqrt{\frac{\sum_b N_T(s,b)}{1 + N_T(s,a)}},$$

where $P(s,a)$ is the neural network's prior probability, $\sum_a P(s,a) = 1$, $P(s,a) \geq 0$, and $C_P > 0$.

**Case 1: Pure Naive MCTS ($c_\epsilon = 1$)**

**Theorem C.1.** *Given bounded rewards $R(s,a) \in (0,1]$, a finite action space $\mathcal{A}$, and $C_U > 0$, MCTS with the* $\arg\max_a UCB$ *selection policy ensures* $\lim_{T\to\infty} N_T(s,a) = \infty$ *for all* $a \in \mathcal{A}$.

*Proof.* Assume there exists an action $a \in \mathcal{A}$ such that $N_T(s,a) \leq M < \infty$ for all $T$. After some iteration $T_0$, $N_T(s,a) = m \leq M$ is fixed. The UCB value is:

$$UCB(s,a) = Q_T(s,a) + C_U \sqrt{\frac{\ln T}{m}}.$$

Since $Q_T(s,a) \in [0,1)$, and $T = \sum_b N_T(s,b)$, as $T \to \infty$, $\ln T \to \infty$, so:

$$UCB(s,a) \to \infty.$$

For another action $a' \neq a$ with $N_T(s,a') \to \infty$:

$$UCB(s,a') = Q_T(s,a') + C_U \sqrt{\frac{\ln T}{N_T(s,a')}}.$$

As $N_T(s,a') \to \infty$, $\sqrt{\frac{\ln T}{N_T(s,a')}} \to 0$, so $UCB(s,a') \to Q_T(s,a') \leq 1$. For large $T$, $UCB(s,a) > UCB(s,a')$, forcing selection of $a$, contradicting the assumption that $N_T(s,a) = m$. Consequently, $\lim_{T\to\infty} N_T(s,a) = \infty$.

**Theorem C.2.** *Given the assumptions of Theorem C.1, unbiased random rollouts, and a unique optimal action $a^*$, we have $N_T(s,a) = O(\ln T)$ for $a \neq a^*$ and $N_T(s,a^*) \sim T$.*

*Proof.* For a suboptimal action $a \neq a^*$, define $\Delta_a = Q^*(s,a^*) - Q^*(s,a) > 0$. Action $a$ is selected when:

$$Q_T(s,a) + C_U \sqrt{\frac{\ln T}{N_T(s,a)}} > Q_T(s,a^*) + C_U \sqrt{\frac{\ln T}{N_T(s,a^*)}}.$$

As $T \to \infty$, $Q_T(s,a) \to Q^*(s,a)$, $Q_T(s,a^*) \to Q^*(s,a^*)$. The exploration term must overcome $\Delta_a$:

$$C_U \sqrt{\frac{\ln T}{N_T(s,a)}} \gtrsim \Delta_a.$$

Squaring gives:

$$N_T(s,a) \leq \frac{C_U^2 \ln T}{\Delta_a^2}.$$

From UCB analysis [41], $N_T(s,a) \leq \frac{8 \ln T}{\Delta_a^2}$ + constants, so $N_T(s,a) = O(\ln T)$. For $K-1$ suboptimal actions:

$$\sum_{a \neq a^*} N_T(s,a) = O((K-1) \ln T).$$

Since $T = \sum_a N_T(s,a)$:

$$N_T(s,a^*) = T - \sum_{a \neq a^*} N_T(s,a) = T - O(\ln T) \sim T.$$

*Remark* 1. Theorems C.1 and C.2 imply $\pi_M^T(s, a^*) \to 1$, $\pi_M^T(s, a) \to 0$ for $a \neq a^*$, ensuring convergence to the optimal policy.

**Case 2: Pure Neural-Guided MCTS ($c_\epsilon = 0$)**

**Theorem C.3.** *Given the assumptions of Theorem C.1 and $C_P > 0$, MCTS with the* $\arg\max_a PUCB$ *policy ensures* $\lim_{T\to\infty} N_T(s, a) = \infty$ *for all $a \in \mathcal{A}$ with $P(s, a) > 0$.*

*Proof.* Assume $N_T(s, a) \leq M < \infty$ for some $a$ with $P(s, a) > 0$. After $T_0$, $N_T(s, a) = m \leq M$. The PUCB value is:

$$PUCB(s, a) = Q_T(s, a) + C_P P(s, a)\sqrt{\frac{T}{1 + m}}.$$

As $T \to \infty$, $\sqrt{T} \to \infty$, so $PUCB(s, a) \to \infty$. For $a' \neq a$ with $N_T(s, a') \to \infty$:

$$PUCB(s, a') = Q_T(s, a') + C_P P(s, a')\sqrt{\frac{T}{1 + N_T(s, a')}}.$$

Since $\sqrt{\frac{T}{1+N_T(s,a')}} \to 0$, $PUCB(s, a') \to Q_T(s, a') \leq 1$. Then, $PUCB(s, a) > PUCB(s, a')$ for large $T$, forcing selection of $a$, contradicting $N_T(s, a) = m$. Hence, $\lim_{T\to\infty} N_T(s, a) = \infty$ if $P(s, a) > 0$.

**Theorem C.4.** *Given the assumptions of Theorem C.1, unbiased neural-guided simulations, a unique optimal action $a^*$, and $P(s, a^*) > 0$, we have $N_T(s, a) = O(\ln T)$ for $a \neq a^*$ and $N_T(s, a^*) \sim T$.*

*Proof.* For $a \neq a^*$, action $a$ is selected when:

$$Q_T(s, a) + C_P P(s, a)\sqrt{\frac{T}{1 + N_T(s, a)}} > Q_T(s, a^*) + C_P P(s, a^*)\sqrt{\frac{T}{1 + N_T(s, a^*)}}.$$

As $T \to \infty$, $Q_T(s, a) \to Q^*(s, a)$, $Q_T(s, a^*) \to Q^*(s, a^*)$, with $\Delta_a = Q^*(s, a^*) - Q^*(s, a) > 0$. Assuming $N_T(s, a^*) \gg N_T(s, a)$, the condition approximates to:

$$C_P P(s, a)\sqrt{\frac{T}{1 + N_T(s, a)}} \gtrsim \Delta_a.$$

Squaring:

$$N_T(s, a) \leq \frac{C_P^2 P(s, a)^2 T}{\Delta_a^2} - 1.$$

Adapting the convergence analysis of UCB [41], if $P(s, a) > 0$, $N_T(s, a) = O(\ln T)$. If $P(s, a) = 0$, $N_T(s, a)$ may be bounded. Aggregating suboptimal visits:

$$\sum_{a\neq a^*} N_T(s, a) = O((K - 1) \ln T).$$

Accordingly:

$$N_T(s, a^*) = T - O(\ln T) \sim T.$$

*Remark* 2. Theorems C.3 and C.4 require $P(s, a^*) > 0$. If $P(s, a^*) = 0$, convergence may fail due to under-exploration of $a^*$.

**Case 3: Hybrid MCTS ($0 < c_\epsilon < 1$)**

**Theorem C.5.** *Given the assumptions of Theorem C.1, unbiased simulations, and $0 < c_\epsilon < 1$, hnMCTS with the $\epsilon$-UCB policy ensures $\lim_{T\to\infty} N_T(s,a) = \infty$ for all $a \in \mathcal{A}$, $N_T(s,a) = O(\ln T)$ for $a \neq a^*$, and $N_T(s,a^*) \sim T$.*

*Proof.* The expected visit count is:

$$E[N_T(s,a)] = c_\epsilon N_T^{\text{UCB}}(s,a) + (1-c_\epsilon) N_T^{\text{PUCB}}(s,a).$$

By Theorem C.1, $N_T^{\text{UCB}}(s,a) \to \infty$ for all $a$. Hence, $E[N_T(s,a)] \to \infty$, even if $P(s,a) = 0$. For suboptimal actions, Theorems C.2 and C.4 give $N_T^{\text{UCB}}(s,a) = O(\ln T)$ and $N_T^{\text{PUCB}}(s,a) = O(\ln T)$, so:

$$E[N_T(s,a)] = O(\ln T) \text{ for } a \neq a^*.$$

For the optimal action:

$$E[N_T(s,a^*)] = T - \sum_{a\neq a^*} E[N_T(s,a)] \sim T.$$

Consequently, $\pi_M^T(s,a^*) \to 1$, $\pi_M^T(s,a) \to 0$ for $a \neq a^*$.

*Remark* 3. The condition $P(s,a^*) > 0$ is relaxed in hnMCTS due to the vanilla MCTS component ($c_\epsilon > 0$), improving convergence over pure neural-guided MCTS.

### C.5.2 Exploration-Exploitation Trade-Off

The $\epsilon$-UCB policy, selecting vanilla MCTS with probability $c_\epsilon$ and neural-guided MCTS with probability $1 - c_\epsilon$, balances exploration and exploitation:

—**Naive MCTS ($c_\epsilon$)**: The UCB exploration term $C_U\sqrt{\frac{\ln \sum_b N_T(s,b)}{N_T(s,a)}}$ promotes exploration when $N_T(s,a)$ is small, and the $Q_T(s,a)$ term promotes exploitation when $N_T(s,a)$ is large.

—**Neural-Guided MCTS ($1 - c_\epsilon$)**: The PUCB term $C_P P(s,a)\sqrt{\frac{\sum_b N_T(s,b)}{1+N_T(s,a)}}$ encourages exploration for low $N_T(s,a)$, moderated by $P(s,a)$, while $Q_T(s,a)$ drives exploitation.

The parameter $c_\epsilon$ controls the trade-off:
—High $c_\epsilon$: Emphasizes vanilla MCTS, favoring exploration via uniform sampling and random rollouts.
—Low $c_\epsilon$: Emphasizes neural-guided MCTS, enhancing exploitation via the neural prior $P(s,a)$.

The regret of hnMCTS:

$$R_T \leq c_\epsilon O(\sqrt{KT\ln T}) + (1-c_\epsilon) O(\sqrt{T}),$$

where vanilla MCTS contributes $O(\sqrt{KT\ln T})$ and neural-guided MCTS contributes $O(\sqrt{T})$ (assuming accurate priors). Tuning $c_\epsilon$ optimizes the trade-off, with $c_\epsilon > 0$ ensuring robust exploration to correct misleading priors, enhancing convergence reliability over pure neural-guided MCTS.

## C.6 Hyperparameter Setting for GSR

Table 4 describes the hyperparameter settings of GSR for our reported experiments in Sections 4, 5, and Appendix D. In these experiments, $c_\epsilon$ not only promotes better convergence and exploration-exploitation trade-off described in Appendix C.5.1, but also facilitates a smooth start-up in R-GCN training. Initially, GSR lacks sufficient information about the constraints, making random roll-outs more effective. Therefore, for the first 10% of the training epochs, we set $c_\epsilon = 1$ to encourage exploration. As training progresses and GSR becomes more informed, $c_\epsilon$ is gradually reduced to 0.1, allowing R-GCN to guide more simulations and prioritize exploitation. Experimental results of the $c_\epsilon$ tuning are detailed in Appendix D.4. To promote better complexity control, we also adopt a dynamic complexity upper bound strategy. Given the maximum complexity upper bound $H^*$, the actual dynamic complexity upper bound $H$ in Algorithms 1 and 2 is initialized as the half of $H^*$ for the first 20% of the total training epochs, then linearly increased to $H^*$ by the 60% of the training epoch mark, leading to more explorations in lower complexity region.

| Notation | Value | Explanation |
|---|---|---|
| $c_\epsilon$ | $1 \to 0.1$ | The switching threshold for the $\epsilon$-UCB policy in (3), initialized to be 1 for the first 10% training epochs, then linearly decreased to 0.1 by the 80% of the training epoch mark. |
| $C_U$ | $\sqrt{2}$ | The exploration coefficient for UCB in (3) |
| $C_P$ | 1 | The exploration coefficient for PUCB in (3) |
| $c_{ij}$ | $\lvert\frac{1}{\Phi}\rvert$ | The normalizer for the relation weight $W_\alpha$ in (5). $\lvert\Phi\rvert$ denotes the total number of operations in the dictionary |
| $H$ | $0.5H^* \to H^*$ | Dynamic complexity upper bound of expressions in Algorithms 1 and 2, initialized to be half of the maximum complexity $H^*$ for the first 20% training epochs, then linearly increased to $H^*$ by the 60% of the training epoch mark. |
| $B$ | - | The batch size for hnMCTS in Algorithm 1, dependent on the dataset. |
| $n$ | 20 | The budget size to store expressions for the transplantation. |

Table 4: Hyperparameter settings for GSR

### C.7 Evaluation Metrics for hnMCTS

During the training, we evaluate $\tau$ through the reward $R(\tau) = 1/(1 + \text{NMAE})$, where NMAE is the normalized mean absolute error defined as:

$$\text{NMAE} = \frac{1}{\sigma_y}\frac{1}{n}\sum_{i=1}^{n}|y_i - f_\tau(X_i)|. \tag{6}$$

## D Additional Experimental Detail

### D.1 Experimental Setting

**Black-box and Feynman dataset:** The Black-box and the Feynman datasets are widely accepted datasets adopted by the state-of-the-art SR benchmarking, SRBench [27]. The Feynman dataset is a synthetic dataset consisting of 119 physics-inspired solution equations derived from the Feynman Lectures on Physics, and the Black-box dataset is a challenging dataset consisting of real-world observation and simulation data without explicit solution expressions [42].

For both the Black-box and Feynman datasets, each of the equations is run ten times with different random seeds. Each of the benchmarking methods is restricted to one million evaluations, a run-time budget of 10 hours, and a maximum complexity of 200 for the expressions. We follow the same dictionary set (available operators and variables) and other default experimental settings in SRBench, where the train-test split ratio is $3:1$, and the default operation set includes $\Phi = \{+, -, \times, \div, \exp, \log, \sin, \cos, x_i\}$. Here $x_i$ denotes the variables in the dataset. Baselines included in SRBench adopt the same hyperparameter settings reported in [27]. GSR adopts a hnMCTS batch size $B = 100$ and other hyperparameters described in Appendix C.6. For the added baselines, DGSR [6] and uDSR [9] utilized the reported hyperparameter setting for the corresponding datasets. TPSR [10] follows the reported $\lambda = 0$ model configurations. SPL [8] for Nguyen benchmarking utilized the maximum Module Transplantation as 20, episodes between module transplantations as 10,000, the regularization factor $\eta$ as 0.99, and the exploration rate $c$ as $\sqrt{2}$.

**Nguyen dataset:** Our proposed GSR method is also evaluated in Nguyen's SR benchmark dataset [26] and Nguyen's SR benchmark with constants dataset [7], two widely adopted synthetic datasets for evaluating the performance and robustness of various SR algorithms.

The "Expression" column in Table 5 includes the ground-truth expressions used to generate synthetic data. In this set of experiments, we adopt results of GP, NGGP and SPL from [8] and follow the same

settings to implement GSR, which uses a dictionary set $\Phi_0 = \{\times, +, -, \div, sin, cos, exp, log, x\}$ for the benchmark Ngugen-1 to Ngugen-8, and $\Phi_1 = \Phi_0 \cup \{y\}$ for the benchmark Ngugen-9 to Ngugen-12. It is worth noticing that for the benchmark Nguyen-8, $\sqrt{x}$ can be recovered from $\exp\left(\frac{x}{x+x}\log\left(x\right)\right)$. For Nguyen-10, $x^y$ can be recovered from $\exp\left(y\log\left(x\right)\right)$. For Nguyen-7 and Nguyen-10, they can also be recovered from $\log\left(x^3+x^2+x+1\right)$ and $\sin\left(x+y\right)$. The maximum complexity $H^*$ for each expression is set to 50. The batch size $B$ is set to 100 for MCTS simulations, and the maximum number of epochs is capped at 1,000, with a total search space of up to one million expressions for MCTS and GSR. The training and testing datasets are divided equally, with 20 randomly generated data points for each.

### D.2 Nguyen's Benchmark Result

Table 5 shows that GSR outperforms competing SR approaches in almost all tasks, particularly in all complex tasks, including Nguyen-12. This advantage is primarily due to GSR's effective management of polynomial terms, facilitated by the equivalence captured by EG, which compresses the search space more efficiently than the expression tree or other representations utilized by alternative methods, highlighting its considerable potential in the domain of Symbolic Regression.

| Benchmark | Expression | GP | NGGP | SPL | GSR |
|---|---|---|---|---|---|
| Nguyen-1 | $x^3+x^2+x$ | 99% | 100% | 100% | 100% |
| Nguyen-2 | $x^4+x^3+x^2+x$ | 90% | 100% | 100% | 100% |
| Nguyen-3 | $x^5+x^4+x^3+x^2+x$ | 34% | 100% | 100% | 100% |
| Nguyen-4 | $x^6+x^5+x^4+x^3+x^2+x$ | 54% | 100% | 99% | 100% |
| Nguyen-5 | $\sin\left(x^2\right)\cos\left(x\right)-1$ | 12% | 80% | 95% | 100% |
| Nguyen-6 | $\sin\left(x\right)+\sin\left(x+x^2\right)$ | 11% | 100% | 100% | 100% |
| Nguyen-7 | $\log\left(x+1\right)+\log\left(x^2+1\right)$ | 17% | 100% | 100% | 96% |
| Nguyen-8 | $\sqrt{x}$ | 100% | 100% | 100% | 100% |
| Nguyen-9 | $\sin\left(x\right)+\sin\left(y^2\right)$ | 76% | 100% | 100% | 100% |
| Nguyen-10 | $2\sin\left(x\right)\cos\left(y\right)$ | 86% | 100% | 100% | 100% |
| Nguyen-11 | $x^y$ | 13% | 100% | 100% | 100% |
| Nguyen-12 | $x^4-x^3+\frac{1}{2}y^2-y$ | 0% | 4% | 28% | 88% |
| Nguyen-$1^c$ | $3.39x^3+2.12x^2+1.78x$ | 0% | 100% | 100% | 100% |
| Nguyen-$2^c$ | $0.48x^4+3.39x^3+2.12x^2+1.78x$ | 0% | 100% | 94% | 100% |
| Nguyen-$5^c$ | $\sin\left(x^2\right)\cos\left(x\right)-0.75$ | 1% | 98% | 95% | 100% |
| Nguyen-$8^c$ | $\sqrt{1.23x}$ | 56% | 100% | 100% | 100% |
| Nguyen-$9^c$ | $\sin\left(1.5x\right)+\sin\left(0.5y^2\right)$ | 0% | 90% | 96% | 100% |
| | Average Solution Rate | 38.2% | 92.5% | 94.5% | **99.1**% |

Table 5: Solution Rate of GSR and other baseline models benchmarked on Nguyen's SR benchmark dataset and Nguyen's SR benchmark with constants dataset (marked with upper index $c$). The solution rate is the ratio of ground-truth equivalent solutions in mathematics to the total of parallel experiments for the same equation.

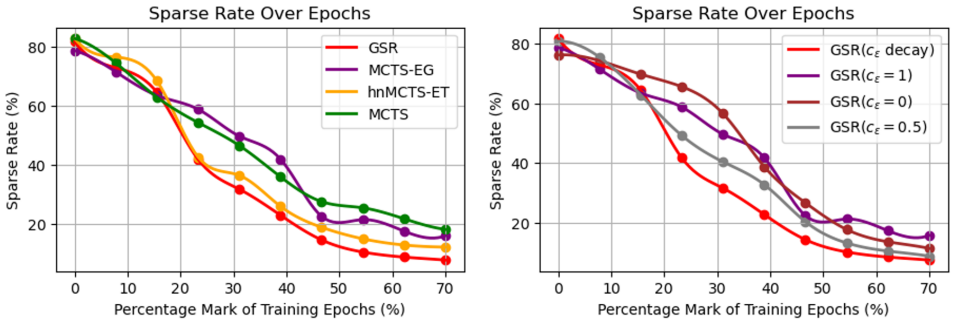
### D.3 Additional Scalability and Efficiency Analysis

EGs are DAGs with node sharing, resulting in fewer nodes than equivalent ETs (Fig. 2), which helps reduce memory and computational overhead. Moreover, the hnMCTS design does not rely solely on neural-guided search, where the UCB component ensures efficiency during exploration. Together, these design choices contribute to the method's practical efficiency as shown in the ablation study (Section 4.2), where GSR achieves strong performance while requiring the least training time among competitive baselines.

Regarding scalability, GSR performs robustly across datasets with varying input dimensionality. For example, in the black-box benchmark (Fig. 4(c)), the average training time remains manageable across increasing dimensions, as shown in Table 6.

| **Dataset Dimension** | 3 | 8 | 13 | 18 | 36 |
|---|---|---|---|---|---|
| **Average Training Time** (s) | 12676 | 31483 | 20781 | 12783 | 9172 |

Table 6: Average training time of GSR in the black-box dataset across different dimensions




(a) (b)

Figure 6: Sparse rate over epochs in the (a) ablation study and (b) $c_\epsilon$ tuning. Sparse rate is calculated based on the average ratio between the number of invalid expressions sampled per epoch and the total number of expressions sampled per epoch.

### D.4 Additional Ablation Analysis

**Ablation Sparsity Analysis:** Figure 6(a) illustrates the average reward sparsity across different ablation models in the ablation study in Section 4.2. During the initial 10% epochs, all models exhibit similar trends due to the random rollout induced by the warm-up $\epsilon$-UCB policy in hnMCTS. However, beyond this phase, models incorporating hnMCTS with constraints encoding demonstrate a significant reduction in reward sparsity, and the models incorporating EG with invariance encoding expedite the decrease of the sparse rate. This highlights the effectiveness of hnMCTS in capturing constraints encoding in integrating priors of expression-level constraints, along with EG's complementary effects of invariance encoding for faster convergence, ultimately enhancing search efficiency.

**Sensitivity Analysis on $c_\epsilon$:** Figure 6(b) illustrates the average reward sparsity across different hyperparameter tuning strategies for $c_\epsilon$ of GSR, based on the same experimental settings as Fig. 6(a). When $c_\epsilon$ is fixed to be 1, hnMCTS acts like vanilla MCTS, so GSR($c_\epsilon = 1$) in Fig. 6(b) is equivalent to MCTS-EG in Fig. 6(a). When $c_\epsilon$ is fixed to be 0, hnMCTS will fully rely on R-GCN guidance. When $c_\epsilon$ is fixed to be 0.5, hnMCTS will switch between vanilla MCTS and neural-guided MCTS by equal chance. GSR($c_\epsilon$ decay) represents the same model as GSR in Fig. 6(a), which adopts a decaying $c_\epsilon$ as described in Appendix C.6. The results of the sparse rate in Fig. 6(b) demonstrate that pure neural-guided MCTS ($c_\epsilon = 0$) will result in slow convergence at the beginning due to the sparse rewards of SR and lack of training of R-GCN, leading to the biased $Q(s, a)$ estimation. But the vanilla MCTS ($c_\epsilon = 1$) also has the highest sparse rate after the long training due to the lack of constraint prior encoding to promote promising candidates. Consequently, compared to another hybrid MCTS with $c_\epsilon$=0.5, hnMCTS adopting the decaying $c_\epsilon$ achieves a better balance between the early exploration and latter exploitation, achieving a faster convergence and higher efficiency.

## E Materials Science Application

### E.1 Problem Background

The physical world is composed of numerous ions and electrons, governed by quantum mechanics, often referred to as many-body problems. While density functional theory (DFT) can simulate simple materials with few ions and electrons, extending DFT to larger systems is challenging. One of the goals relevant to the physical properties of materials is to find an analytical approximation that relates DFT-calculated potential energy to the interatomic distance (i.e., pairwise distances) between atoms, adhering to the strong electrostatic repulsion at short interatomic distances and weaker interaction at long interatomic distances. Using copper (Cu) crystals as an example, we aim to identify analytical relationships between interatomic distances and total potential energy. This functional form is crucial for materials science as it enables large-scale molecular dynamics simulations to study materials'

mechanical, thermal, and kinetic properties and explain fundamental physical mechanisms at the atomic level.

The following provides some explanations for domain-specific terms:

- Molecular dynamics (MD) simulates the structure and properties of materials under constant or varying environments (e.g., temperature or mechanical strain). In MD simulations, the atomic forces are computed from the derivatives of interatomic potential energy functions with respect to the atomic distance placement, which then moves the atoms based on Newton's second law of motion. The updated atomic positions lead to new interatomic potential energy. The calculations are performed iteratively until a desired number of time steps. Statistics of total potential energy, kinetic energy, atomic forces, and stresses, etc., provide a systematic understanding and quantitative estimate of the physical properties of materials. MD simulations are often performed for systems with hundreds to millions of atoms, making them computationally expensive. This necessitates machine learning algorithms like those employed in our study. Analytical expressions with low complexity and high accuracy are particularly valuable for large-scale MD simulations.
- In MD simulations, the system state can be specified using ensembles like NVT (constant number of atoms, volume, and temperature), allowing other properties such as pressure and chemical potential to vary.
- Atomistic simulations based on machine learning-derived potential energy functions use atomic species and interatomic forces to determine material properties. This method is efficient but requires accurate fitting from first-principles DFT or other quantum mechanics-based calculations.
- Periodic boundary conditions (PBCs) enable the modeling of infinite or effectively infinite systems by repeating a "unit cell" in all directions. This unit cell represents the smallest section of the structure that can be repeated to create the correct crystal structure, allowing large systems to be modeled efficiently without losing structural or symmetrical integrity.

### E.2 Pratical Usage of GSR's Solution

- **Framework Generality**: GSR is not limited by system size or atom types. The 32-atom copper dataset is chosen to serve as a benchmark to compare GSR with prior work and demonstrate its advantages. The derived potential energy function can be applied to larger systems.
- **Dataset Relevance**: The 32-atom copper dataset is representative of materials studies, consistent with the average system size ($\sim$ 31 atoms/structure) of the widely used MPtraj dataset [43].
- **Scalability**: While DFT methods are limited to simulating small systems (1–1000 atoms) over very short timescales (a few picoseconds), our method enables large-scale simulations (millions of atoms) over extended timescales (microseconds)- which can then help determine several physical properties of materials that are not pratical for quantum chemistry methods such as DFT. The analytical function from GSR can be applied to study real material problems with sizes far beyond 32 atoms for long-time dynamics that are completely inaccessible to DFT or other quantum chemistry methods – which is the key purpose of developing a force field based on accurate analytical potential energy functions from the GSR method. As a result, one can simulate the mechanical deformation process of single or polycrystalline copper consisting of millions of copper atoms using large-scale molecular dynamics simulations with the potential energy function developed here. For example, studying the novel stacking of copper in the incubation period of crystallization [44] requires the simulation of millions of copper atoms.
- **DFT-Level Accuracy**: The dataset is derived from DFT calculations, enabling DFT-level accuracy but without explicit electron degree of freedom - which is the major purpose of developing the potential energy function (i.e., machine learning force field) from quantum chemistry datasets such as DFT. The high accuracy, the analytical nature, and the proper limiting trend at short bond length make this method and the derived potential energy function especially useful in practice.

### E.3 Physics Constraints and Explanations

We offer detailed explanations and examples for some physics constraints defined in Section 5.2:

**(1) Scalar Output** *(operation-level constraint)*: Because the input variable $\boldsymbol{r}$ is a vector of variable length for different samples, a $\sum$ operator must be included in the expression before any $\boldsymbol{r}$.

**(2) Unit Consistency** *(operation-level constraint)*: It is defined to ensure that the unit calculation aligns with physical meaning, as $const$ can introduce an extra unit as a coefficient. For a specific example of the unit consistency constraint, the expression of $(r^2 + r)$ is not physically meaningful, as $r$ has a length unit Å. and we would have an inconsistent unit (Å$^2$ +Å) from the expression. But a constant $c$ can include unit Å so that we have a consistent calculation with $r^2 + c * r$. The introduced $const$ can ensure that the final output unit aligns with the target $E$'s unit as electron-volt ($eV$; $1\ eV = 1.6 \times 10^{-19}$ Joule) from the input $\boldsymbol{r}$'s unit Angstrom (Å; 1Å= $10^{-10}m$).

**(3) Electrostatic Repulsion** *(expression-level constraint)*: It's a subtle consequence of Pauli's exclusion principle of quantum mechanics: When the distance between two atoms approaches zero, their electron wavefunctions start to overlap significantly, which is excluded by Pauli's principle. As a result, the atomic orbitals hybridize and form molecular orbitals with bonding and antibonding characters, and electron density is hence pushed away from nuclei, leaving the repulsive nucleus-nucleus electrostatic interaction being the dominant one and approaching infinite potential energy at very short distances.

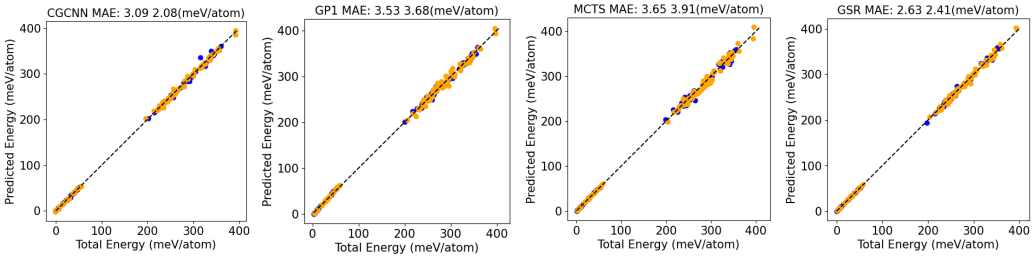
### E.4 Application Experimental Setting

**Dataset Description:** The 32-atom copper dataset consists of 150 snapshots generated by the Vienna Ab initio Simulation Package (VASP), a first-principles DFT package. Each sample includes total formation energy $E$ as the target and a crystal structure of copper as the variable. To map a copper crystal structure to $E$, we first convert the 3D coordinates of the structure into pairwise distances $\boldsymbol{r}$ between atoms within the unit cell in the crystal under the periodic boundary condition (PBC) [45], then use the surrogate model as the interatomic potential energy function $f(\cdot)$ to obtain $E = f(\boldsymbol{r})$. During the conversion, we only consider pairwise distances within a cutoff range of $r < r_{\text{cutoff}} = 5$Å.

**SR Settings:** In the symbolic regression configurations, we utilize a dictionary set $\Phi = \{+, -, \times, \div, \wedge, \sum, exp, \text{const}, \boldsymbol{r}\}$, where "$\sum$" is used for the summation-based aggregation operation, "const" denotes constants optimized through non-linear regression. The maximum complexity $H$ is set at 35, and the batch size $B$ is 1,000, with 5,000 episodes allocated for both MCTS and GSR methods. For the black-box CGCNN method, we adhere to the default hyperparameter settings with a maximum of 5,000 epochs. For the train-test split, we follow [16] with 50% for training and 50% for validation.

### E.5 Application Result Figure

Figure 7 compares the data fitting plots by different SR solution models in Table 2, corresponding to the performance discussions in Section 5.3.




(a) (b) (c) (d)

Figure 7: Training (orange dots) and Testing (blue dots) MAEs of the formation energy predictions by (a) CGCNN, (b) GP1, (c) MCTS, and (d) GSR on DFT-based molecular dynamics simulations of FCC copper are provided in the captions of the plots on the top, including both testing MAE (left) and training MAE (right). The dashed lines mark the identity mapping.

### E.6 Physical Meaning

Figure 5 presents the potential energy curves as a function of the interatomic distance (bond length) for the expressions generated by the GP1, GP2, MCTS, and GSR in Table 2. These curves demonstrate

whether the fitted models satisfy electrostatic repulsion at zero bond length. Figure 5 shows that GP1 and GP2 violate the constraint due to the finite value at zero bond length ($r$=0). In contrast, GSR and MCTS yield infinite potential energy at $r$=0, satisfying the principle. Such a constraint is crucial not only for the underlying physics but also for practical applications in materials system simulations. The missing divergence at $r = 0$ may cause atoms to collapse to each other during molecular dynamics simulations and yield incorrect results.

Additionally, Fig. 5 reveals two critical issues with GP1 and GP2: (i) a discontinuity at $r$=1Å, and (ii) an infinite value when $r$ approaches 5Å. These issues can lead to incorrect energy and force predictions, forcing atoms to remain unrealistically close together.

### E.7 Generation of the New Dataset for Testing Model Transferability

To test the transferability of the models, we generated a test dataset from DFT calculations using the VASP package [31] where the projector augmented wave method was applied to treat core electrons [46]. We used the Perdew-Burke-Ernzerhof (PBE) form of exchange-correlation energy functional within the generalized gradient approximation [47] and a Monkhorst-Pack $k$-point sampling grid of $4 \times 4 \times 4$ [48]. A hydrostatic compression was applied to FCC copper with strain of -0.2 along all lattice vectors. We then performed first-principles molecular dynamics simulations in the NVT ensemble for total 6,000 steps with a time step of 3 fs at two different temperatures, *i.e.* 300 K and 1,400 K. For the NVT calculation at each temperature, we excluded the first 1,000 steps of the initial equilibration process, and extracted the atomic structures for every 100 steps from the remaining equilibrated 5,000 steps, which yields total 100 snapshots (or samples) to test model transferability.

The compressed dataset has the following physical meaning:

- **Scientific Context**: Understanding matter in extreme conditions such as high pressure and high temperature is an important and active subject of materials research. Copper is one of the systems of particular interest. As shown in [49], the density in experiments reaches $\sim$18 g/cm$^3$, double the density at the standard condition of 8.95 g/cm$^3$, that is, the volume contraction by 50%. Another example is done by [50] at the National Ignition Facility (NIF) at the U.S. Lawrence Livermore National Laboratory (LLNL), in which the solid copper was even compressed to $\sim$28 g/cm$^3$ at terapascal conditions, corresponding to $\sim$67% volume contraction.
- **Practical Necessity:** Another key motivation to apply large compression is to provide a more accurate trend away from equilibrium towards $r \to 0$. This is particularly important as the machine learning interatomic potentials or machine learning force fields very often have wrong limiting behavior. When they were applied to simulating long-time dynamics at high temperature or high pressure, there will be an increasing probability of "direct crossing or fusion" of atoms which are pure artifacts due to the wrong limiting trend, consequently, the results can be completely non-physical and wrong.