Symbolic Regression via Control Variable Genetic Programming

Nan Jiang[0000−0001−6863−2897]✉ and Yexiang Xue[0000−0002−4533−0543]

Department of Computer Science, Purdue University, West Lafayette IN, US
{jiang631, yexiang}@purdue.edu

Abstract. Learning symbolic expressions directly from experiment data is a vital step in AI-driven scientific discovery. Nevertheless, state-of-the-art approaches are limited to learning simple expressions. Regressing expressions involving many independent variables still remain out of reach. Motivated by the control variable experiments widely utilized in science, we propose **C**ontrol **V**ariable **G**enetic **P**rogramming (CVGP) for symbolic regression over many independent variables. CVGP expedites symbolic expression discovery via customized experiment design, rather than learning from a fixed dataset collected a priori. CVGP starts by fitting simple expressions involving a small set of independent variables using genetic programming, under controlled experiments where other variables are held as constants. It then extends expressions learned in previous generations by adding new independent variables, using new control variable experiments in which these variables are allowed to vary. Theoretically, we show CVGP as an incremental building approach can yield an exponential reduction in the search space when learning a class of expressions. Experimentally, CVGP outperforms several baselines in learning symbolic expressions involving multiple independent variables.

Keywords: Control Variable Experiment · Symbolic Regression

1 Introduction

Discovering scientific laws automatically from experiment data has been a grand goal of Artificial Intelligence (AI). Its success will greatly accelerate the pace of scientific discovery. Symbolic regression, *i.e.*, learning symbolic expressions from data, consists of a vital step in realizing this grand goal. Recently, exciting progress [52, 57, 20, 45, 43, 45, 51, 48, 22] has been made in this domain, especially with the aid of deep neural networks. Despite great achievements, state-of-the-art approaches are limited to learning relatively simple expressions, often involving a few independent variables. Regressing symbolic expressions involving multiple independent variables still remains out of reach of current approaches. The difficulty mainly lies in the exponentially large search space of symbolic expressions.

Our work attacks this major gap of symbolic regression, leveraging control variable experimentation – a classic procedure widely implemented in the science community [38, 50]. In the analysis of complex scientific phenomena involving

many contributing factors, control variable experiments are conducted where a set of factors are held constant (*i.e.*, controlled variables), and the dependence between the output variable and the remaining input variables is studied [34, 27]. The result is a reduced-form expression that models the relationship only between the output and the non-controlled variables. Once the reduced-form equation is validated, scientists introduce more variables into play by freeing a few controlled variables in previous experiments. The new goal is to extend the previous equation to a general one including the newly introduced variables. This process continues until all independent variables are introduced.

Our proposed **C**ontrol **V**ariable **G**enetic **P**rogramming (CVGP) approach implements the aforementioned scientific discovery process using Genetic Programming (GP) for symbolic regression over many independent variables. The key insight of CVGP is to learn from *a customized set of control variable experiments*; in other words, the experiment data collection adapts to the learning process. This is in contrast to the current learning paradigm of most symbolic regression approaches, where they learn from a fixed dataset collected a priori. In CVGP, first, we hold all independent variables except for one as constants and learn an expression that maps the single variable to the dependent variable using GP. GP maintains a pool of candidate expressions and improves the fitness of these equations via mating, mutating, and selection over several generations. Mapping the dependence of one independent variable is easy. Hence GP can usually recover the ground-truth reduced-form equation. Then, CVGP frees one independent variable at a time. In each iteration, GP is used to modify the equations learned in previous generations to incorporate the new independent variable, via mating, mutating, and selection. Such a procedure repeats until all the independent variables have been incorporated into the symbolic expression.

After discovering CVGP independently, the authors learned in private communications a line of research work [31, 32, 36, 29, 28, 9] that also implemented the human scientific discovery process using AI, pioneered by the BACON systems developed by Langley, P. in 1978-1981 [31, 32, 36]. While BACON's discovery was driven by rule-based engines and our CVGP uses modern machine-learning approaches such as genetic programming. Indeed, both approaches share a common vision – the *integration of experiment design and model learning* can further expedite scientific discovery.

Theoretically, we show CVGP as an incremental builder can reduce the exponential-sized search space for candidate expressions into a polynomial one when fitting a class of symbolic expressions. Experimentally, we show CVGP outperforms a number of state-of-the-art approaches on symbolic regression over multiple independent variables. Our contributions can be summarized as:

1. We propose CVGP, an incremental builder for symbolic regression over many independent variables. CVGP fits increasingly more complex equations via conducting control variable experiments with fewer and fewer controlled variables.
2. Theoretically, we show such an incremental builder as CVGP can reduce exponential-sized search spaces for symbolic regression to polynomial ones when searching for a class of symbolic expressions.

3. Empirically, we demonstrate CVGP outperforms state-of-the-art symbolic regression approaches in discovering multi-variable equations from data[1].

2 Preliminaries

Symbolic Expression. A symbolic expression ϕ is expressed as variables and constants connected by a set of operators. Variables are allowed to vary while constants remain the same. Each operand of an operator is either a variable, a constant, or a self-contained symbolic expression. A symbolic expression can also be drawn as a tree, where variables and constants reside in leaves, and operators reside in inner nodes. See Fig. 1(a) for an example. In this paper, we deal with expressions involving real numbers. The semantic meaning of a symbolic expression follows its standard definition in arithmetics.

Symbolic Regression. Given a dataset $\{(\mathbf{x}_i, y_i)\}_{i=1}^n$ and a loss function $\ell(\cdot, \cdot)$, where $\mathbf{x}_i \in \mathbb{R}^m$ and $y_i \in \mathbb{R}$, the objective of symbolic regression (SR) is to search for the optimal symbolic expression ϕ^* within the space of all candidate expressions Π that minimizes the average loss:

$$\phi^* = \arg\min_{\phi \in \Pi} \frac{1}{n} \sum_{i=1}^{n} \ell(\phi(\mathbf{x}_i), y_i), \tag{1}$$

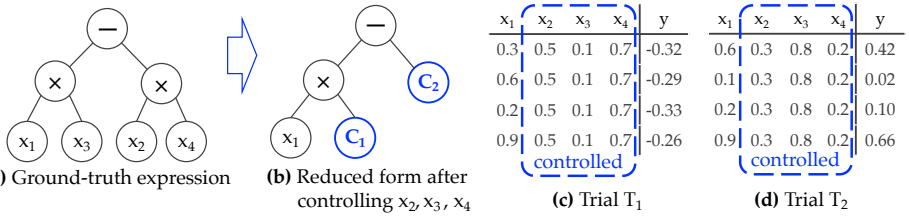
in addition to regularizers. Symbolic regression is challenging and is in NP-hard [58], due to the exponentially large space of candidate symbolic expressions.

Genetic Programming for Symbolic Regression. Genetic Programming (GP) has been a popular method to solve symbolic regression. Recently, a few other approaches based on neural networks surpassed the performance of GP in symbolic regression. We leave the discussions of these methods to the related work section. The high-level idea of GP is to maintain a pool of candidate symbolic expressions. In each generation, candidate expressions are *mutated* with probability P_{mu} and *mated* with probability P_{ma}. Then in the *selection* step, those with the highest fitness scores, measured by how each expression predicts the output from the input, are selected as the candidates for the next generation, together with a few randomly chosen ones to maintain diversity. After several generations, expressions with high fitness scores, *i.e.*, those fit data well survive in the pool of candidate solutions. The best expressions found in all generations are recorded as *hall-of-fame* solutions.

3 Control Variable Genetic Programming

In this section, we present our control variable genetic programming algorithm. Before we dive into the algorithm description, we first need to study what are the outcomes of a control variable experiment and what conclusions we can draw on the symbolic regression expression by observing such outcomes.

[1] The code is at: https://github.com/jiangnanhugo/cvgp/. Please refer to the extended version (https://arxiv.org/abs/2306.08057) for the Appendix.



(a) Ground-truth expression **(b)** Reduced form after controlling x_2, x_3, x_4 **(c)** Trial T_1 **(d)** Trial T_2

Fig. 1. An example of two trials of a control variable experiment. **(a)** The data of the experiment is generated by the ground-truth expression $\phi = x_1x_3 - x_2x_4$. **(b)** If we control $\mathbf{v}_c = \{x_2, x_3, x_4\}$ and only allow $\mathbf{v}_f = \{x_1\}$ to vary, it *looks like* the data are generated from the reduced-form equation $\phi' = C_1x_1 - C_2$. **(c, d)** The generated data in two trials of the control variable experiments. The controlled variables are fixed within each trial but vary across trials.

3.1 Control Variable Experiment

A control variable experiment $\texttt{CVExp}(\phi, \mathbf{v}_c, \mathbf{v}_f, \{T_k\}_{k=1}^K)$ consists of the trial symbolic expression ϕ, a set of controlled variables $\mathbf{v}_c$, a set of free variables $\mathbf{v}_f$, and K trial experiments $T_1, \ldots, T_K$. The expression ϕ may have zero or multiple *open constants*. The values of open constants are determined by fitting the equation to the training data.

One Trial in a Control Variable Experiment. A single trial of a control variable experiment T_k fits the symbolic expression ϕ with a batch of data. To avoid abusing notations, we also use T_k to denote the batch of data. In the generated data T_k, every controlled variable is fixed to the same value while the free variables are set randomly. We assume the values of the dependent variables in a batch are (noisy observations) of the ground-truth expressions with the values of independent variables set in the batch. In science, this step is achieved by conducting real-world experiments, *i.e.*, controlling independent variables and performing measurements on the dependent variable.

For example, Fig. 1(c,d) demonstrates two trials ($K = 2$) of a control variable experiment in which variable x_2, x_3, x_4 are controlled, *i.e.*, $\mathbf{v}_c = \{x_2, x_3, x_4\}$. They are fixed to one value in trial T_1 (in Fig. 1(c)) and another value in trial T_2 (in Fig. 1(d)). x_1 is the only free variable, *i.e.*, $\mathbf{v}_f = \{x_1\}$.

Reduced-form Expression in a Control Variable Setting. We assume there is a ground-truth symbolic expression that produces the experiment data. In other words, the observed output is the execution of the ground-truth expression from the input, possibly in addition to some noise. In control variable experiments, because the values of controlled variables are fixed in each trial, what we observe is the ground-truth expression in its *reduced form*, where sub-expressions involving only controlled variables are replaced with constants.

Fig. 1(b) provides an example of the reduced form expression. Assume the data is generated from the ground-truth expression in Fig. (a): $\phi = x_1x_3 - x_2x_4$. When we control the values of variable in $\mathbf{v}_c = \{x_2, x_3, x_4\}$, the data *looks like* they are generated from the *reduced* expression: $\phi' = C_1x_1 - C_2$. We can see

both C_1 and C_2 hold constant values in each trial. However, their values vary across trials because the values of controlled variables change. In trial T_1, when x_2, x_3, and x_4 are fixed to 0.5, 0.1, 0.7, C_1 takes the value of x_3, *i.e.*, 0.1. C_2 takes the value of x_2x_4, *i.e.*, 0.35. In trial T_2, $C_1 = 0.8$ and $C_2 = 0.06$.

We call constants which represent sub-expressions involving controlled variables in the ground-truth expression *summary constants*, and refer to constants in the ground-truth expression *stand-alone constants*. For example, C_1 and C_2 in Fig. 1(b) are both summary constants, because C_1 replaces the controlled variable x_3 and C_2 replaces a sub-expression x_2x_4 in the ground-truth expression. Notice the types of constants are *unknown* in the process of fitting an expression to control variable experiment data. However, the best-fitted values of these constants across several trials reveal important information: a constant is probably a summary constant if its fitted values vary greatly across trials, while a constant that remains the almost same value across trials is probably stand-alone.

Outcome of One Trial. The outcomes of one trial are two-fold: (1) the values of the constants which best fit the given batch of data. We denote these values as vector $\mathbf{c}$. (2) the fitness score measuring the goodness-of-fit, denoted as o. One typical fitness score is the mean squared error (MSE). See Appendix B.2 for the exact definition of MSE. For the example in Fig. 1, if we fit the reduced expression in (b) to data in trial T_1, the best-fitted values are $\mathbf{c}_1 = (C_1 = 0.1, C_2 = 0.35)$. For trial T_2, the best-fitted values are $\mathbf{c}_2 = (C_1 = 0.8, C_2 = 0.06)$. In both trials, the fitness scores (i.e., the MSE value) are 0, indicating no errors.

Outcome of Multiple trials. We let the values of control variables vary across different trials. This corresponds to changing experimental conditions in real science experiments. The outcomes of an experiment with K trials are: (1) $\phi.\mathbf{o} = (o_1, \ldots, o_K)$, where each o_k is the fitness score of trial k and (2) $\phi.\mathbf{c} = (\mathbf{c}_1, \ldots, \mathbf{c}_K)$, the best-fitted values to open constants across trials.

Key information is obtained by examining the outcomes of multi-trials control variable experiments: (1) consistent close-to-zero fitness scores $\phi.\mathbf{o}$ suggest the fitted expression is close to the ground-truth equation in the reduced form. (2) given the equation is close to the ground truth, an open constant having similar best-fitted values across K trials $\phi.\mathbf{c}$ suggests the open constants are stand-alone.

3.2 Control Variable Genetic Programming

The high-level idea of the CVGP algorithm is to build more complex symbolic expressions involving more and more variables based on control variable experiments with fewer and fewer controlled variables.

To fit an expression of m variables, initially, we control the values of all $m-1$ variables and allow only one variable to vary. Using Genetic Programming (GP), we find a pool of expressions $\{\phi_{1,1}, \ldots, \phi_{1,M}\}$ which best fit the data from this controlled experiment. Notice $\{\phi_{1,1}, \ldots, \phi_{1,M}\}$ are restricted to contain the only one free variable. This fact renders fitting them a lot easier than fitting the expressions involving all m variables. Next, for each $\phi_{1,l}$, we examine (1) if the errors of the fitting are consistently small across all trials. A small error implies $\phi_{1,l}$ is close to the ground-truth formula reduced to the one free variable. We

Algorithm 1 Control Variable Genetic Programming (CVGP)

Input: GP pool size M; #generations `#Gen`; #trials K; #expressions in hall-of-fame set `#Hof`; mutate probability P_{mu}; mate probability P_{ma}; operator set O_p.

1: $\mathbf{v}_c \leftarrow \{x_1, \ldots, x_m\}$; $\mathbf{v}_f \leftarrow \emptyset$.
2: $\mathcal{P}_{gp} \leftarrow$ `CreateInitGPPool`(M).
3: **for** $x_i \in \{x_1, \ldots, x_m\}$ **do**
4: $\quad \mathbf{v}_c \leftarrow \mathbf{v}_c \setminus \{x_i\}$; $\mathbf{v}_f \leftarrow \mathbf{v}_f \cup \{x_i\}$. ▷ Set x_i to be free variable
5: $\quad \mathcal{D}^o \leftarrow$ `DataOracle`$(\mathbf{v}_c, \mathbf{v}_f)$.
6: $\quad$ **for** $\phi \in \mathcal{P}_{gp}$ **do**
7: $\quad\quad \{T_k\}_{k=1}^K \leftarrow$ `GenData`$(\mathcal{D}^o)$. ▷ Query Oracle for the trial data
8: $\quad\quad \phi.\mathbf{o}, \phi.\mathbf{c} \leftarrow$ `CVExp`$(\phi, \mathbf{v}_c, \mathbf{v}_f, \{T_k\}_{k=1}^K)$. ▷ Control variable experiments
9: $\quad \mathcal{P}_{gp}, \mathcal{H} \leftarrow$ `GP`$(\mathcal{P}_{gp}, \mathcal{D}_i^o, K, M,$ `#Gen`, `#Hof`$, P_{mu}, P_{ma}, O_p \cup \{\text{const}, x_i\})$.
10: $\quad$ **for** $\phi \in \mathcal{P}_{gp}$ **do**
11: $\quad\quad$ `FreezeEquation`$(\phi, \phi.\mathbf{o}, \phi.\mathbf{c})$.
return The set of hall-of-fame equations $\mathcal{H}$.

hence freeze all operators of $\phi_{1,l}$ in this case. Freezing means GP in later steps cannot change these operators. (2) In the case of a small fitting error, we also inspect the best-fitted values of each open constant in $\phi_{1,l}$ across different trials. The constant is probably a summary constant if its values vary across trials. In other words, these constants represent sub-expressions involving the controlled variables. We thus mark these constants as *expandable* for later steps. The remaining constants are probably stand-alone. Therefore we also freeze them.

After the first step, CVGP adds a second free variable and starts fitting $\{\phi_{2,1}, \ldots, \phi_{2,M}\}$ using the data from control variable experiments involving the two free variables. Similar to the previous step, all $\phi_{2,l}$ are restricted to only contain the two free variables. Moreover, they can only be mated or mutated by GP from the first generation $\{\phi_{1,1}, \ldots, \phi_{1,M}\}$. The mutation can only happen on non-frozen nodes. After GP, a similar inspection is conducted for every equation in the GP pool, and corresponding variables and/or operators are frozen. This process continues to involve more and more variables. Eventually, the expressions in the GP pool consider all m variables.

The whole procedure of CVGP is shown in Algorithm 1. Here, $x_1, \ldots, x_m$ are moved from the controlled to free variables in numerical order. We agree other orders may boost its performance even further. However, we leave the exploration of this direction as future work. When a new variable becomes free, the control variable experiment `CVExp` needs to be repeated for every equation ϕ in the GP pool $\mathcal{P}_{gp}$ (Line 5-9 in Algorithm 1). This is because the fitness scores and the fitted open constant values will both change when the set of controlled variables is updated. Then function `GP` is called. `GP` is a minimally modified genetic programming algorithm for symbolic regression whose pseudocode is in Algorithm 2. The only differences are that it uses data from control variable experiments and the mutation operation at step i only allows to use all the operators, the constant node, and variable x_i at non-frozen nodes. Finally, in Lines 12-14 of Algorithm 1, `FreezeEquation` is called for every equation in

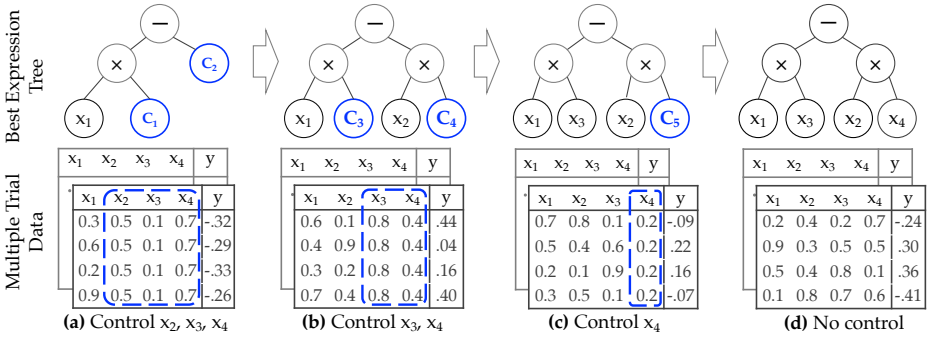


Fig. 2. Running example of Algorithm 1. **(a)** Initially, a reduced-form equation $\phi' = C_1x_1 - C_2$ is found via fitting control variable data in which x_2, x_3, x_4 are held as constants and only x_1 is allowed to vary. Two leaves nodes C_1, C_2 are as summary constants (colored blue). **(b)** This equation is expanded to $C_3x_1 - C_4x_2$ in the second stage via fitting the data in which only x_3, x_4 are held as constants. **(c,d)** This process continues until the ground-truth equation $\phi = x_1x_3 - x_2x_4$ is found. The data generated for control variable experiment trials in each stage are shown at the bottom.

the GP pool. The high-level idea of freezing is discussed above. $\mathcal{H}$ is returned as the set of "hall of fame" expressions.

Fig. 2 shows the high-level idea of fitting an equation using CVGP. Here the process has four stages, each stage with a decreased number of controlled variables. The trial data in each stage is shown at the bottom and the best expression found is shown at the top. The expandable constants are bold and blue. The readers can see how the fitted equations grow into the final ground-truth equation, with one free variable added at a time.

The Availability of a Data Oracle. A crucial assumption behind the success of CVGP is the availability of a data oracle $\mathcal{D}^o$ that returns a (noisy) observation of the dependent output variable with input variables in $\mathbf{v}_c$ controlled and $\mathbf{v}_f$ free. This differs from the classical setting of symbolic regression, where a dataset is obtained before learning [41, 49]. Such a data oracle represents conducting control variable experiments in the real world, which can be expensive.

However, we argue that the integration of experiment design in the discovery of scientific knowledge is indeed the main driver of the successes of CVGP. This idea has received tremendous success in early works [31, 32, 36] but unfortunately has been largely forgotten in today's symbolic regression community. Our work does not intend to show the superiority of one approach. Instead, we would like to point out that carefully designed experiments can improve any method, and GP is used as an example. We acknowledge that fully controlled experiments may be difficult in some scenarios. In cases where it is difficult to obtain such a data oracle, one possible solution is to use deep neural networks to learn a data generator for the given set of controlled variables. We leave it as future work.

Algorithm 2 `GP`$(\mathcal{P}_{gp}, \mathcal{D}^o, K, M,$ `#Gen`, `#Hof`, P_{mu}, P_{ma}, $O_p)$

Input: Initial GP Pool $\mathcal{P}_{gp}$; data Oracle $\mathcal{D}^o$; #trials K; GP pool size M; #generations `#Gen`; #expressions in hall-of-fame set `#Hof`; mutate probability P_{mu}; mate probability P_{ma}; mutation node library O_p.

for $j \leftarrow 1$ *to* `#Gen` **do**
  $\mathcal{P}_{new} \leftarrow \emptyset$;
  for $\phi \in \mathcal{P}_{gp}$ **do**
    if with probability P_{mu} **then** ▷ Mutation
      $\phi \leftarrow$ `Mutate`(ϕ, O_p);
      $\{T_k\}_{k=1}^K \leftarrow$ `GenData`$(\mathcal{D}^o)$;
      $\phi.\mathbf{o}$, $\phi.\mathbf{c} \leftarrow$ `CVExp`$(\phi, \mathbf{v}_c, \mathbf{v}_f, \{T_k\}_{k=1}^K)$;
    $\mathcal{P}_{new} \leftarrow \mathcal{P}_{new} \cup \{\phi\}$;
  $\mathcal{P}_{gp} \leftarrow \mathcal{P}_{new}$; $\mathcal{P}_{new} \leftarrow \emptyset$;
  for $\phi_l, \phi_{l+1} \in \mathcal{P}_{gp}$ **do**
    if with probability P_{ma} **then** ▷ Mating
      $\phi_l, \phi_{l+1} \leftarrow$ `Mate`(ϕ_l, ϕ_{l+1});
      $\{T_k\}_{k=1}^K \leftarrow$ `genData`$(\mathcal{D}^o)$;
      $\phi_l.\mathbf{o}$, $\phi_l.\mathbf{c} \leftarrow$ `CVExp`$(\phi_l, \mathbf{v}_c, \mathbf{v}_f, \{T_k\}_{k=1}^K)$.
      $\phi_{l+1}.\mathbf{o}$, $\phi_{l+1}.\mathbf{c} \leftarrow$ `CVExp`$(\phi_{l+1}, \mathbf{v}_c, \mathbf{v}_f, \{T_k\}_{k=1}^K)$.
    $\mathcal{P}_{new} \leftarrow \mathcal{P}_{new} \cup \{\phi_l, \phi_{l+1}\}$;
  $\mathcal{H} \leftarrow$ `TopK`$(\mathcal{P}_{new} \cup \mathcal{H}, K =$ `#Hof`$)$; ▷ Update the hall of fame set
  $\mathcal{P}_{gp} \leftarrow$ `selection`$(\mathcal{P}_{new}, M)$;
return GP pool and hall-of-fame $\mathcal{P}_{gp}, \mathcal{H}$.

3.3 Theoretical Analysis

We show in this section that the idea of control variable experiments may bring an exponential reduction in the search space for particular classes of symbolic expressions. To see this, we assume the learning algorithm follows a search order from simple to complex symbolic expressions and the data is noiseless.

Definition 1. *The search space of symbolic expression trees of l nodes $S(l)$ is the set of all symbolic expression trees involving at most l nodes.*

Lemma 1. *For simplicity, assume all operators are binary, and let o be the number of operators and m be the number of input variables. The size of the search space of symbolic expression trees of l nodes scales exponentially; more precisely at $\mathcal{O}((4(m+1)o)^{\frac{l-1}{2}})$ and $\Omega((4(m+1)o)^{\frac{l-1}{4}})$.*

The proof of Lemma 1 mainly involves counting binary trees. We leave its detailed proof in Appendix A. For our purposes, it is sufficient to know the size is exponential in the size of expression tree l.

Definition 2 (Simple to complex search order). *A symbolic regression algorithm follows a simple to complex search order if it expands its search space from short to long symbolic expressions; i.e., first search for the best symbolic expressions in $S(1)$, then in $S(2) \setminus S(1)$, etc.*

In general, it is difficult to quantify the search order of any symbolic regression algorithms. However, we believe the simple to complex order reflects the search procedures of a large class of symbolic regression algorithms, including our CVGP. In fact, [12] explicitly use regularizers to promote the search of simple and short expressions. Our CVGP follows the simple to complex search order approximately. Indeed, it is possible that genetic programming encounters more complex equations before their simpler counterparts. However, in general, the expressions are built from simple to complex equations by mating and mutating operations in genetic programming algorithms.

Proposition 1 (Exponential Reduction in the Search Space). *There exists a symbolic expression ϕ of $(4m-1)$ nodes, a normal symbolic regression algorithm following the simple to complex search order has to explore a search space whose size is exponential in m to find the expression, while CVGP following the simple to complex order only expands $\mathcal{O}(m)$ constant-sized search spaces.*

Proof. Consider a dataset generated by the ground-truth symbolic expression made up of 2 operators $(+, \times)$, $2m$ input variables, and $(4m-1)$ nodes:

$$(x_1 + x_2)(x_3 + x_4)\dots(x_{2m-1} + x_{2m}). \tag{2}$$

To search for this symbolic regression, a normal algorithm following the simple to complex order needs to consider all expression trees up to $(4m-1)$ nodes. According to Lemma 1, the normal algorithm has a search space of at least $\Omega((16m+8)^{m-1/2})$, which is exponential in m.

On the other hand, in the first step of CVGP, $x_2, \dots, x_{2m}$ are controlled and only x_1 is free. In this case, the ground-truth equation in the reduced form is

$$(x_1 + C_1)D_1, \tag{3}$$

in which both C_1 and D_1 are summary constants. Here C_1 represents x_2 and D_1 represents $(x_3 + x_4)\dots(x_{2m-1} + x_{2m})$ in the control variable experiments. The reduced equation is quite simple under the controlled environment. CVGP should be able to find the ground-truth expression exploring search space $S(5)$.

Proving using induction. In step $2i$ $(1 \le i \le m)$, variables $x_{2i+1}, x_{2i+2}, \dots, x_{2m}$ are held as constants, $x_1, \dots, x_{2i}$ are allowed to vary. The ground-truth expression in the reduced form found in the previous $(2i-1)$-th step is:

$$(x_1 + x_2)\dots(x_{2i-1} + C_{2i-1})D_{2i-1}. \tag{4}$$

CVGP needs to extend this equation to be the ground-truth expression in the reduced form for the $2i$-th step, which is:

$$(x_1 + x_2)\dots(x_{2i-1} + x_{2i})D_{2i}. \tag{5}$$

We can see the change is to replace the summary constant C_{2i-1} to x_{2i}. Assume the data is noiseless and CVGP can confirm expression (4) is the ground-truth reduced-form expression for the previous step. This means all the operators and

variables will be frozen by CVGP, and only C_{2i-1} and D_{2i-1} are allowed to be replaced by new expressions. Assume CVGP follows the simple to complex search order, it should find the ground-truth expression (5) *by searching replacement expressions of lengths up to 1.*

Similarly, in step $2i+1$, assume CVGP confirms the ground-truth expression in the reduced form in step $2i$, CVGP also only needs to search in constant-sized spaces to find the new ground-truth expression. Overall, we can see only $\mathcal{O}(m)$ searches in constant-sized spaces are required for CVGP to find the final ground-truth expression.

4 Related Work

Symbolic Regression. Symbolic Regression is proven to be NP-hard [58], due to the search space of all possible symbolic expressions being exponential in the number of input variables. Early works in this domain are based on heuristic search [33, 39]. Genetic programming turns out to be effective in searching for good candidates of symbolic expressions [54, 57, 51, 22]. RL-based methods propose a risk-seeking policy gradient to find the expressions [45, 51, 43]. Other works use RL to adjust the probabilities of genetic operations [11]. Also, there are works that reduced the search space by considering the composition of base functions, *e.g.* Fast function extraction [42] and elite bases regression [10]. In terms of the families of expressions, research efforts have been devoted to searching for polynomials with single or two variables [55], time series equations [3], and also equations in physics [54]. Existing works for multi-variable regression are mainly based on pre-trained encoder-decoder methods with a massive training dataset (e.g., millions of datasets [4]), and even larger generative models (e.g., about 100 million parameters [26]). Our CVGP is a tailored algorithm to solve multi-variable symbolic regression problems.

AI-driven Scientific Discovery. Recently AI has been highlighted to enable scientific discoveries in diverse domains [37, 59]. Early work in this domain focuses on learning logic (symbolic) representations [6, 7]. Recently, learning Partial Differential Equations (PDEs) from data has also been studied extensively [16, 8, 61, 63, 23, 14, 47, 46, 40, 62, 13]. In this domain, a line of works develops robots that automatically refine the hypothesis space, some with human interactions [56, 29, 28]. These works are relevant to ours because they actively probe the hypothesis spaces, albeit they are in biology and chemistry.

Active Learning and Reasoning. Active learning considers querying data points actively to maximize the learning performance [21, 19]. Our approach is related to active learning because control variable experiments can be viewed as a way to actively collect data. However, besides active data collection, our CVGP builds simple to complex models, which is not in active learning.

Meta-reasoning – Thinking Fast and Slow. The co-existence of fast and slow cognition systems marks an interesting side of human intelligence [25, 2, 5]. Our CVGP is motivated by this dual cognition process. In essence, we argue instead of entirely relying on the brute-force way of learning using big data and

heavy computation (fast thinking), incrementally expanding from reduced-form equations to the full equation may result in better outcomes (slow thinking).
Causality. Control variable experiments are closely related to the idea of intervention, which is commonly used to discover causal relationships [53, 35, 18, 24, 44]. However, we mainly use control variable experiments to accelerate symbolic regression, which still identifies correlations.

5 Experiments

In this section, we demonstrate CVGP finds the symbolic expressions with the smallest Normalized Mean-Square Errors (NMSE) among all 7 competing approaches on 21 noiseless benchmark datasets (in Table 1) and 20 noisy benchmark datasets (in Table 2). In the ablation studies, we show our CVGP is consistently better than the baselines when evaluated in different evaluation metrics, evaluating different quantiles of the NMSE metric, with different amounts of Gaussian noise added to the data (Fig. 3, more complete results in Fig. 4 and 5 in the appendix). In Table 3, we show our CVGP has a higher rate of recovering the ground-truth expressions than baselines.

5.1 Experimental Settings

Datasets. To highlight the performance of CVGP in regressing multi-variable expressions, we consider synthesized datasets, involving randomly generated expressions with multiple variables. A dataset is labeled by the ground-truth equation that generates it. The ground-truth equations we consider are multi-variable polynomials characterized by their operators and a tuple (a, b, c). Here a is the number of independent variables. b is the number of singular terms. A singular term can be an independent variable (like x_1), or a unary operator with a variable (like $\sin(x_1)$). c is the number of cross terms. They look like $C_1 x_3 x_4$ or $C_2 \sin(x_1)\texttt{inv}(x_5)$, etc. Here C_1, C_2 are randomly generated constants. The tuples and operators listed in different tables and charts indicate how the ground-truth expressions are generated. For each dataset configuration, we repeat our experiments 10 times, each time with a randomly generated symbolic expression of the given configuration. For noiseless datasets, the output is exactly the evaluation of the ground-truth expression. For noisy datasets, the output is further perturbed by Gaussian noise of zero means and a given standard deviation.
Remarks on Public Available Datasets. Most public datasets are black-box [30], containing randomly generated input and output pairs of an unknown symbolic equation. The point of our paper is to show customized collected control variable experiment data improves symbolic regression, and hence we cannot use these randomly generated data. In addition, most datasets are on equations of a small number of independent variables. We intentionally test on benchmark sets involving many variables to highlight our approach.
Evaluation. In terms of the evaluation metric, the median (50%) and 75%-percentile of the NMSE across these 10 experiments are reported. We choose to

Table 1. Median (50%) and 75%-quantile NMSE values of the symbolic expressions found by all the algorithms on several *noiseless* benchmark datasets. Our CVGP finds symbolic expressions with the smallest NMSEs.

Dataset	CVGP (ours)		GP		DSR		PQT		VPG		GPMeld		Eureqa	
configs	50%	75%	50%	75%	50%	75%	50%	75%	50%	75%	50%	75%	50%	75%
(3,2,2)	0.001	0.004	0.015	0.135	1.53	43.09	0.58	1.13	0.83	1.32	1.06	2.18	<**1e-6**	<**1e-6**
(4,4,6)	**0.008**	0.059	0.012	**0.054**	1.006	1.249	1.006	2.459	1.221	2.322	1.127	2.286	1.191	6.001
(5,5,5)	**0.011**	**0.019**	0.025	0.177	1.038	8.805	1.048	4.736	1.401	38.26	1.008	1.969	0.996	6.340
(5,5,8)	**0.007**	**0.013**	0.010	0.017	1.403	5.161	1.530	41.27	4.133	27.42	1.386	8.092	1.002	1.495
(6,6,8)	**0.044**	**0.074**	0.058	0.200	1.963	90.53	4.212	8.194	4.425	22.91	15.58	269.6	1.005	1.150
(6,6,10)	**0.012**	**0.027**	0.381	0.820	1.021	1.036	1.006	1.048	1.003	1.020	1.022	1.689	1.764	49.041
(a) Datasets containing operators $\{\texttt{inv}, +, -, \times\}$														
(3,2,2)	0.005	0.123	0.023	0.374	0.087	0.392	0.161	0.469	0.277	0.493	0.112	0.183	<**1e-6**	<**1e-6**
(4,4,6)	**0.028**	0.132	0.044	**0.106**	2.815	9.958	2.381	13.844	2.990	11.316	1.670	2.697	0.024	0.122
(5,5,5)	0.086	0.402	**0.063**	**0.232**	2.558	3.313	2.168	2.679	1.903	2.780	1.501	2.295	0.158	0.377
(5,5,8)	**0.014**	**0.066**	0.102	0.683	2.535	2.933	2.482	2.773	2.440	3.062	2.422	3.853	0.284	0.514
(6,6,8)	**0.066**	**0.166**	0.127	0.591	0.936	1.079	0.983	1.053	0.900	1.018	0.964	1.428	0.433	1.564
(6,6,10)	**0.104**	**0.177**	0.159	0.230	6.121	16.32	5.750	16.29	3.857	19.82	7.393	21.709	0.910	1.927
(b) Datasets containing operators $\{\sin, \cos, +, -, \times\}$.														
(3,2,2)	0.039	0.083	0.043	0.551	0.227	7.856	0.855	2.885	0.233	0.400	0.944	1.263	<**1e-6**	<**1e-6**
(4,4,6)	**0.015**	**0.121**	0.042	0.347	1.040	1.155	1.039	1.055	1.049	1.068	1.886	4.104	0.984	1.196
(5,5,5)	**0.038**	**0.097**	0.197	0.514	3.892	69.98	4.311	23.66	5.542	8.839	9.553	16.92	0.901	1.007
(5,5,8)	**0.050**	**0.102**	0.111	0.177	2.379	2.526	1.205	2.336	1.824	2.481	1.142	1.874	1.002	2.445
(6,6,8)	**0.029**	**0.038**	0.091	0.151	1.605	8.005	1.718	7.783	4.691	39.03	1.398	16.60	1.001	1.008
(6,6,10)	**0.018**	**0.113**	0.087	0.194	2.083	23.57	1.797	4.521	1.888	35.45	2.590	8.784	1.001	1.008
(c) Datasets containing opeartors $\{\sin, \cos, \texttt{inv}, +, -, \times\}$.														

report median values instead of mean due to outliers (see box plots in Fig. 3(a-d)). This is a common practice for combinatorial optimization problems. The mathematical definition of NMSE and other metrics are in Appendix B.2.

Baselines. We consider the following baselines based on evolutionary algorithms: 1) Genetic Programming (GP) [17]. 2) Eureqa [15]. We also consider a series of baselines using reinforcement learning: 3) Priority queue training (PQT) [1]. 4) Vanilla Policy Gradient (VPG) that uses the REINFORCE algorithm [60] to train the model. 5) Deep Symbolic Regression (DSR) [45]. 6) Neural-Guided Genetic Programming Population Seeding (GPMeld) [43].

We leave detailed descriptions of the configurations of our CVGP and baseline algorithms in Appendix B and only mention a few implementation notes here. We implemented GP and CVGP. They use a data oracle, which returns (noisy) observations of the ground-truth equation when queried with inputs. We cannot implement the same Oracle for other baselines because of code complexity and/or no available code. To ensure fairness, the sizes of the training datasets we use for those baselines are larger than the total number of data points accessed in the full execution of those algorithms. In other words, their access to data would have no difference if the same oracle has been implemented for them because it does not affect the executions whether the data is generated ahead of the execution or on the fly. The reported NMSE scores in all charts and tables are based on separately generated data that have never been used in training. The threshold to freeze operators in CVGP is if the MSE to fit a data batch is below

Table 2. Median (50%) and 75%-quantile NMSE values of the symbolic expressions found by all the algorithms on several *noisy* benchmark datasets (Gaussian noise with zero mean and standard deviation 0.1 is added). Our CVGP finds symbolic expressions with the smallest NMSEs.

Dataset	CVGP (ours)		GP		DSR		PQT		VPG		GPMeld	
configs	50%	75%	50%	75%	50%	75%	50%	75%	50%	75%	50%	75%
(4,4,6)	**0.036**	**0.088**	0.038	0.108	1.163	3.714	1.016	1.122	1.087	1.275	1.058	1.374
(5,5,5)	0.076	0.126	**0.075**	**0.102**	1.028	2.270	1.983	4.637	1.075	2.811	1.479	2.855
(5,5,8)	**0.061**	**0.118**	0.121	0.186	1.004	1.013	1.005	1.006	1.002	1.009	1.108	2.399
(6,6,8)	**0.098**	**0.144**	0.104	0.167	1.006	1.027	1.006	1.020	1.009	1.066	1.035	2.671
(6,6,10)	**0.055**	**0.097**	0.074	0.132	1.003	1.009	1.005	1.008	1.004	1.015	1.021	1.126
(a) Datasets containing operators $\{\sin, \cos, \texttt{inv}, +, -, \times\}$.												
(3,2,2)	**0.098**	**0.165**	0.108	0.425	0.350	0.713	0.351	1.831	0.439	0.581	0.102	0.597
(4,4,6)	**0.078**	**0.121**	0.120	0.305	7.056	16.321	5.093	19.429	2.458	13.762	2.225	3.754
(5,5,5)	**0.067**	**0.230**	0.091	0.313	32.45	234.31	36.797	229.529	14.435	46.191	28.440	421.63
(5,5,8)	**0.113**	**0.207**	0.119	0.388	195.22	573.33	449.83	565.69	206.06	629.41	363.79	666.57
(6,6,8)	**0.170**	**0.481**	0.186	0.727	1.752	3.824	4.887	15.248	2.396	7.051	1.478	6.271
(6,6,10)	**0.161**	**0.251**	0.312	0.342	11.678	26.941	5.667	24.042	7.398	25.156	11.513	28.439
(b) Datasets containing operators $\{\sin, \cos, +, -, \times\}$.												
(3,2,2)	0.049	**0.113**	**0.023**	0.166	0.663	2.773	1.002	1.992	0.969	1.310	0.413	2.510
(4,4,6)	**0.141**	**0.220**	0.238	0.662	1.031	1.051	1.297	1.463	1.051	1.774	1.093	1.769
(5,5,5)	**0.157**	0.438	0.195	**0.337**	1.098	3.617	1.018	5.296	1.012	1.27	1.036	3.617
(5,5,8)	**0.122**	**0.153**	0.166	0.186	1.009	1.103	1.017	1.429	1.007	1.132	1.07	2.904
(6,6,8)	**0.209**	**0.590**	**0.209**	0.646	1.003	1.153	1.047	1.134	1.059	1.302	1.029	3.365
(6,6,10)	0.139	0.232	**0.073**	**0.159**	1.654	3.408	1.027	1.069	1.009	1.654	1.445	2.106
(c) Datasets containing operators $\{\sin, \cos, \texttt{inv}, +, -, \times\}$.												

0.01. The threshold to freeze the value of a constant in CVGP is if the variance of best-fitted values of the constant across trials drops below 0.001.

5.2 Experimental Analysis

Learning Result. Our CVGP attains the smallest median (50%) and 75%-quantile NMSE values among all the baselines mentioned in Section 5.1, when evaluated on noiseless datasets (Table 1) and noisy datasets (Table 2). This shows our method can better handle multiple variables symbolic regression problems than the current best algorithms in this area.

Ablation Studies. We use box plots in Fig. 3(a-d) to show that the superiority of our CVGP generalizes to other quantiles beyond the 50% and 75%-quantile. We also show the performance is consistent under the variations of evaluation metrics in Fig. 3(a-d), and noise levels in Fig. 3(e-f).

Recovering Ground-truth Equations. For relatively less challenging noiseless datasets (*i.e.*, $(2, 1, 1)$ with various operators sets), our CVGP sometimes recovers ground-truth expressions. We evaluate the percentage that each algorithm successfully detects the ground-truth expressions on 50 randomly generated benchmark datasets. Table 3 shows that our CVGP algorithm has a higher chance to recover ground-truth expressions than the GP method.

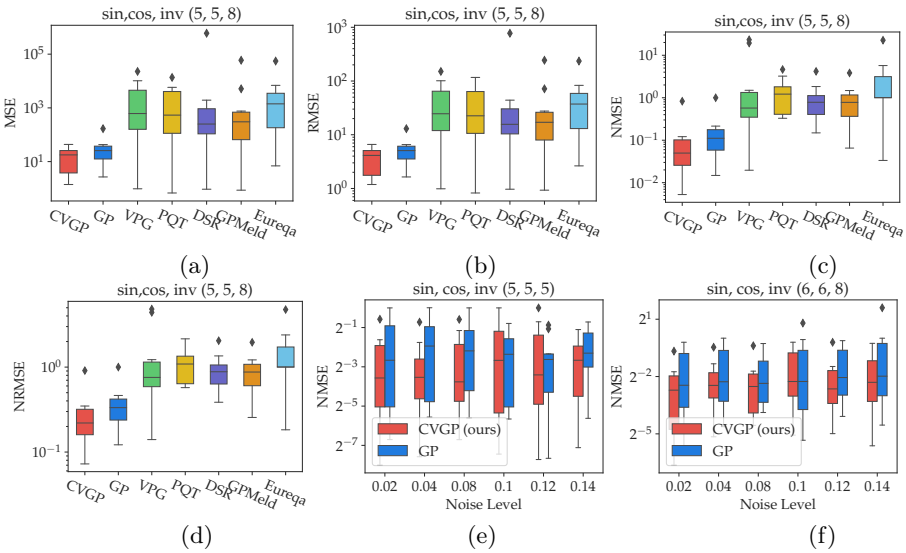


Fig. 3. (a-d) Box plots of evaluation metrics for the expressions found by different algorithms on the noiseless dataset. **(e-f)** Box plots in NMSE values for the expressions found by CVGP and GP over benchmark datasets with different noise levels. Our CVGP is consistently the best regardless of the evaluation metrics and noise levels.

Table 3. Ground-truth recovery rate comparison. Our CVGP has a higher rate to recover the ground-truth expressions compared to GP on 3 simple datasets.

Operator set	Dataset configs	CVGP (ours)	GP
$\{\texttt{inv}, +, -, \times\}$	(2,1,1)	**64**%	44%
$\{\sin, \cos, +, -, \times\}$		**46**%	22%
$\{\sin, \cos, \texttt{inv}, +, -, \times\}$		**44**%	32%

6 Conclusion

In this research, we propose Control Variable Genetic Programming (CVGP) for symbolic regression with many independent variables. This is beyond current state-of-the-art approaches mostly tested on equations with one or two variables. CVGP builds equations involving more and more independent variables via control variable experimentation. Theoretically, we show CVGP as an incremental building approach can bring an exponential reduction in the search spaces when learning a class of expressions. In experiments, CVGP finds the best-fitted expressions among 7 competing approaches and on dozens of benchmarks.

Acknowledgments

We thank all the reviewers for their constructive comments. This research was supported by NSF grant CCF-1918327.

References

1. Abolafia, D.A., Norouzi, M., Le, Q.V.: Neural program synthesis with priority queue training. CoRR **abs/1801.03526** (2018)
2. Anthony, T., Tian, Z., Barber, D.: Thinking fast and slow with deep learning and tree search. In: NIPS. pp. 5360–5370 (2017)
3. Balcan, M., Dick, T., Sandholm, T., Vitercik, E.: Learning to branch. In: ICML. Proceedings of Machine Learning Research, vol. 80, pp. 353–362. PMLR (2018)
4. Biggio, L., Bendinelli, T., Neitz, A., Lucchi, A., Parascandolo, G.: Neural symbolic regression that scales. In: ICML. Proceedings of Machine Learning Research, vol. 139, pp. 936–945. PMLR (2021)
5. Booch, G., Fabiano, F., Horesh, L., Kate, K., Lenchner, J., Linck, N., Loreggia, A., Murugesan, K., Mattei, N., Rossi, F., Srivastava, B.: Thinking fast and slow in AI. In: AAAI. pp. 15042–15046. AAAI Press (2021)
6. Bradley, E., Easley, M., Stolle, R.: Reasoning about nonlinear system identification. Artificial Intelligence **133**(1), 139–188 (2001)
7. Bridewell, W., Langley, P., Todorovski, L., Džeroski, S.: Inductive process modeling. Machine Learning **71**, 1–32 (2008)
8. Brunton, S.L., Proctor, J.L., Kutz, J.N.: Discovering governing equations from data by sparse identification of nonlinear dynamical systems. Proceedings of the National Academy of Sciences **113**(15), 3932–3937 (2016)
9. Cerrato, M., Brugger, J., Schmitt, N., Kramer, S.: Reinforcement learning for automated scientific discovery. In: AAAI Spring Symposium on Computational Approaches to Scientific Discovery (2023)
10. Chen, C., Luo, C., Jiang, Z.: Elite bases regression: A real-time algorithm for symbolic regression. In: ICNC-FSKD. pp. 529–535. IEEE (2017)
11. Chen, D., Wang, Y., Gao, W.: Combining a gradient-based method and an evolution strategy for multi-objective reinforcement learning. Appl. Intell. **50**(10), 3301–3317 (2020)
12. Chen, Q., Xue, B., Zhang, M.: Rademacher complexity for enhancing the generalization of genetic programming for symbolic regression. IEEE Trans. Cybern. **52**(4), 2382–2395 (2022)
13. Chen, R.T., Rubanova, Y., Bettencourt, J., Duvenaud, D.K.: Neural ordinary differential equations. Advances in neural information processing systems **31** (2018)
14. Cranmer, M.D., Sanchez-Gonzalez, A., Battaglia, P.W., Xu, R., Cranmer, K., Spergel, D.N., Ho, S.: Discovering symbolic models from deep learning with inductive biases. In: NeurIPS (2020)
15. Dubcáková, R.: Eureqa: software review. Genet. Program. Evolvable Mach. **12**(2), 173–178 (2011)
16. Dzeroski, S., Todorovski, L.: Discovering dynamics: From inductive logic programming to machine discovery. J. Intell. Inf. Syst. **4**(1), 89–108 (1995)
17. Fortin, F.A., De Rainville, F.M., Gardner, M.A., Parizeau, M., Gagné, C.: DEAP: Evolutionary algorithms made easy. Journal of Machine Learning Research **13**, 2171–2175 (jul 2012)
18. Glymour, C., Scheines, R., Spirtes, P.: Discovering causal structure: Artificial intelligence, philosophy of science, and statistical modeling. Academic Press (2014)
19. Golovin, D., Krause, A., Ray, D.: Near-optimal bayesian active learning with noisy observations. Advances in Neural Information Processing Systems **23** (2010)
20. Guimerà, R., Reichardt, I., Aguilar-Mogas, A., Massucci, F.A., Miranda, M., Pallarès, J., Sales-Pardo, M.: A bayesian machine scientist to aid in the solution of challenging scientific problems. Science advances **6**(5), eaav6971 (2020)

21. Hanneke, S.: Theory of disagreement-based active learning. Found. Trends Mach. Learn. **7**(2-3), 131–309 (2014)
22. He, B., Lu, Q., Yang, Q., Luo, J., Wang, Z.: Taylor genetic programming for symbolic regression. In: GECCO. pp. 946–954. ACM (2022)
23. Iten, R., Metger, T., Wilming, H., Del Rio, L., Renner, R.: Discovering physical concepts with neural networks. Physical review letters **124**(1), 010508 (2020)
24. Jaber, A., Ribeiro, A., Zhang, J., Bareinboim, E.: Causal identification under markov equivalence: calculus, algorithm, and completeness. Advances in Neural Information Processing Systems **35**, 3679–3690 (2022)
25. Kahneman, D.: Thinking, fast and slow. Macmillan (2011)
26. Kamienny, P., d'Ascoli, S., Lample, G., Charton, F.: End-to-end symbolic regression with transformers. In: NeurIPS (2022)
27. Kibler, D.F., Langley, P.: The experimental study of machine learning (1991)
28. King, R.D., Rowland, J., Oliver, S.G., Young, M., Aubrey, W., Byrne, E., Liakata, M., Markham, M., Pir, P., Soldatova, L.N., Sparkes, A., Whelan, K.E., Clare, A.: The automation of science. Science **324**(5923), 85–89 (2009)
29. King, R.D., Whelan, K.E., Jones, F.M., Reiser, P.G., Bryant, C.H., Muggleton, S.H., Kell, D.B., Oliver, S.G.: Functional genomic hypothesis generation and experimentation by a robot scientist. Nature **427**(6971), 247–252 (2004)
30. La Cava, W., Orzechowski, P., Burlacu, B., de França, F.O., Virgolin, M., Jin, Y., Kommenda, M., Moore, J.H.: Contemporary symbolic regression methods and their relative performance. arXiv preprint arXiv:2107.14351 (2021)
31. Langley, P.: BACON: A production system that discovers empirical laws. In: IJCAI. p. 344. William Kaufmann (1977)
32. Langley, P.: Rediscovering physics with BACON.3. In: IJCAI. pp. 505–507. William Kaufmann (1979)
33. Langley, P.: Data-driven discovery of physical laws. Cognitive Science **5**(1), 31–54 (1981)
34. Langley, P.: Machine learning as an experimental science. Mach. Learn. **3**, 5–8 (1988)
35. Langley, P.: Scientific discovery, causal explanation, and process model induction. Mind & Society **18**(1), 43–56 (2019)
36. Langley, P., Bradshaw, G.L., Simon, H.A.: BACON.5: the discovery of conservation laws. In: IJCAI. pp. 121–126. William Kaufmann (1981)
37. Langley, P.W., Simon, H.A., Bradshaw, G., Zytkow, J.M.: Scientific Discovery: Computational Explorations of the Creative Process. The MIT Press (02 1987)
38. Lehman, J.S., Santner, T.J., Notz, W.I.: Designing computer experiments to determine robust control variables. Statistica Sinica pp. 571–590 (2004)
39. Lenat, D.B.: The ubiquity of discovery. Artificial Intelligence **9**(3), 257–285 (1977)
40. Liu, Z., Tegmark, M.: Machine learning conservation laws from trajectories. Phys. Rev. Lett. **126**, 180604 (May 2021)
41. Matsubara, Y., Chiba, N., Igarashi, R., Taniai, T., Ushiku, Y.: Rethinking symbolic regression datasets and benchmarks for scientific discovery. arXiv preprint arXiv:2206.10540 (2022)
42. McConaghy, T.: Ffx: Fast, scalable, deterministic symbolic regression technology. In: Genetic Programming Theory and Practice IX, pp. 235–260. Springer (2011)
43. Mundhenk, T.N., Landajuela, M., Glatt, R., Santiago, C.P., Faissol, D.M., Petersen, B.K.: Symbolic regression via deep reinforcement learning enhanced genetic programming seeding. In: NeurIPS. pp. 24912–24923 (2021)
44. Pearl, J.: Causality. Cambridge university press (2009)

45. Petersen, B.K., Landajuela, M., Mundhenk, T.N., Santiago, C.P., Kim, S., Kim, J.T.: Deep symbolic regression: Recovering mathematical expressions from data via risk-seeking policy gradients. In: ICLR. OpenReview.net (2021)
46. Raissi, M., Perdikaris, P., Karniadakis, G.: Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. Journal of Computational Physics **378**, 686–707 (2019)
47. Raissi, M., Yazdani, A., Karniadakis, G.E.: Hidden fluid mechanics: Learning velocity and pressure fields from flow visualizations. Science **367**(6481), 1026–1030 (2020)
48. Razavi, S., Gamazon, E.R.: Neural-network-directed genetic programmer for discovery of governing equations. CoRR **abs/2203.08808** (2022)
49. Ryan, T.P., Morgan, J.P.: Modern experimental design. Journal of Statistical Theory and Practice **1**(3-4), 501–506 (2007)
50. Santner, T.J., Williams, B.J., Notz, W.I.: The Design and Analysis of Computer Experiments. Springer series in statistics, Springer (2003)
51. Scavuzzo, L., Chen, F.Y., Chételat, D., Gasse, M., Lodi, A., Yorke-Smith, N., Aardal, K.: Learning to branch with tree mdps. In: NeurIPS (2022)
52. Schmidt, M., Lipson, H.: Distilling free-form natural laws from experimental data. Science **324**(5923), 81–85 (2009)
53. Simon, H.A.: Spurious correlation: A causal interpretation. Journal of the American statistical Association **49**(267), 467–479 (1954)
54. Udrescu, S.M., Tegmark, M.: Ai feynman: A physics-inspired method for symbolic regression. Science Advances **6**(16) (2020)
55. Uy, N.Q., Hoai, N.X., O'Neill, M., McKay, R.I., López, E.G.: Semantically-based crossover in genetic programming: application to real-valued symbolic regression. Genet. Program. Evolvable Mach. **12**(2), 91–119 (2011)
56. Valdés-Pérez, R.: Human/computer interactive elucidation of reaction mechanisms: application to catalyzed hydrogenolysis of ethane. Catalysis Letters **28**, 79–87 (1994)
57. Virgolin, M., Alderliesten, T., Bosman, P.A.N.: Linear scaling with and within semantic backpropagation-based genetic programming for symbolic regression. In: GECCO. pp. 1084–1092. ACM (2019)
58. Virgolin, M., Pissis, S.P.: Symbolic regression is NP-hard. Transactions on Machine Learning Research (2022)
59. Wang, H., Fu, T., Du, Y., Gao, W., Huang, K., Liu, Z., Chandak, P., Liu, S., Katwyk, P.V., Deac, A., Anandkumar, A., Bergen, K., Gomez, C.P., Ho, S., Kohli, P., Lasenby, J., Leskovec, J., Liu, T.Y., Manrai, A., Marks, D., Ramsundar, B., Song, L., Sun, J., Tang, J., Velickovic, P., Welling, M., Coley, C., Bengio, Y., Zitnik, M.: Enabling scientific discovery with artificial intelligence. Nature (2022)
60. Williams, R.J.: Simple statistical gradient-following algorithms for connectionist reinforcement learning. Mach. Learn. **8**, 229–256 (1992)
61. Wu, T., Tegmark, M.: Toward an artificial intelligence physicist for unsupervised learning. Phys. Rev. E **100**, 033311 (Sep 2019)
62. Xue, Y., Nasim, M., Zhang, M., Fan, C., Zhang, X., El-Azab, A.: Physics knowledge discovery via neural differential equation embedding. In: ECML/PKDD (5). Lecture Notes in Computer Science, vol. 12979, pp. 118–134. Springer (2021)
63. Zhang, S., Lin, G.: Robust data-driven discovery of governing physical laws with error bars. Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences **474**(2217), 20180305 (2018)