Expediting Symbolic Regression for Science Using Scientific Approaches

Md Nasim$^{[0000-0002-6930-9466]}$, Nan Jiang$^{[0000-0001-6863-2897]}$, and Yexiang Xue$^{[0000-0002-4533-0543]}$

Purdue University, West Lafayette, IN, USA
{mnasim,jiang631,yexiang}@purdue.edu

Abstract. Automating the discovery of symbolic equations from data, i.e., symbolic regression, is a grand goal of Artificial Intelligence (AI) but has never been fully realized. Despite the tremendous progress made in this area, scientific discovery using scientific approaches was largely overlooked in AI. This chapter presents an overview of successful approaches over the last few decades that leverage scientific approaches to expedite scientific discovery. We discuss the architecture and the lessons learned deploying BACON, LAGRANGE, Adam, Eve, and AI Feynman systems. We also describe Control Variable Genetic Programming (CVGP) – our recently developed symbolic regression algorithm, which uses control variable experiments to expedite symbolic regression over equations with many independent variables. CVGP expedites symbolic expression discovery via customized control variable experiments. It starts by fitting simple expressions involving a small set of independent variables using genetic programming, under controlled experiments where the remaining variables are held as constants. It then adds new independent variables into these equations, harnessing new control variable experiments in which these variables are allowed to vary. We demonstrate that CVGP outperforms state-of-the-art symbolic regressors in learning equations involving many independent variables.

Keywords: Scientific Approaches · Control Variable Experiment · Symbolic Regression · AI-driven Scientific Discovery

1 Introduction

Scientific discovery of new knowledge from experimental data can revolutionize our understanding of the physical world and lead to breakthroughs in new technologies. AI-driven scientific discovery refers to the process of using artificial intelligence to accelerate this scientific discovery process in various domains such as physics, chemistry, biology, etc. Automatic AI-driven methods for scientific discovery offer several benefits – efficient analysis of high-volume experimental data, detailed record-keeping of the experimental setups to ensure reproducibility, high throughput scientific hypothesis forming and testing, etc. Although many attempts have been made to use machines for automating scientific discovery, dating back to the foundation of AI [31], autonomous scientific discovery still remains a challenging task.

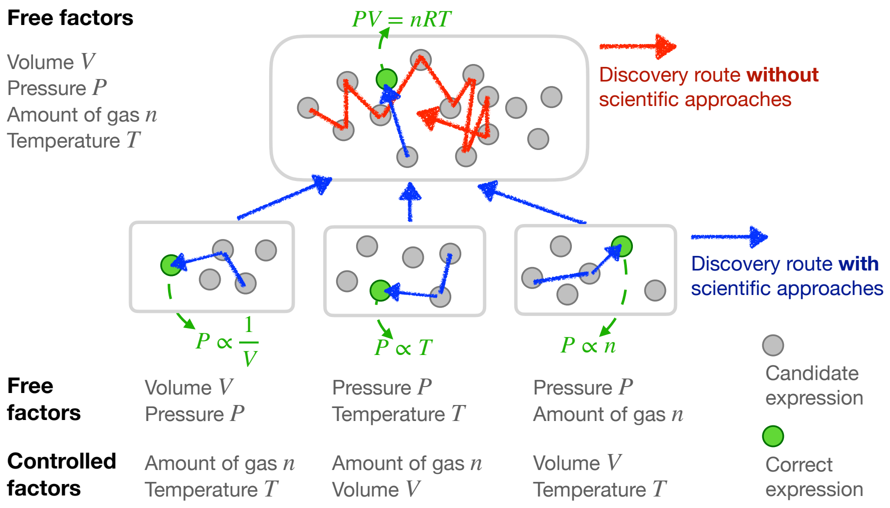


Fig. 1. Using scientific approaches (especially control variable experiments) accelerates the discovery of the ideal gas law, compared to non-scientific reasoning processes.

One of the problems that have particularly intrigued the interest of the AI and machine learning research community is the scientific discovery of symbolic equations from experimental data. Symbolic equations succinctly describe verbose scientific knowledge and provide insights into the relationship between interrelated variables of a system. An example is the ideal gas law $PV = nRT$ representing the relation between pressure P, volume V, amount of gas n, and temperature T for an ideal gas. The equation $PV = nRT$ describes several relationships at once:

1. $P \propto T$, when n, V are constant [27];
2. $P \propto \frac{1}{V}$, when n, T are constant [2];
3. $P \propto n$, when V, T are constant.

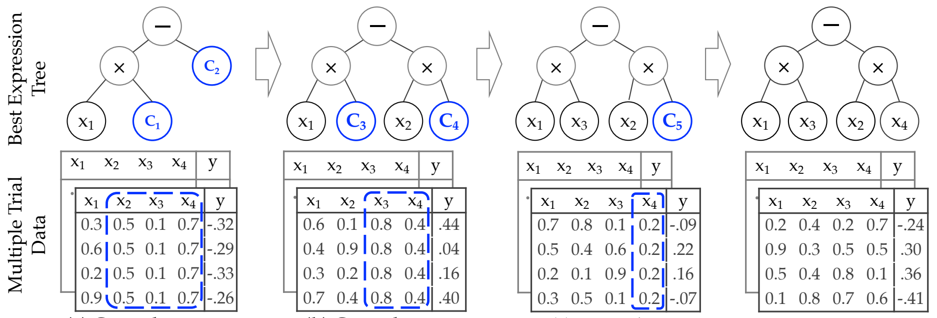
The benefit of symbolic equations – the succinct representation of multitudes of scientific knowledge – unfortunately poses a key challenge in the automatic machine discovery of such equations. The discovery of the ideal gas law equation requires knowledge of three separate physical processes. For complex systems with many contributing variables and/or interactive physical processes, the discovery of underlying governing equations becomes even more challenging.

Researchers have put much effort into automatic symbolic equation discovery i.e., symbolic regression, notably using genetic programming, search-based methods, reinforcement learning, deep neural networks, sparse identification, integrated systems, etc. A common theme that many of the current symbolic regression algorithms share is that they search for the optimal symbolic equation involving all relevant variables. Since the hypothesis space of possible equations grows exponentially, such methods are better suited for simple equations involving a small number of variables. For complex equations with many interacting variables, searching the space of all possible equations becomes computationally intractable.

Interestingly, human scientists have been able to discover complex symbolic equations including algebraic, and differential equations involving many variables. Their secret weapon is scientific approaches, specifically control variable experiments. In control variable experiments, scientists study the relationship between a subset of the variables of a system at a time, while *controlling* the remaining variables i.e., keep them at fixed constant values. Studying the relationship between free variables i.e., variables that are not controlled, is a much simpler task than studying the relationship of all variables at once. The result of this controlled study is a reduced-form equation representing the relationship among free variables. Once this simple equation is validated using observation data, scientists can improve upon this reduced-form equation iteratively, by freeing up some of the controlled variables in each iteration and searching for the optimal symbolic equation involving those newly freed variables. This process continues until an equation involving all variables has been discovered.

We see an example of how scientific approaches in the form of the control variable experiment can be used to discover symbolic equations such as ideal gas law $PV = nRT$ in Figure 1. At the beginning, we control gas amount n and temperature T, and explore symbolic equation relating pressure P and volume V of the gas. This results in the discovery that pressure is inversely proportional to volume, $P \propto \frac{1}{V}$. We then study the relationship between the pressure P and temperature T, with the rest of the variables held as constants. The result is $P \propto T$. Similarly, we can find the relationship between pressure P and the amount of gas n, which is $P \propto n$. Finally, when considering all the variables, the expressions that can be reduced to all the pre-discovered equations would be $PV = nRT$. Controlling a subset of variables in this manner is beneficial because it eliminates the effect of the controlled variable on the measured outputs, and thus greatly reduces the hypothesis space of possible symbolic equations.

Over the last several decades, a group of scientists went beyond the mainstream machine learning paradigm and tested how scientific principles can be incorporated into the AI-driven knowledge discovery process. They explored how an AI agent using scientific approaches, i.e., building and validating increasingly more complex models using self-designed control variable experiments, can expedite scientific discovery. This effort started from the BACON system [24], which used a rule-based production system and controlled experiments to discover empirical laws in the form of symbolic equations from data. BACON looks for simple patterns in data such as proportionality, constant, etc. The discovered patterns in each iteration are expressed in symbolic form, and used in later iterations to find more complex patterns. BACON showed promising results in discovering algebraic equations. Later on, LAGRANGE [6] extended the scope of such automatic machine discovery to differential equations. LAGRANGE used systematic hypothesis generation and linear regression to discover differential equation models for dynamical systems i.e., systems that evolve over time. In the initial stages, the hypothesis that equations are linear in system variables is



x_1	x_2	x_3	x_4	y
0.3	0.5	0.1	0.7	-.32
0.6	0.5	0.1	0.7	-.29
0.2	0.5	0.1	0.7	-.33
0.9	0.5	0.1	0.7	-.26

(a) Control x_2, x_3, x_4

x_1	x_2	x_3	x_4	y
0.6	0.1	0.8	0.4	.44
0.4	0.9	0.8	0.4	.04
0.3	0.2	0.8	0.4	.16
0.7	0.4	0.8	0.4	.40

(b) Control x_3, x_4

x_1	x_2	x_3	x_4	y
0.7	0.8	0.1	0.2	-.09
0.5	0.4	0.6	0.2	.22
0.2	0.1	0.9	0.2	.16
0.3	0.5	0.1	0.2	-.07

(c) Control x_4

x_1	x_2	x_3	x_4	y
0.2	0.4	0.2	0.7	-.24
0.9	0.3	0.5	0.5	.30
0.5	0.4	0.8	0.1	.36
0.1	0.8	0.7	0.6	-.41

(d) No control

Fig. 2. The high-level idea of Control Variable Genetic Programming (CVGP). CVGP first learns an equation mapping the single input variable to the output variable, through control variable experiments where all other variables are controlled as constants. The learning is completed using genetic programming. The learned reduced-form equation is expanded to include the second, and third input variables sequentially, using a series of control variable experiments where the remaining variables are constants. This process continues until all the input variables are introduced into the modeling equation.

explored, and in later iterations, more complex equations involving higher-order terms are explored.

Scientific discovery systems, such as BACON and LAGRANGE, focus on finding patterns in experimental data, and rely on human scientists to perform the actual experiments. Robot scientist projects like Adam [17] and Eve [16] were proposed later to make the whole discovery process autonomous, where robots take charge of designing and conducting the experiments, in addition to data analysis. Robot scientist Adam was deployed for functional genomics - the task of identifying the role of different genes in an organism. While robot scientist Eve was designed to automate the early stage of drug design and reduce the overall cost of drug discovery. Of the recent notable works for the scientific discovery of symbolic equations, AI-Feynman [33] uses a neural network as an interpolation-extrapolation system when data is scarce, and uses a series of scientific reasoning with this neural network prediction to infer symbolic equations. Since forward prediction in neural networks is much cheaper than controlled experiments, this leads to an overall reduction of the discovery process cost.

Building an AI agent that is capable of making scientific discoveries using scientific approaches requires a rapid synergy between model learning and experiment design. In this regard, BACON combines rule-based pattern matching with controlled experiments, LAGRANGE uses systematic search to reduce the need for new experiments, robot scientists Adam and Eve use actual robots to automate experiment design and execution, and AI-Feynman uses a neural network as an oracle that can mimic controlled experiments.

The aim of this book chapter is two-fold: (1) we give a historical overview of successful approaches integrating scientific approaches into AI-driven scientific discovery. (2) We report our recent endeavor in this domain with the new system of Control Variable Genetic Programming (CVGP) [15].

Our CVGP for symbolic equation discovery uses genetic programming (GP) together with scientific approaches to discover complex symbolic equations involving many variables. Previous approaches to scientific discovery using heuristic algorithms such as genetic programming are limited to simple equations with few variables, due to the large search space of candidate equations. The key contribution of our CVGP system is the introduction of the control variable experiments in the GP workflow. The high-level idea of CVGP is as follows. CVGP first discovers a simple reduced-form equation involving a subset of variables. This equation is discovered through control variable experiments where the variables outside of the subset are held as constants. Then, CVGP incrementally expands the equations discovered in previous generations, introducing new independent variables one at a time. The discovery of these equations again relies on new control variable experiments where the newly introduced variables are allowed to vary. This process continues until the final equation involving all variables is found. Intuitively, the search spaces of CVGP in the first few steps where a lot of variables are controlled as constants are significantly smaller than the full hypothesis space involving all independent variables. Moreover, the expansion step of CVGP involves introducing one variable at a time, also cheaper than identifying equations of many variables. Hence, CVGP has the potential to supercharge existing symbolic regression tools in identifying complex equations with many intertwined variables and processes.

Our work is tightly connected to the previous work which we have reviewed that implements scientific approaches for symbolic regression. While many previous approaches are rule-based (a limitation because of the computing power at the time these approaches were introduced), we used a more recent genetic programming approach for hypothesis generation and testing. GP allows us to trade-off exploration with exploitation computation - we exploit already discovered “good” equations by maintaining a large pool of candidate equations in the GP pool, and we explore the hypothesis space of the equations by randomly mating and mutating from the candidate equations in the GP pool. However, we emphasize the commonalities of our current CVGP and our reviewed previous approaches. All these systems share the same vision of using scientific reasoning principles in AI-driven knowledge discovery.

Experiment results demonstrate that scientific reasoning together with GP (i.e., CVGP) greatly improves the fitness score (i.e., normalized mean square error) of the predicted equation than the pure GP algorithm, given noiseless and noisy data. Compared to current popular deep neural network-based approaches, our CVGP also finds expressions with better fitness scores.

AI-driven symbolic regression using scientific approaches suggests that the merging of isolated branches of AI can achieve what is beyond the capabilities of the technologies of each branch alone. Traditionally, symbolic regression is considered a learning task, because the input is experimental data, and the output is a symbolic equation – a model. Nevertheless, scientific approaches, such as controlled variable experiments, hypothesis forming, validation, involve goal setting, planning, designing, deductive reasoning, which are typically not considered in learning. Philosophically, our research, and the prior work discussed in this chapter, raise a question: whether any substantial human intelligent activities, scientific discovery as one of the most sophisticated, can be achieved by one cognitive function (or one line of thought in AI research), or require the interplay of multiple functions. A healthy debate on this topic will potentially have long-lasting implications on the development of the next generation AI.

2 AI-driven Scientific Discovery using Scientific Approaches

2.1 Principles of Scientific Approaches

Scientific approaches are the backbone of scientific progress. While the precise methodology may differ across disciplines, most scientific investigations share a common structure rooted in rational inquiry and empirical validations. These fundamental components include:

1. **Problem Specification.** Scientific inquiry begins with a well-defined question or problem, often arising from unexplained phenomena, inconsistencies in existing theories, or emerging empirical patterns. The clarity of the problem defines the trajectory of the investigation and frames its potential impact.
2. **Hypothesis Formulation.** A hypothesis offers a tentative explanation or predictive model addressing the problem. It must be falsifiable and precise enough to enable systematic testing. In computational contexts, hypotheses may take the form of symbolic expressions or probabilistic assumptions encoded within algorithms.
3. **Experimental Testing.** Through controlled experimentation or computational simulation, researchers collect data to test the validity of competing hypotheses. In computational science, this step often involves large-scale simulations, synthetic data generation, or algorithmic reasoning under varied initial conditions.
4. **Data Analysis.** Collected data is analyzed to determine which hypotheses are supported or refuted. Statistical and computational tools, such as regression models, uncertainty quantification, or Bayesian inference, play an increasingly central role in making this process scalable and reproducible.

Over the years, numerous methodologies have adopted these scientific principles to automate the discovery of new knowledge. In the following section, we describe four such approaches, each illustrating how computational tools can operationalize the scientific method.

2.2 BACON – A System for the Discovery of Empirical Laws

BACON [24] is one of the first attempts at automating the scientific discovery of new knowledge from data. Named after Sir Francis Bacon, the BACON systems attempt to find hidden patterns in data by the computer program to search for scientific hypotheses. BACON expresses these patterns with simple symbolic equations. There have been 5 versions of the BACON system, each building on top of the previous release [21–23]. Later, the KEKADA [20] system followed the same idea to actively execute experiments to model the heuristics in the discovery of the urea cycle, which was a major event in biochemistry.

Production Systems. BACON is written in the production system language OPS [7] for historical reason. Before we move on to the details of how BACON works, let's first review how production system [18] languages such as OPS work. Production systems used to develop BACON consist of two basic data structures:

1. Working memory, containing a collection of symbolic data elements.
2. Production memory, containing a set of condition-action rules called productions.

The two data structures work together through a recognize-then-act cycle. This cycle contains the following distinct steps:

1. Match process, through which we find which production rule condition matches the current state of working memory.
2. Act process, which applies the action of the matched production rules on the working memory. In case multiple production rule conditions are matched, intermediate conflict resolution steps determine which actions to apply.

Let us go through a simple example, where we have two real-valued variables x_1 and x_2, stored in tabular format with 2 columns. Each row represents a tuple of (x_1, x_2) values. Using the production system, we will add another column x_3 in this table, which will indicate if x_1 and x_2 have the same sign or not. Example production rules paraphrased in simple text here can be as follows:

1. IF $sign(x_1) = sign(x_2)$, THEN write "YES" in column x_3.
2. IF $sign(x_1) \neq sign(x_2)$, THEN write "NO" in column x_3.

These rules are then applied row by row. The production rules for BACON systems are more complex than this simple example. Now that we have a basic understanding of what a production system is, let us go through the working principle of the BACON system.

Working Mechanism of BACON System. BACON mimics the human discovery process, starting with the goal of discovering the relationship between different variables of a system, collecting experimental data with controlled settings, looking for simple linear and constant relationship patterns in data, consolidating the discovered relationship with the symbolic equation, and repeating the process until a relation between all variables is found.

Let's work through an example of how BACON can discover ideal gas law from experimental data. The ideal gas law $PV = nRT$ denotes the relationship between pressure P, volume V, temperature T, and number of moles n for ideal gas. As we can see there are a good number of moving parts in this equation (except for R, the ideal gas constant, all others are variables), and given a dataset depicting P, V, n, T, it is quite challenging for a machine learning model to arrive at $PV = nRT$ relation due to overfitting.

BACON breaks this task of finding ideal gas law into a hierarchical relation-finding task. In the lowest level of this hierarchy, BACON looks for simple relations involving a subset of variables using controlled setup. As we go up this hierarchy, additional variables are added to the task, and at the top level, we get the final relationship involving all variables. A typical workflow could look like this:

1. For constant n, T, new experiments are conducted and corresponding P, V values are recorded. Upon inspecting these values, BACON finds that the product term PV is constant. BACON records this term PV for use in future iterations.
2. For constant T, new experiments are conducted and corresponding P, V, n, PV values are recorded. BACON finds that the quotient $\frac{PV}{n}$ is constant. BACON records this new relation $\frac{PV}{n}$ for use in the next iteration.
3. At the last stage, without controlling for any of the variables P, V, n, T, new experiments are conducted, and corresponding $P, V, n, PV, \frac{PV}{n}, T$ are recorded. Upon inspecting these values, BACON finds that $\frac{PV}{nT}$ is a constant approximately equal to the ideal gas constant 8.32. Therefore BACON terminates and outputs the relation $\frac{PV}{nT} = 8.32$ as the newly discovered law.

The basic principles behind BACON's success in finding empirical laws are the heuristics search process (searching for known patterns in human-discovered empirical laws, such as constant, linear relations in data) and the access to data collected using scientific approaches such as controlled experiments. It is this scientific approach behind data collection that enables BACON to infer laws from even a *small* dataset.

2.3 LAGRANGE – Scientific Discovery of Dynamic System Model

LAGRANGE [5, 6] is a system for discovering the governing model for the dynamic system. Dynamic systems change state over time, while static systems such as systems in equilibrium do not change state. LAGRANGE is able to discover differential/algebraic equations that model the evolution of a dynamic system, given the behavior trace of the system. Behavior trace of the system means the value of the system variables, measured at regular intervals over a fixed period of time. The core contribution of LAGRANGE is the extension of machine discovery to dynamic systems involving differential equations, while previous works mainly focused on static systems.

Working Mechanism of LAGRANGE. The input to the LAGRANGE system is the behavior trace of a dynamic system. LAGRANGE uses an iterative generate-and-test approach to explore the hypothesis space of all possible equations. Initially, only linear equations involving the system variables are tested via linear regression. Afterwards, new terms are introduced systematically and the more complex equations are tested via linear regression until a pre-specified limit is reached. The basic building block of the LAGRANGE algorithm consists of 3 parts:

1. **Hypothesis Formulation:** In the LAGRANGE system, hypotheses are the differential/algebraic equations that can explain the behavioral trace of the system in interest. Starting with the set of variables $S = \{x_1, x_2, \ldots, x_N\}$, LAGRANGE first computes the numerical derivatives of these variables from the observed data. New terms are then introduced by repeatedly multiplying the variables in S and their time derivatives. The order of the highest time derivative and number of multiplication steps i.e., depth, are pre-specified as hyperparameters.
2. **Hypothesis Testing:** Given the set of all variables generated in the hypothesis formulation, LAGRANGE uses linear regression to test if there exists any linear relation among any subset of the variables. Once a significant relation is found (significant according to some pre-specified criterion), this is added to the set of discovered symbolic equations. The hypothesis formulation and testing are intertwined and repeated until the pre-specified depth is reached.

Let's work through an example of how LAGRANGE works in practice. Suppose we have a biological system containing bacteria and nutrient materials. Over time, bacteria take in the nutrient materials and the concentration of nutrients x_n decreases. The concentration of bacteria x_b initially increases due to the abundance of nutrients, and then decreases as nutrient concentration depletes. The time derivatives of the two variables are noted as $\frac{\partial x_n}{\partial t}$, $\frac{\partial x_b}{\partial t}$, representing the rate of change of nutrient and bacteria concentration respectively. The change in bacteria and nutrient concentration is governed by the following partial differential equations:

$$\begin{aligned}\frac{\partial x_n}{\partial t} &= c_1 \frac{x_n}{x_n + c_2} x_b,\\ \frac{\partial x_b}{\partial t} &= \left(c_3 \frac{x_n}{x_n + c_2} - c_4\right) x_b,\end{aligned}$$

where c_1, c_2, c_3, c_4 are all constants. Given the snapshots of x_n, x_b at regular time intervals of Δt, LAGRANGE aims to discover the governing equations for $\frac{\partial x_n}{\partial t}$ and $\frac{\partial x_b}{\partial t}$.

Initially, the variable set S contains only the system variables, $S = \{x_n, x_b\}$. We assume that the maximum depth of the new terms generated by multiplication during LAGRANGE hypothesis formulation is $d = 2$, the terms x_n, x_b are at depth 1, the terms $x_n^2, x_n x_b, \ldots$ are at depth 2 and so on. The maximum size of regression variables is 3, which means that we are looking for equations containing at most $(3+1) = 4$ terms.

LAGRANGE first computes the partial derivatives $D = \{\frac{\partial x_n}{\partial t}, \frac{\partial x_b}{\partial t}\}$ numerically. The second step generates all terms of depth 2, generating the following set:

$$V = \left\{ x_n^2, x_n x_b, x_n \frac{\partial x_n}{\partial t}, x_n \frac{\partial x_b}{\partial t}, x_b^2, x_b \frac{\partial x_n}{\partial t}, x_b \frac{\partial x_b}{\partial t}, \left(\frac{\partial x_n}{\partial t}\right)^2, \frac{\partial x_n}{\partial t}\frac{\partial x_b}{\partial t}, \left(\frac{\partial x_b}{\partial t}\right)^2 \right\}.$$

After generating set V, from the combined set $(S \cup D \cup V)$, LAGRANGE considers all subsets of size 1 to 4 and performs linear regression on these subsets. These include the subsets $\{x_n \frac{\partial x_b}{\partial t}, \frac{\partial x_n}{\partial t}, x_n x_b\}$ and $\{x_n \frac{\partial x_b}{\partial t}, \frac{\partial x_b}{\partial t}, x_n x_b, x_b\}$, which generates the 2 model equations. During hypothesis testing, to judge the significance of an equation, multiple regression coefficients [35] metric are used.

2.4 Robot Scientists – Adam and Eve

The robot scientist project aims to build AI systems that can mimic human scientists – memorize available background knowledge, formulate new hypotheses, design and conduct experiments to validate/reject hypotheses, analyze experimental data, refine the hypothesis in case of rejection, and add the accepted hypothesis as new knowledge. Two systems that have been designed so far with this grand goal in mind are Adam [16] and Eve [36]. Although Adam and Eve are designed for two different applications, the string that connects both of these systems and all the future systems in this line of work, is the philosophy of using scientific approaches to make the scientific discoveries of new knowledge. Adam uses abductive reasoning to generate a new hypothesis.

Basic Workflow of Adam. Robot scientist Adam aims to identify the roles of different genes in *Saccharomyces cerevisiae* (also known as baker's/brewer's yeast) growth. To do so, Adam uses background knowledge of yeast metabolism process and scientific approach to generate and test new hypotheses about the role of unknown genes through growth experiments.

1. **Background Knowledge:** Many of the genes in *S. cerevisiae* have known functions, while many don't. The background domain knowledge of different genes' role in yeast metabolism is stored as a labeled hyper-graph in Adam, using logic programming. The metabolites (small molecules) are represented as nodes while the arcs represent enzymes the facilitate transformation of one molecule to another. An arc labeled E, going from node A to B represents that A can be transformed into B molecules using enzyme E.
2. **Hypothesis Formulation:** The goal of Adam is to predict missing labels/edges in the hyper-graph of yeast metabolism. This prediction task is equivalent to identifying the genes responsible for encoding the enzymes that catalyze biochemical reactions in yeast. Adam uses abductive reasoning to predict possible missing labels/edges in the metabolism graph. The missing

information can then be used to explain the observed experimental results. The hypotheses that fail to explain observed results should be discarded, while the successful ones are accepted as new knowledge.

3. **Experiment Design:** Adam uses controlled experiments for hypothesis testing. Once a gene is identified as a potential candidate encoding an enzyme, two samples – one with the gene and one without the genes are grown in similar conditions. The correctness of the hypothesis can then be measured using the difference in growth of these two samples.
4. **Data Analysis:** Adam uses machine learning techniques to analyze experimental data and relate the findings to the hypotheses. Specifically, decision trees and random forests with resampling methods were used to decide whether the yeast growth in different media, controlled for the hypothesized gene was significant. Classical statistical methods were used to verify the results manually.

Adam formulated and tested 20 hypotheses about genes encoding 13 different enzymes, and 12 of these hypotheses were confirmed with statistically significant observations. Later these were tested with standard experiments by human scientists. While robot scientist Adam focused on automating the full process of functional genomics, the other robot scientist Eve was designed to automate the early stage of drug discovery. However, the working philosophy for Eve was the scientific approach of hypothesis generation and testing, the same as that of Adam.

Basic Workflow of Eve. Robot scientist Eve was designed with the purpose of making the drug discovery process cheaper and faster. Specifically, Eve aims to automate the early stage of drug design - the identification of a possible "lead" compound to be used as a drug, using automated scientific experiments. Given a set (library) of potential compounds for a drug, Eve performs the following steps to identify possible lead compounds:

– **Background Knowledge:** In Eve, background knowledge is represented using graph and chemoinformatic methods. Unlike robot scientist Adam, background knowledge exploitation is largely conducted by human experts. Human scientists use standardized synthetic biology experiments to design appropriate experiments (also called 'assay'), which is then used by Eve.
– **Hypothesis Formulation:** Eve's hypothesis takes the form of a quantitative structure-activity relationship (QSAR). Given the structure of a compound, QSAR predicts how a compound will perform on experimental evaluation. Eve uses inductive reasoning, specifically ridge regression to form QSARs. This is equivalent to the Gaussian process with a linear kernel, which enables the computation of posterior uncertainty.
– **Hypothesis Testing:** Eve uses active learning to test new hypotheses. The active learning task here is to choose which compound to test from a library of available compounds. Eve uses a reward function combining high estimated activity and high estimated variance for each compound and selects the compounds with the highest rewards for testing.

On real-world deployment, Eve was able to identify that the anti-cancer compound TNP-470 is a potent inhibitor of dihydrofolate reductase from the malaria-causing parasite *Plasmodium vivax*. This discovery was later validated by additional testing. The repositioning of drugs – the application of known drug compounds to new diseases is of high potential value, and thus Eve was able to demonstrate that AI methods fueled by scientific principles have the potential to greatly accelerate the drug discovery pipeline.

2.5 AI-Feynman – Combining Physics Knowledge and Neural Networks for Equation Discovery

AI-Feynman [32, 33] systems aim to find symbolic equations from static datasets. Named after famous physicist Richard Feynman, the AI-Feynman systems take inspiration from known physics laws. Given a set of data points $(x_1, x_2, \ldots, x_n, y)$ where $y = f(x_1, x_2, \ldots, x_n)$, the goal of the AI-Feynman system for symbolic regression is to find unknown function f. Such function f can be extremely complicated, and the possible number of such functions is exponentially large. However, in physics and other science domains, such functions exhibit some simplifying properties:

- Physical unit property. The variables x_i and f have known physical units.
- Low-order property. The entire function f of part of it is polynomial in a subset of variables $\{x_1, x_2, \ldots, x_n\}$.
- Compositional property. The function f can be decomposed as a composition of simpler functions.
- Smoothness property. The function f is continuous and smooth.
- Symmetry property. The function f is symmetrical with respect to a subset of variables $\{x_1, x_2, \ldots, x_n\}$.
- Separability property. The function f can be separated into 2 parts with a disjoint set of variables, connected via addition/multiplication.

AI-Feynman systems look for these simplifying properties in the data and discover the symbolic relationship f step-by-step.

Unlike BACON which assumes that new experimental data can be acquired under a control setup, AI-Feynman assumes a static dataset. To probe data points that are not present in the static dataset, AI-Feynman uses a neural network to perform interpolation/extrapolation.

Working process of AI-Feynman. AI-Feynman applies the following procedures in sequence:

1. **Dimension Analysis:** Every variable in a symbolic equation has physical units, and the units on both sides of the equation must match. Using this idea, this step tries to transform the problem of finding an equation of n variables into a problem involving less than n variables.

2. **Polynomial Fitting:** Many physics systems can be represented as low-order polynomials. In this step of AI-Feynman, low-order polynomials up to degree 4 are tried exhaustively to find a symbolic equation.
3. **Brute Force Search:** The number of possible symbolic equations is exponential in the equation length (the number of symbols in the equation). When the equation length is small, a brute force method to try all possible equations can be feasible. In this step of AI-Feynman, a brute force approach is applied to try to fit equations up to a certain length and find the best possible one. In cases where multiple equations fit the data, the equation with the lowest root mean squared error (RMSE) within a bound is selected.
4. **Neural network-based Tests and Transformations:** This step of the AI-Feynman workflow is the second core contribution of this method, the first one is the heuristics obtained from physics knowledge. Scientific experiments can be costly both in terms of time and effort. In the absence of data from scientific experiments, scientific models suffer from errors. To bridge this gap of data scarcity, AI-Feynman uses a neural network to extrapolate/interpolate missing data from available data points. Once this neural network is trained, scientists can make scientific queries and generate new data to test hypothesis models.

AI-Feynman uses cutting-edge deep learning techniques to reduce the cost of scientific experiments for finding symbolic equations from data. At the core of the workflow, the algorithm still relies on scientific approaches for making new knowledge discoveries.

3 Control Variable Genetic Programming for Symbolic Equation Discovery

Control Variable Genetic Programming (CVGP) is a modern computational framework designed to accelerate symbolic equation discovery from experimental data by incorporating scientific reasoning—specifically through control variable experiments—into traditional equation discovery methods. Our CVGP approach scales effectively to discover complex algebraic and differential equations involving many interacting variables. In the following section, we outline the core principles of CVGP and present empirical evidence demonstrating its effectiveness in symbolic equation discovery.

3.1 Motivation

Since the inception of BACON system, there have been many exciting works in this direction of symbolic equation discovery [30, 34, 10, 26, 25, 26, 29, 28, 14]. The use of learning algorithms such as genetic programming and neural networks are of special mention here. Despite such progress, the current state-of-the-art methods are limited to learning very simple symbolic equations. The main reason is that the hypothesis space of the symbolic equation is exponential in the length of the equation (the length of the equation is measured in terms of the number of operands and operators in the equation). Searching the exponentially large space of possible equations has shown to be an NP-hard problem. Therefore, we need scalable methods that can automate the discovery of complex symbolic equations involving many interrelated variables, which are prevalent in physical science domains.

3.2 Preliminaries

Control Variable Experiment. A control variable experiment takes in several input arguments – a hypothesis equation ϕ, a list of free variables $\mathbf{v}_f$, a list of controlled variables $\mathbf{v}_c$ and the total number of control variable trials K. Out of all variables in $\mathbf{v}_f$, we designate one as the output variable and measure its values in the varying setup of other variables. The constants of the equation ϕ if present, are calculated by fitting the equation to the training dataset.

In each trial, we generate and collect a dataset by keeping the controlled variables in $\mathbf{v}_c$ to constant values, and randomly varying variables in $\mathbf{v}_f$ and measuring the output variable. In the real world, this is done by conducting scientific experiments, while in our case, we do this by using a data oracle that mimics the output of scientific experiments. The data oracle simulates the ground truth equation and yields noisy output, similar to real-world experiments.

Assuming there is a ground-truth equation that governs the output of such experiments, during the controlled setup, we actually observe simulation from a reduced form of the ground-truth equation. In this reduced-form equation, subexpressions involving the controlled variables appear as constants. Depending on the control setup, the value of these constants may vary. In each trial, we record two critical information:

1. **Fitness score of the hypothesis equation:** This is done by defining a reward function such as the inverse mean squared error. The reward function measures how well the hypothesis equation can explain the generated controlled dataset in the trial.
2. **Summary constants of the equation:** We also record the value of the constants in the hypothesis equation. The variability of these constants across multiple trials gives us an idea about whether a constant depends on the controlled variables or not.

Genetic Programming for Symbolic Regression. Genetic programming (GP) is a heuristic algorithm inspired by evolutionary phenomena such as selection, mutation, crossover, etc. Initially, we start with a random set of possible candidate equations i.e., a population. We define criteria to test how good the equations are. An example criterion can be a loss function that penalizes the mismatch between equation evaluation and observation data. After this initialization, we repeatedly apply 3 basic steps to the population:

1. **Selection.** Keep the best candidate in the population according to test criteria.

2. **Mutation.** Randomly pick equations, and change one of their subexpressions randomly.
3. **Crossover.** Randomly pick a pair of equations, and exchange subexpressions to form a pair of new equations.

In GP, we repeat these 3 procedures repeatedly for a number of fixed iterations and keep track of the best equations in each iterations. At the end of the final iterations, the set of best equations is returned as the final output of GP.

3.3 Working Mechanism of Control Variable Genetic Programming

Our control variable genetic programming (CVGP) algorithm integrates scientific reasoning in the form of control variable experiment trials, and randomization in the form of genetic programming to scale up the scientific discovery of symbolic equations. The basic building blocks of the CVGP framework are as follows:

1. **Background Knowledge:** CVGP assumes the mathematical operators (e.g., addition, multiplication, etc.) and the variables (given as part of the dataset for testing candidate equations) are known.
2. **Hypothesis Formulation with Control Variable Experiment:** Hypotheses in CVGP are the possible symbolic equations that can explain a given dataset. In CVGP, control variable experiment trials combined with genetic programming, are used for hypothesis formulation. Lets assume there are N variables in a dataset $x_1, x_2, \ldots, x_N$. Instead of trying out equations involving all N variables, CVGP will first generate and test equations involving x_1, and control the rest of the variables to be constants. Once an equation is discovered, CVGP will use this equation to generate new candidate equations involving x_1, x_2 only, controlling the rest of the variables. In the final iteration of CVGP, the generated equations will involve all N variables with no control.
3. **Hypothesis Testing:** Once the structure of the hypothesis is formed, the constants of the hypothesis equations if present, are calculated using gradient-based optimization to minimize difference with a dataset. The dataset is generated with a control variable experiment setup. To test how well the hypothesis equations can describe the dataset, a loss function such as mean squared error can be used. The hypothesis with the minimum loss functions is retained for the next iterations.

We see the outline of CVGP in Algorithm 1. Initially, all variables are added to the control set (line $1-2$). A set of random equations constitutes the initial population for candidate equations (line 3). The parameter set θ includes many hyperparameters of genetic programming such as the probability of mutation and crossover, the size of the hall-of-fame set, which keeps track of the best equations across GP iterations, etc. After fixing the order of the variables, we move one variable at a time from the control set to the free set (line $5-6$). For each equation

Algorithm 1 Control Variable Genetic Programming (CVGP)

```
Input: Set of variables $\{x_1, x_2, \ldots, x_N\}$; Number of control variable trials $K$; Size $M$
    of the best equation set $\mathcal{H}$.
Output: The best-fitted expressions.
 1: $\mathbf{v}_c \leftarrow \{x_1, \ldots, x_N\}$;
 2: $\mathbf{v}_f \leftarrow \emptyset$;
 3: $\mathcal{P}_{gp} \leftarrow$ CreateInitialPool();
 4: for $x_i \in \{x_1, \ldots, x_N\}$ do
 5:     $\mathbf{v}_c \leftarrow \mathbf{v}_c \setminus \{x_i\}$;
 6:     $\mathbf{v}_f \leftarrow \mathbf{v}_f \cup \{x_i\}$;
 7:     for $\phi \in \mathcal{P}_{gp}$ do
 8:         $\phi$.constants, $\phi$.score $\leftarrow$ ControlVariableExperiment($\phi, \mathbf{v}_c, \mathbf{v}_f, K$);
 9:     $\mathcal{P}_{gp} \leftarrow GP(\mathcal{P}_{gp}, \mathbf{v}_c, \mathbf{v}_f)$;
10:     for $\phi \in \mathcal{P}_{gp}$ do
11:         FreezeEquation($\phi$);
12: $\mathcal{H} \leftarrow$ Top $M$ equations of $\mathcal{P}_{gp}$ according to fitness score;
13: return $\mathcal{H}$

    Helper Procedures
14: procedure $GP(\mathcal{P}_{gp}, \mathbf{v}_c, \mathbf{v}_f)$
15:     Generate new equations from $\mathcal{P}_{gp}$ by selection,mutation and crossover;
16:     Use only the variables in set $\mathbf{v}_f$ during mutation and crossover process;
17:     return The set of all equations.
18: procedure ControlVariableExperiment($\phi, \mathbf{v}_c, \mathbf{v}_f, K$)
19:     for $k \in \{1, \ldots, K\}$ do
20:         $T_k \leftarrow$ Generate datasets by controlling $\mathbf{v}_c$ and varying $\mathbf{v}_f$;
21:         $C_k \leftarrow$ Find optimum constants in $\phi$ using $T_k$;
22:         $F_k \leftarrow$ Find fitness score of $\phi$ using $T_k$;
23:     return $\{C_k\}_{k=1}^{K}$,$\{F_k\}_{k=1}^{K}$.
24: procedure FreezeEquation($\phi$)
25:     Except for constants, mark the operands of equation $\phi$ as immutable;
26:     For constants of $\phi$ that vary little across trials, mark them as immutable;
```

in the population, we perform K trials of the control variable experiment and record the constants and fitness scores (line $7-8$). We use genetic programming to update the pool of candidate equations (line 9). For every equation, if a constant in the equation shows little variability across multiple trials of the control variable experiment, we denote them as independent of the controlled variables and mark them immutable for later generations of GP (line 11). Finally, we rank all the candidate equations of $\mathcal{P}_{gp}$ according to fitness score, and return the set $\mathcal{H}$, the set of top-ranked equations (line 12).

Example Discovery of the Ideal Gas Law. Using our CVGP algorithm, an example of successful scientific discovery of empirical laws such as ideal gas law $PV = nRT$ depicting a relationship between pressure P, volume V, number of moles of gas n and temperature T, can take the following course of actions:

1. CVGP first fix a library of mathematical operators – addition, subtraction, multiplication, and division and 4 variables P, V, n, T, which will be used to generate hypothesis equations. P is chosen as the output variable which will be measured after varying the other variables.
2. Controlling V (volume), n (gas amount), and T (temperature) as constants, CVGP generates hypothesis equations involving pressure P only, and eventually finds the equation $P = c_1$, where c_1 indicates a constant coefficient. The value of constant c_1 is calculated using gradient-based optimization to minimize the difference between equation output and observed data.
3. In the next iteration, CVGP will control only n and T, and generate equations by looking for replacement of c_1 in equation $P = c_1$. Eventually, CVGP finds the equation $P = \frac{c_2}{V}$, where c_2 is a constant.
4. In the third iteration, CVGP will control only n, and generate hypothesis equations by replacing the constant c_2 in $P = \frac{c_2}{V}$, which will lead to the discovery of $P = \frac{(c_3 T)}{V}$, where c_3 is a constant.
5. In the next and final iteration, CVGP does not control any variable and generates hypothesis equations by replacing constant c_3 in the equation $P = \frac{(c_3 T)}{V}$, and eventually finds the equation $P = \frac{(c_4 n) T}{V}$. The value of constants c_3 and c_4 are calculated using gradient-based optimization, similar to c_1, c_2, and the final value of c_4 is found to the ideal gas constant $R = 8.31$.

3.4 Experimental Evaluation

Experiment Settings. [1] We consider the following baselines based on evolutionary algorithms: 1) Genetic Programming (GP) [8]. We also consider a series of baselines using reinforcement learning: 2) Priority queue training (PQT) [1]. 3) Vanilla Policy Gradient (VPG) that uses the REINFORCE algorithm [37] to train the model. 4) Deep Symbolic Regression (DSR) [26]. 5) Neural-Guided Genetic Programming Population Seeding (GPMeld) [25].

The ground-truth equations we consider are multi-variable polynomials characterized by their operands 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, such as x_1, or a unary operand on a variable, such as $\sin(x_1)$. 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 operands 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.

We mainly consider the goodness-of-fit measure (NMSE) for the learning algorithms tested in our work, which indicates how well the learning algorithms perform in discovering symbolic expressions. The full quartiles of the NMSE

[1] All code available publicly at https://github.com/jiangnanhugo/cvgp

are reported. Given a separated testing dataset $D_{\text{test}} = \{(\mathbf{x}_i, y_i)\}_{i=1}^m$ generated from the ground-truth expression, we measure the goodness-of-fit of a predicted expression $\bar{\phi}$, by evaluating the normalized-mean-squared error (NMSE):

$$\text{NMSE} = \frac{1}{m\sigma_y^2}\sum_{i=1}^{m}(y_i - \bar{\phi}(\mathbf{x}_i))^2 \tag{1}$$

where the empirical variance $\sigma_y^2 = \frac{1}{m}\sum_{i=1}^{n}\left(y_i - \frac{1}{m}\sum_{i=1}^{m} y_i\right)^2$.

Learning Result. The CVGP method attains the smallest NMSE values among all the baselines when evaluated on noisy datasets (Figure 3). This shows our method can better handle multiple variables symbolic regression problems than the current best algorithms in this area.

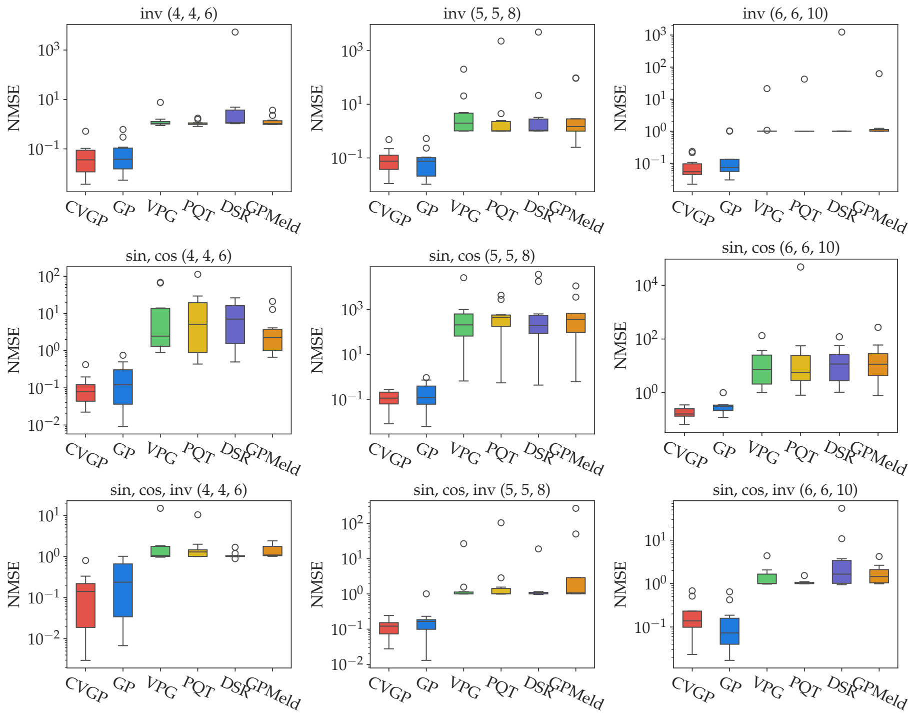


Fig. 3. Quartiles of Normalized Mean squared error (NMSE) values of all the methods over several *noisy* datasets. Gaussian noise with zero mean and standard deviation 0.1 is added. CVGP shows a consistent improvement over all the baselines considered, among all the datasets. The reason is because of the introduction of scientific reasoning, i.e., the control variable experiment.

4 Related Works for Symbolic Equation Discovery

There have been a vast amount of contemporary research works for the automatic discovery of symbolic equations besides the already discussed works in this chapter. Many of the algorithms for symbolic equation discovery are based on evolutionary algorithms like genetic programming [19, 33, 34, 14]. Most notable in this line of research is the commercially available software Eureqa [4]. Recently, reinforcement learning-based methods for symbolic equation discovery has garnered much attention [26, 25, 3]. Active learning, which considers the query data point for maximum learning performance have also been used for symbolic equation discovery [11, 9, 12, 13].

In terms of similarity, our CVGP method is most similar to the BACON system [24] as both of them use control variable experiments. Compared to the benefit of heuristic knowledge provided by BACON, our CVGP algorithm provides the exploration-exploitation trade-off in searching the huge hypothesis space of possible symbolic equations. CVGP also saves in time and space in equation discovery as the equations discovered in different controlled settings are reused.

Limitations of current symbolic equation discovery algorithms: Our Control Variable Genetic Programming (CVGP) approach and other symbolic equation discovery approaches discussed in this chapter assume that all relevant and irrelevant variables are present in the data. If important variables are missing or have been erroneously excluded due to incorrect assumptions, the performance of these methods, including ours, can degrade significantly or fail altogether.

Furthermore, in systems where input variables exhibit complex, confounding, or unpredictable interactions, standard control-variable experiments, a key novelty of our CVGP algorithm, become inapplicable. Future research may consider first identifying and modeling the confounding interactions between variables and then partitioning them into coherent groups. Control-variable experiments can then be applied at the group level, rather than on individual variables, to better account for interdependencies and improve robustness in complex settings.

5 Conclusion

In this book chapter, we have provided a summary of the key milestones in the history of scientific approach-based symbolic equation discovery. We started with BACON, a production-system-based equation discovery algorithm, which was a pioneer in this domain. We then moved on to LAGRANGE – which extended the equation discovery process to differential equations. We described Robot scientists Adam and Eve, which aim to make the whole scientific discovery process autonomous by minimizing human intervention in physical experiments. We then described our own work, control variable genetic programming (CVGP), which integrates scientific reasoning and randomization for scalable scientific discovery of symbolic equations involving many interrelated variables. Aside from these frameworks, there is an abundance of research works which aim to automate scientific discovery using computational systems.

The growing reliance on computational systems to automate elements of the scientific process prompts critical reflection on the nature and limitations of machine-guided discovery. Unlike humans, computational systems lack intuition, creativity, and broader contextual understanding—traits that human scientists are good at. Instead, these systems excel in exhaustive search, pattern recognition, and probabilistic inference, offering a powerful but incomplete perspective on scientific discovery. In this book chapter, we have chosen to highlight computation approaches with integrated scientific reasoning in some form. This represents a line of effort to enhance machine systems with intuition and creativity-based active exploration. It also demonstrates that the merging of several branches of AI may result in novel capabilities that are beyond the approaches in each branch alone. We believe that scientific reasoning is a key to automating the discovery of new knowledge from a huge volume of experimental data, and this chapter will serve as a helpful guide to interested readers who are interested in pursuing future works in this direction.

Acknowledgement

This research was primarily supported by NSF grant CCF-1918327 and IIS-2339844. We also acknowledge financial support from NSF grant IIS-1850243 and DOE – Fusion Energy Science, under grant number: DE-SC0024583.

References

1. Abolafia, D.A., Norouzi, M., Le, Q.V.: Neural program synthesis with priority queue training. CoRR **abs/1801.03526**, 1–16 (2018)
2. Boyle, R., Sharrock, R.: A Defence of the Doctrine Touching the Spring and Weight of the Air. F.G. for Thomas Robinson (1662)
3. 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)
4. Dubčáková, R.: Eureqa: software review (2011)
5. Džeroski, S., Todorovski, L.: Discovering dynamics. In: Proc. tenth international conference on machine learning. pp. 97–103 (1993)
6. Dzeroski, S., Todorovski, L.: Discovering dynamics: From inductive logic programming to machine discovery. J. Intell. Inf. Syst. **4**(1), 89–108 (1995)
7. Forgy, C., McDermott, J.P.: Ops, a domain-independent production system language. In: IJCAI. vol. 5, pp. 933–939 (1977)
8. 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)
9. Golovin, D., Krause, A., Ray, D.: Near-optimal bayesian active learning with noisy observations. In: NIPS. pp. 766–774. Curran Associates, Inc. (2010)
10. 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)

11. Hanneke, S.: Theory of disagreement-based active learning. Found. Trends Mach. Learn. **7**(2-3), 131–309 (2014)
12. Haut, N., Banzhaf, W., Punch, B.: Active learning improves performance on symbolic regression tasks in stackgp. In: GECCO Companion. pp. 550–553. ACM (2022)
13. Haut, N., Punch, B., Banzhaf, W.: Active learning informs symbolic regression model development in genetic programming. In: GECCO Companion. pp. 587–590. ACM (2023)
14. He, B., Lu, Q., Yang, Q., Luo, J., Wang, Z.: Taylor genetic programming for symbolic regression. In: GECCO. pp. 946–954. ACM (2022)
15. Jiang, N., Xue, Y.: Symbolic regression via control variable genetic programming. In: Koutra, D., Plant, C., Gomez Rodriguez, M., Baralis, E., Bonchi, F. (eds.) Machine Learning and Knowledge Discovery in Databases: Research Track. pp. 178–195. Springer Nature Switzerland, Cham (2023)
16. 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)
17. 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)
18. Klahr, D., Langley, P., Neches, R.: Production system models of learning and development. MIT press (1987)
19. Koza, J.R.: Genetic programming as a means for programming computers by natural selection. Statistics and computing **4**, 87–112 (1994)
20. Kulkarni, D., Simon, H.A.: The processes of scientific discovery: The strategy of experimentation. Cognitive Science **12**(2), 139–175 (1988)
21. Langley, P.: BACON: A production system that discovers empirical laws. In: IJCAI. p. 344. William Kaufmann (1977)
22. Langley, P.: Rediscovering physics with BACON.3. In: IJCAI. pp. 505–507. William Kaufmann (1979)
23. Langley, P., Bradshaw, G.L., Simon, H.A.: BACON.5: the discovery of conservation laws. In: IJCAI. pp. 121–126. William Kaufmann (1981)
24. Langley, P.W., Simon, H.A., Bradshaw, G., Zytkow, J.M.: Scientific Discovery: Computational Explorations of the Creative Process. The MIT Press (02 1987)
25. 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)
26. 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)
27. Radvanyi, P.: Sur la combinaison des substances gazeuses, les unes avec les autres. Bibnum. Textes fondateurs de la science (2009)
28. Razavi, S., Gamazon, E.R.: Neural-network-directed genetic programmer for discovery of governing equations. CoRR **abs/2203.08808** (2022)
29. Scavuzzo, L., Chen, F.Y., Chételat, D., Gasse, M., Lodi, A., Yorke-Smith, N., Aardal, K.I.: Learning to branch with tree mdps. In: NeurIPS (2022)
30. Schmidt, M., Lipson, H.: Distilling free-form natural laws from experimental data. Science **324**(5923), 81–85 (2009)
31. Simon, H.A.: Spurious correlation: A causal interpretation. Journal of the American statistical Association **49**(267), 467–479 (1954)

32. Udrescu, S.M., Tan, A., Feng, J., Neto, O., Wu, T., Tegmark, M.: Ai feynman 2.0: Pareto-optimal symbolic regression exploiting graph modularity. Advances in Neural Information Processing Systems **33**, 4860–4871 (2020)
33. Udrescu, S.M., Tegmark, M.: Ai feynman: A physics-inspired method for symbolic regression. Science Advances **6**(16), 1–16 (2020)
34. 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)
35. Volk, W.: Applied Statistics for Engineers. McGraw-Hill series in chemical engineering, McGraw-Hill (1969)
36. Williams, K., Bilsland, E., Sparkes, A., Aubrey, W., Young, M., Soldatova, L.N., De Grave, K., Ramon, J., De Clare, M., Sirawaraporn, W., et al.: Cheaper faster drug development validated by the repositioning of drugs against neglected tropical diseases. Journal of the Royal society Interface **12**(104), 20141289 (2015)
37. Williams, R.J.: Simple statistical gradient-following algorithms for connectionist reinforcement learning. Mach. Learn. **8**, 229–256 (1992)