Explainable Artificial Intelligence by Genetic Programming: A Survey

Yi Mei, *Senior Member, IEEE,* Qi Chen, *Member, IEEE,* Andrew Lensen, *Member, IEEE,* Bing Xue, *Senior Member, IEEE,* and Mengjie Zhang, *Fellow, IEEE*

***Abstract*—Explainable artificial intelligence has received great interest in the recent decade, due to its importance in critical application domains such as self-driving cars, law and healthcare. Genetic programming is a powerful evolutionary algorithm for machine learning. Compared with other standard machine learning models such as neural networks, the models evolved by GP tend to be more interpretable due to their model structure with symbolic components. However, interpretability has not been explicitly considered in genetic programming until recently, following the surge in popularity of explainable artificial intelligence. This paper provides a comprehensive review of the studies on genetic programming that can potentially improve the model interpretability, both explicitly and implicitly, as a byproduct. We group the existing studies related to explainable artificial intelligence by genetic programming into two categories. The first category considers the *intrinsic interpretability*, aiming to directly evolve more interpretable (and effective) models by genetic programming. The second category focuses on *post-hoc interpretability*, which uses genetic programming to explain other black-box machine learning models, or explain the models evolved by genetic programming by simpler models such as linear models. This comprehensive survey demonstrates the strong potential of genetic programming for improving the interpretability of machine learning models and balancing the complex trade-off between model accuracy and interpretability.**

I. Introduction

Genetic Programming (GP) [1], [2] is a powerful evolutionary algorithm for automated programming. It has been successfully applied to a variety of machine learning tasks [3] such as classification [4], [5], regression [6], [7], clustering [8], [9], reinforcement learning [10]–[13] and automatic heuristic design [14], [15]. The evolved GP models can make accurate predictions or make effective decisions.

There has been extensive research into improving the performance of GP from various perspectives such as the *test performance* (e.g., prediction accuracy on unseen test data) and *training efficiency* (i.e., training time and convergence speed). Recently, improving the *explainability or interpretability* of machine learning models through GP has attracted heightened research focus as the importance of eXplainable Artificial Intelligence (XAI) continues to increase.

There are several reasons to consider explainability/interpretability of machine learning [16]. First, it is important to be able to identify and avoid "Clever Hans" predictors, which exploit tricks and misuse the coincidental patterns that happened to exist in the training data but cannot generalise [17]. A good example is the image classifier that learns to distinguish husky dogs and wolves in images by checking for the presence of snow in the background of images, simply because the training data consisted of many images of wolves in the snow [18]. Second, in many applications, preserving the trust and confidence in a machine learning system is paramount. For example, an automatic bank loan approval system must be able to provide the reasons for declining a loan application to satisfy the applicant and ensure fairness and transparency. It was horrifying to find that the COMPAS system makes racially-biased recidivism predictions [19]. In other areas where safety and reliability are non-negotiable, such as self-driving cars and manufacturing industries, explanations can help predict the mistakes the system may make, avoiding disastrous accidents. Third, there are increasing legislative requirements for explainability. For instance, the European Union's General Data Protection Regulation (GDPR) [20] requires that machine learning models must be able to be explained. For example, the use cases of credit scoring and medical imaging are used in [21] to highlight the challenge for machine learning models to meet the GDPR requirements. Finally, explanations can provide new insights and discover new knowledge beyond those found by human scientists. In the Go competition between AlphaGo and Lee Sedol, the AlphaGo made a move that appeared useless to professional human players but turned out to be critical to the final win. Explaining such a move would provide new insights into the strategy and structure of the game. Scientists not only want to achieve good performance but also want to know "why" and "how"; from an engineering point of view, users want to understand how solutions/models function step-by-step to complete a (complex) task. XAI research has seen a rapid growth in recent years, with a range of XAI survey papers [22]–[27]. However, these papers mainly focus on the general concepts of XAI or on specific techniques in deep neural networks.

As a major evolutionary machine learning approach, GP has made significant contributions to the AI and machine learning

This work was supported in part by the Marsden Fund of New Zealand Government under Contracts VUW1913 and VUW1914, the Science for Technological Innovation Challenge (SfTI) fund under grant E3603/2903, the Victoria University of Wellington Research grant number 225233/4077, 223805/3986, 226161/4164 and 410128/4223, MBIE Data Science SSIF Fund under the contract RTVU1914, National Natural Science Foundation of China (NSFC) under Grant 61876169, Huayin Medical under grant E3791/4165, and MBIE Endeavor Research Programme under contract C11X2001.

Yi Mei, Qi Chen, Andrew Lensen, Bing Xue and Mengjie Zhang are with the Evolutionary Computation and Machine Learning Research Group, School of Engineering and Computer Science, Victoria University of Wellington, Wellington 6140, New Zealand (e-mail: yi.mei@ecs.vuw.ac.nz; qi.chen@ecs.vuw.ac.nz; andrew.lensen@ecs.vuw.ac.nz; bing.xue@ecs.vuw.ac.nz; mengjie.zhang@ecs.vuw.ac.nz).

fields, and has a great potential to contribute to XAI. With its symbolic nature, GP, with tree-based [2], linear [28], [29] or graph [30], [31] representation, has an inherent interpretability that is consistent with how a human understands the structure of a computer program. Moreover, strongly-typed GP [32] and the use of grammar [33], [34] constrain the model structure and avoid the less meaningful programs so that the obtained GP models tend to be more interpretable.

Note that there are other inherently interpretable models such as decision trees and rule sets. Typically, these models are built by some greedy heuristics. For example, the well-known C4.5 algorithm [35] builds the decision tree recursively. For each node, it selects a feature that can best distinguish the training data (e.g., with the highest normalised information gain). In contrast, GP has a higher flexibility, and can learn more accurate models by searching in a richer model space. For example, GP has been successfully used to learn decision trees [36]–[39], rule sets [40]–[42] and fuzzy pattern trees [43]–[47]. Overall, GP is a promising XAI technique due to its inherent interpretability and capability to better learn other interpretable machine learning models.

However, this natural/built-in interpretability is not enough when addressing complex problems, and the evolved GP models are typically too large to be interpreted. Therefore, it is necessary to factor interpretability into the design of GP algorithms, in order to learn GP models that not only perform useful tasks (e.g., prediction, recognition, and decision making) but can also explain their decision-making. This enables users to understand the inner mechanism of the model, its strengths and weaknesses, when and in which situations it may make mistakes, and even how to further improve it.

There have been many studies in GP that consider improving explainability and interpretability. Recent work has *explicitly* considered explainability, whereas older work *implicitly* improved explainability as a byproduct of improved search/learning algorithms. However, so far, there has been no survey that summarises the progress in XAI by GP and identifies challenges and opportunities in the field.

This paper fills this gap by providing the first survey on achieving better machine learning model interpretability through GP. The major contributions of this paper are as follows:

- For GP researchers, this paper provides a summary of how to improve the explainability and interpretability of the evolved GP models.
- For general AI and machine learning researchers, this paper demonstrates the potential to achieve better XAI by using a GP approach.
- For practitioners with their real-world prediction and decision making tasks, this paper can provide interpretable solutions (i.e., models obtained by GP-based approaches) for solving their tasks.

We performed a comprehensive literature review by searching the databases of IEEE Xplore, ACM Digital Library, Elsevier, Springer, and the GP bibliography (http://gpbib.cs.ucl.ac.uk/). We used the keywords of "genetic programming", "explainability", "interpretability", "explainable", "interpretable", as well as related keywords, including "program size", "bloat", "model complexity", "feature selection", "simplification", "grammar" and "decision tree".

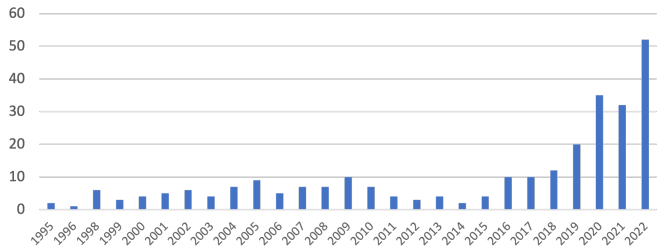

Fig. 1. The number of publications related to XGP over the years since 1995.

For the sake of convenience, we will call the research direction of achieving better eXplainable machine learning models by GP as XGP. Fig. 1 shows the number of publications related to XGP over the years from 1995. The figure shows a clear and rapid increase in the number of publications over the recent five years, demonstrating the current growth of the XGP research direction.

The rest of the paper is organised as follows. Section II introduces relevant background, including fundamental XAI concepts and a taxonomy of XGP approaches. Then, Section III reviews the *intrinsic* XGP approaches that directly take explainability into account during the training for more explainable GP models. Section IV describes the *post-hoc* XGP approaches that improve interpretability at the end of the learning process. This includes both approaches of evolving interpretable GP models that explain other pre-trained black-box models; and using simple interpretable models such as linear regression to explain a pre-trained complex GP model. Section V shows how visualisation techniques can be used to improve GP interpretability and how GP itself can produce interpretable visualisations. Section VI summarises the existing real-world applications of interpretable GP. Section VII details challenges and possible future directions for XGP. Finally, the paper is concluded in Section VIII.

II. Background

In this section, we first describe the important concepts and terms for XGP and the position of XGP within the general XAI area. Then, we provide the taxonomy of the XGP approaches.

A. Important Concepts

There have been a variety of similar concepts and terms for explanations used in the existing literature, such as *understandability*, *comprehensibility*, *interpretability*, *explainability*, and *transparency*. Barredo et al. [26] gave the definitions of these terms to distinguish between them. Lipton [48] described different levels of transparency in machine learning models: *simulatability*, which reflects how well the model simulates the human thinking process; *decomposability*, which indicates how well the model can be decomposed to smaller human-understandable modules; and *algorithmic transparency*, which expresses if the user can understand the process of the model from taking a possible input to returning the final output.

Despite the subtle differences in their definitions, the above terms — especially explainability and interpretability — are often used interchangeably in the literature. As this paper is focused on surveying the breadth of XGP *techniques* rather than XAI concepts, we too use explainability and interpretability interchangeably in this paper.

Defining the criteria to evaluate the quality of an explanation is still an open research area, and it involves considerations from various fields outside of computer science, such as psychology, cognitive science, and philosophy [22], [49]. Miller [50] considered the qualities of a good explanation from a social science perspective. For example, he argued that a good explanation should be *contrastive* ("*Why A rather than B*"), *selected* (focusing on only one or two causes/features rather than all the possible causes/features) and *social* (e.g., presented as a conversation). In addition, he suggested a good explanation should not have associated probabilities. Doshi-Velez and Kim [49] gave a taxonomy of interpretability evaluation, which consists of the following three levels:

1) **Application-grounded**: where a human directly evaluates the explanation on a specific task that the model is trained for (e.g., medical diagnosis, face recognition), often using visualisation and human-computer interaction. For example, surveying doctors as to how easily they can understand the explanation produced by an AI diagnostic system.
2) **Human-grounded**: conducting human experiments to evaluate the explanation. For example, in a forward simulation, given an explanation and an input, it checks how well humans can replicate the output of the simulation based on the explanation. In the counterfactual simulation experiment, given an explanation, an input and an undesired output, humans are asked how they could alter the input to change the prediction to the desired output. This is essentially evaluating the model *simulatability* defined in [48].
3) **Functionally-ground**: designing proxy metrics (e.g., model representation and complexity) of the model that attempt to reflect its interpretability. For example, decision trees tend to be more interpretable than deep neural networks. After the proxy metrics have been designed, the problem becomes an optimisation task, which aims to optimise both the model quality (e.g., classification accuracy) and proxy interpretability metrics.

Among the above three evaluation approaches, the application- and human-grounded evaluations require active human feedback and thus are more time consuming and expensive than the functionally-grounded evaluation. However, it is extremely challenging (if not intractable) to design a proxy metric that can capture all the relevant factors needed for ensuring human-interpretable explanations. So far, most existing studies focus on the functionally-ground evaluation approaches, due to their lower "barrier to entry".

Gunning and Aha [51] argued that there is a trade-off between learning performance and explainability, and give a notional graph of such trade-off for various machine learning models. Fig. 2 shows the trade-off of some representative machine learning techniques/models adapted from [51].

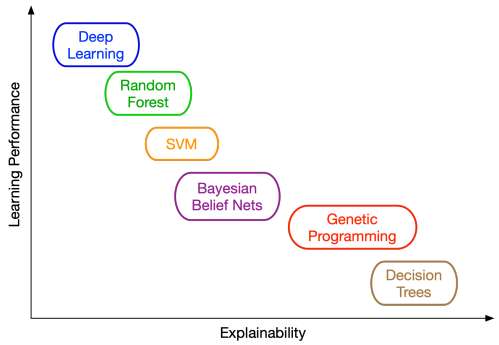

Fig. 2. The trade-off between learning performance and explainability of some representative machine learning models, adapted from [51].

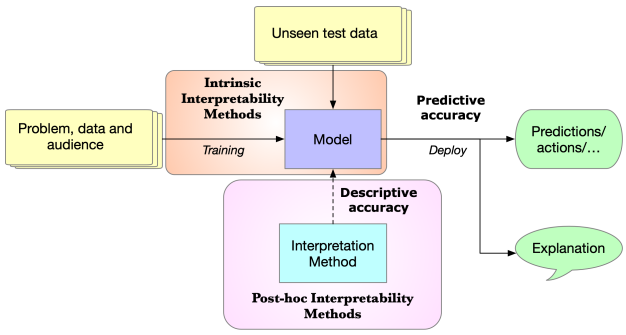

Fig. 3. The main stages in solving a machine learning problem where interpretability is important, adapted from [52].

Murdoch et al. [52] proposed a Predictive, Descriptive, Relevant (PDR) framework with three desiderata for evaluating and constructing interpretations, i.e., *predictive accuracy*, *descriptive accuracy*, and *relevancy*. The predictive accuracy of a machine learning model measures how accurate the model approximates the underlying relationship in the data. Then, the model may be interpreted by an interpretation method, and the descriptive accuracy measures how well an interpretation method captures the relationships learned by the model. An interpretation is defined as relevant if it provides insight for a particular audience into a chosen problem domain [52]. The relevance is typically judged by human users.

Based on the PDR framework, the XAI methods are grouped into the *model-based interpretability* (also called *intrinsic interpretability* [25], [53]) methods and *post-hoc interpretability* methods. Fig. 3 shows the main stages of solving a machine learning problem where interpretability is important, adapted from [52]. The intrinsic interpretability methods are used during the training process to train more interpretable models, e.g., by choosing an inherently interpretable model such as decision trees. The post-hoc interpretability methods are applied after the model has been trained. They typically learn a more interpretable model to approximate the behaviour of the trained model. The intrinsic interpretability methods generally increase the descriptive accuracy, but may decrease the predictive accuracy. On the other hand, the post-hoc interpretability methods have no effect on the predictive accuracy, but increase the descriptive accuracy [52].

The intrinsic and post-hoc interpretability methods have their own pros and cons, and might be used in different scenarios. For example, it is important to use intrinsic interpretability methods to obtain inherently interpretable models for making high-stakes decisions [54], [55]. This can be achieved when the different machine learning models perform similarly on the dataset (the Rashomon set is large [55]).

B. Taxonomy

Fig. 4 shows the taxonomy of common XGP approaches. Based on the stage to be used in the maching learning problem solving process [52], we group the XGP approaches into *intrinsic interpretability*, *post-hoc interpretability* and *visualisation* XGP approaches.

The intrinsic interpretability XGP approaches consider the model interpretability during the model training process, aiming to obtain a self-explanatory GP model directly. This is the main category of XGP methods, with the largest number of studies (189 out of 217, or 87% of the XGP references in this survey). These include both the recent studies that *explicitly* consider interpretability of the evolved GP models and previous studies that can *implicitly* improve the interpretability of the evolved GP models as a "bonus". According to how the interpretability is measured, we further divide the intrinsic interpretability XGP methods into that with smaller GP model size, lower (non-size) GP model complexity, fewer distinct features in GP models, interpretable GP (sub)-model structures, and those learning GP model interpretability measures. Note that model size is a special type of model complexity; however, there are a large number of studies on reducing model size (e.g., bloat control). Thus, we separate the methods with smaller model size from other non-size complexity measures to have a dedicated review in this area.

The post-hoc interpretability XGP approaches are applied after a high-quality but complex black-box model has been trained. They use a second, more explainable, model to approximate the behaviour of this black-box model. Specifically, the *global interpretability* approximates the behaviour on all the possible instances, and the *local interpretability* approximates the behaviour on a given single instance.

Visualisation is a natural way to improve interpretability. The visualisation XGP approaches are divided into the visualisation *by* GP (e.g., using GP for dimensionality reduction to facilitate data visualisation) and visualisation *for* GP (using visualisation tools to interpret the GP models/process).

III. Intrinsic Interpretability XGP Methods

The intrinsic interpretability XAI approaches aim to obtain more interpretable models without losing accuracy, typically by selecting an inherently interpretable model. Although GP models are already inherently interpretable, most existing GP-evolved models are still too large and complex to be interpreted. To further improve the interpretability, a key question is how to measure the interpretability of GP models. Based on existing studies, a GP model should usually be more interpretable if it has one or more of the following properties:

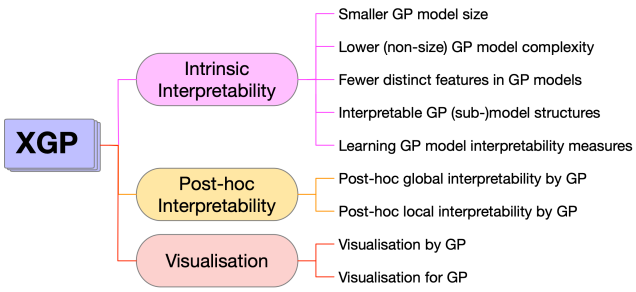

Fig. 4. The taxonomy of the XGP approaches.

- The GP model has a smaller size, e.g., fewer nodes in the tree-based representation. This leads to fewer steps for simulating the model behaviour by a human user, and thus better simulatability.
- The GP model has a lower (non-size) complexity, e.g., with simpler functions and flatter/shallower tree structure. This also leads to better simulatability.
- The GP model uses fewer distinct features, containing fewer concepts to comprehend when interpreting the model.
- The GP model has more interpretable model structures, such as following some grammar (e.g., speed plus speed). This leads to better decomposibility and simulatability.

In addition, one can learn new interpretability measures for GP models through questionnaires, and use them as fitness functions to evolve more interpretable GP models.

In the following, we will review each of the above types of methods one by one.

A. Smaller GP Model Size

Reducing GP model size is not a new research direction, and numerous studies have been conducted, which is known as *bloat control* [56]. It has been observed that the GP model size tends to grow rapidly as the evolution proceeds, leading to unnecessarily large models with many redundant components [57], [58], and extensive studies have analysed the causes of bloat [59]–[61]. Bloat control aims to reduce model size without (significantly) decreasing accuracy.

To reduce GP model size, one can modify different stages in the GP process. Fig. 5 shows the common strategies to reduce GP model size, and to which stages they can be applied.

1) Depth/Size Limit: Directly limiting the model size (e.g., tree depth for tree-based GP and number of instructions for linear GP) is the most straightforward method to control model size. This can be applied in initialisation, offspring generation and environment selection. With the predefined limit, we can design initialisation and genetic operators to generate only offspring within the limit, or simply discard any generated individual that exceeds the limit.

It is non-trivial to find a proper fixed limit to balance between the model accuracy and size. To address this issue, the work in [61] adjusts the limit dynamically based on the population distribution. The limit is initialised to a small value. During the GP process, an offspring exceeding the limit is discarded unless it is the new best individual [62]. The limit

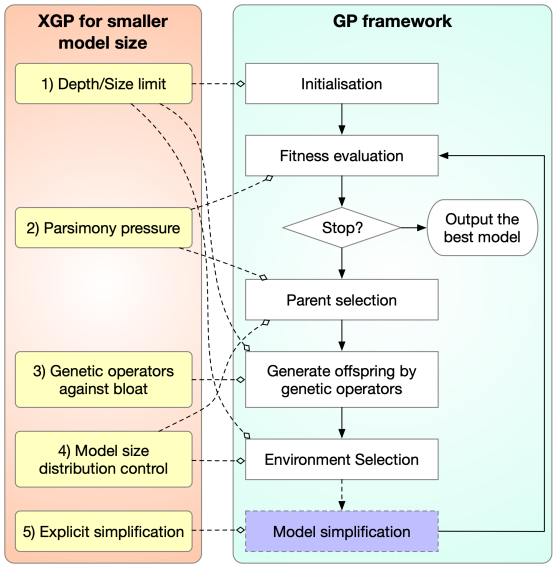

Fig. 5. The common strategies to reduce GP model size, and to which stages in the GP framework they can be applied.

TABLE I
PARSIMONY PRESSURE STRATEGIES USED IN DIFFERENT STAGES OF GP.

Stage in GP	Strategy	References
Fitness Evaluation	Fitness with penalty	[69]–[75]
Parent Selection	Pareto-based multi-objective	[76]–[91]
	Lexicographic	[92], [93]
	Double tournament	[94]

is also increased to the depth/size of the new best individual. On the other hand, if the new best individual has a smaller depth/size, the limit can also be decreased accordingly [63].

Other works use fixed-size model templates, such as Cartesian GP [31] with fixed number of rows and columns of nodes, or a full r-ary tree with depth d [64], [65]. The depth/size of the template is the limit. Note that any node in the template can be the terminal, thus the actual model size can be smaller than the limit. Existing studies (e.g., [66], [67]) have shown that Cartesian GP is lack of bloat and can obtain more interpretable models than other models such as decision tree [68].

2) Parsimony pressure: The parsimony pressure methods are based on the idea of penalising large models during the GP process, mainly used in fitness evaluation and parent selection.

Table I shows the parsimony pressure strategies used in different stages of GP. In fitness evaluation, a typical strategy is to add a penalty term to penalise large individuals. The augmented fitness function g is defined as $g = f + p(s)$, where f is the original fitness function (e.g., mean squared error), s is the model size (e.g., number of nodes in the GP tree), and $p(s)$ is the penalty term related to s.

Common strategies to define $p(s)$ include the *death penalty* and *linear penalty*. The death penalty strategy sets an infinitely bad fitness to large individuals (e.g., whose size are larger than the average size of the population [69]).

The linear penalty strategy [70] defines $p(s) = \beta \times s$, where β is the penalty coefficient. A proper β value is critical to balance between model accuracy and size. If β is large enough, then the size is used to break the tie between two models with the same accuracy (e.g., [71]). Existing studies set it to a fixed value based on domain knowledge/trial-and-error (e.g., [70], [72]–[74]) or change it adaptively based on the accuracy and size of the current best individuals [75].

We can also consider parsimony pressure during the parent selection. Multi-objective techniques have been commonly used for this purpose. It simply treats minimising the model size as an additional objective, and uses the multi-objective parent selection schemes such as the ones used in NSGA-II [80], [81], [85] or SPEA2 [77]–[79]. The advantage of multi-objective techniques over the penalty strategy is that we can obtain a comprehensive set of models with different trade-offs between accuracy and complexity for users to select from.

When using multi-objective techniques for parsimony pressure, an issue is that the search tends to bias towards small models [79], but can hardly find individuals with good fitness, since they generally have large size. To address this issue, some works [79], [81] enhance the population diversity to reduce the number of small individuals in the population. Other works [89]–[91] use the α-dominance instead of the traditional dominance relation to balance between the fitness and model size. Another common approach is to use a two-stage search process, where the first stage focus only on the fitness, and the second stage takes model size into account [76], [86]–[88].

Instead of treating fitness and size as equally important, the lexicographic parsimony pressure methods [92], [93] treat the fitness as the primary objective and model size as the secondary objective. When comparing two individuals during the parent selection (e.g., tournament selection), an individual is considered to be better if (1) it has a better fitness or (2) it has the same fitness but smaller size.

As an extension of the lexicographic parsimony pressure methods, the double tournament selection [94] selects the parents through a "qualifier" tournament and a "final" tournament. They are based on parsimony and fitness, respectively, or vice versa. The contestants for the "final" tournament are the winners of the "qualifier" tournament.

3) Genetic operators against bloat: Instead of randomly generating offspring and discarding/penalising large ones, another alternative is to design specific genetic operators to actively generate small offspring. A common idea is to make the offspring have similar structure as the parents. For example, in tree-based GP, for crossover, we can select only the sub-trees that inherit the same structure of the upper part of the parents (e.g., one-point crossover [95], [96], as shown in Fig. 6) or with similar size (e.g., size-fair crossover [97]). We can also consider both the sub-tree size and the common structure they inherit (e.g., Homologous crossover [97]–[99]). For mutation, we can mutate the value of a single node instead of replacing a sub-tree (e.g., point mutation [96], [100]) or control the newly generated sub-tree so that its size is not much larger than the replaced sub-tree (e.g., size-fair mutation [101] and the prune and plant operator [102]).

Another strategy for linear GP [103] considers each individual (a sequence of instructions) as a number of blocks

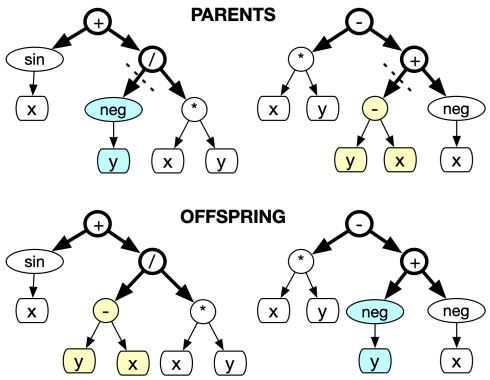

Fig. 6. The one-point crossover to control model size. The thick lines show the common structures of the parents. The sub-tree $neg(y)$ from parent 1 and $y - x$ from parent 2 are swapped, as they inherit the same structure of the upper part of the parents.

of instructions (so-called "pages"), each with a fixed number of instructions. The crossover swaps a single block (page) between the parents. This way, the size of the individuals are not changed after crossover.

4) Model size distribution control: The above strategies mainly work at the individual level. In contrast, the model size distribution control strategy works at the population level to control the distribution of model size in the GP population. It is generally used in the parent selection and environment selection stages.

For parent selection, a typical method is the use of external archive, which stores small and high-quality individuals (designed by users [104] or selected from previous populations [105]). By selecting parents from the archive, there will be more small offspring generated in the population. Another idea is to use the niching technique to divide the population into species, and conduct crossover between parents in the same species (e.g., the Spatial Structure + Elitism Model [106], [107], neat-GP [108], [109] and parallel GP [110]). This has shown to be effective to control bloat implicitly.

For environment selection, a commonly used strategy is the operator equalisation [111]. It predefines a target distribution of model size in the population (e.g., uniform distribution [112], [113]). The probability of accepting a new offspring depends on how much the offspring will drive the current distribution towards the target distribution [112]. Further extensions including adapting the target distribution during the GP process [114] and mutating the offspring to fine-tune the distribution [115].

5) Explicit Simplification: Compared with the aforementioned strategies that modify/enhance the existing GP steps, the explicit simplification strategy [116] is an extra post-processing step to prune the redundant parts of GP individuals.

For simplifying GP models, the key issue is to balance between the model size and using redundancies to protect the truly useful building blocks from being destroyed [59]. Specific decisions include (1) "*when*" to simplify, (2) "*which individuals*" to simplify, and (3) "*how*" to simplify. There have been various strategies to address these design issues, which are summarised in Table II.

TABLE II
THE STRATEGIES FOR GP WITH SIMPLIFICATION REGARDING "WHEN", "WHICH INDIVIDUAL" AND "HOW" TO SIMPLIFY.

Decision	Strategy	References
When	Every generation	[2], [117]–[122]
	Final generation	[2], [123], [124]
	At certain interval	[119], [125]–[128]
Which individuals	All individuals in the population	[2], [119], [122], [125]–[127]
	The best individuals in the population	[2], [121], [123], [124], [128]
	Randomly with some probability	[117]
	The parents for breeding	[118], [120]
How	Genotypic (Structural)	[2], [117]–[119], [122], [123], [125]–[127], [129]–[131]
	Phenotypic (Behavioural)	[120], [121], [124], [126]–[128], [132]–[135]

Regarding "when" to simplify, the commonly used strategies are based on frequency (i.e., after every k generations). If $k = 1$, then the simplification is conducted after every generation. If k equals the number of generations, then the simplification is applied only to the final individuals after the whole GP process.

Regarding "which individuals" to simplify, the commonly used strategies mainly include (1) all the individuals in the population; (2) the best individuals in the population; (3) randomly selected individuals; and (4) the parents for breeding (use the simplification as a special operator [118] or simplifying the parents before the genetic operator [120]).

The decisions on "when" and "which individuals" to simplify should be made together to balance between model size and protection of useful building blocks. For example, if the simplification is applied in every generation, then we should not simplify all the individuals in the population.

Regarding "how" to simplify the individuals, the existing strategies can be divided into *genotypic* and *phenotypic* simplification approaches. The genotypic simplification approaches directly simplify the model based on its structure. The most straightforward way is to use existing basic algebraic simplification rules, such as $x - x \to 0$ and $x/x \to 1$ (e.g., [117]–[119], [125], [129]). If domain knowledge is available, we can develop more simplification rules, e.g., $\max\{x, x - y\} \to x$ if the variable y is always positive [122]. Hashing has been employed to speed up the simplification [119], [131].

The phenotypic simplification approaches simplify programs based on behaviour, i.e., it replaces a large sub-tree with a smaller one with similar behaviour. An intuitive approach arbitrarily changes the model to make it smaller. If the model behaviour is not significantly changed, then the original model is replaced by the smaller one [132], [134], [135]. Also, a part of the model can be pruned if it is never executed during the model evaluation [136]. A more advanced strategy in tree-based representation is to consider sub-tree *contributions*. For

example, if there is always little difference between the output of a sub-tree and that of its child, then the sub-tree can be replaced by that child [120], [126], [127]. Permutation test is also commonly used to estimate the contribution of a sub-tree (e.g., [124], [128], [137]). If the overall model output is not affected by shuffling the output of a sub-tree, then the sub-tree can be pruned. This is generally more accurate but slower than directly checking the outputs. Another approach is to replace a sub-tree with a smaller sub-tree with similar semantics as the replaced one [133]–[135] or the desired semantics back-propagated from the root node [121].

B. Lower (Non-size) GP Model Complexity

Model size reflects only a narrow aspect of interpretability, and it is highly likely to have a small but uninterpretable model with complex operators (in combinations with features). To address this issue, various model complexity measures have been developed to consider the complexity of GP models beyond model size.

Model complexity measure and control methods in GP can be broadly grouped into two categories: *structural* and *behavioural/functional* complexity. These two types of methods will be reviewed below.

1) Structural Complexity Reduction: The structural complexity of models contributes directly to the interpretability of models. Intuitively, a model with a simpler structure should be more comprehensible on feature importance, feature interactions, and the process of prediction inference.

Apart from the most straightforward and commonly used structural complexity measure, i.e., model size that has been reviewed in Section III-A, a number of structural complexity measures and control methods have been proposed in the last two decades. Many structural complexity measures consider the shape of the GP trees and distinct the complexity between balanced and skewed trees with the same size. Some complexity measures share the same idea, e.g., the expressional complexity [138] and the visitation length [139]. They both define the complexity as the sum of nodes in all the subtrees, thus consider flatter trees with fewer nested function nodes less complex than deep unbalanced trees. A structural complexity presented in [140] recursively aggregates the complexity of the nodes underneath each internal node according to its function. Therefore, with their measure, the total complexity of a GP model is heavily dependent on the level/depth of internal nodes with more complicated functions, for example, the complexity of a node with a complicated function, e.g., "sin"/"cos"/"exp", increases exponentially with the size of the subtree rooted on the node. These complexity measures can be utilised in designing new fitness functions and/or genetic operators to prefer and generate simpler models, and by multi-objective GP methods to generate models with a good complexity-performance trade-off. It is much easier for users to pick up models with the right level of complexity for further analysis or deployment.

Another group of GP methods seeks simpler models in their structure with some novel representations. A fixed model structure with adaptive weighted splines was presented in [141], where each GP model is composed of a number of feature splines. The proposed semi-structured GP method no longer fully presents the symbolic capability of GP, but the models developed can be regulated far more easily, thus are simpler and potentially easier to interpret. The work in [142] incorporates an epigenetic layer to specify active genes in linear GP with stack-based representation that has the advantages of guaranteeing syntactic validity regardless of the change to GP models.

2) Functional Complexity Control: Perfectly accurate models are often obscure and infeasible to convince process engineers in industry who deploy and use the models [143], thus it is necessary to measure and control functional/behavioural complexity for more interpretable GP models.

To discourage GP models with highly nonlinear behaviour, various functional complexity measures and control methods have been developed. Some measures approximate the function/behaviour of GP models with a simple structure, e.g., Chebyshev polynomial [143]. The degree of the polynomial is then treated as the complexity of GP models. By minimising the functional complexity along with the modelling errors, GP can produce models with smoother response space and easier to interpret. Different from approximating the function represented by GP models, another group of methods directly work on the function itself and proposed various indicators of the nonlinearity of GP models. Some typical measures used in previous research are described as follows. The curvature of functions [144] counts the number of different slopes while assigning higher weights to inversions in the slope signs. [145] further develops this idea and quantifies the degree of curvature by the correlation between two vectors; one contains the pairwise distances between the input data and the other one contains the corresponding pairwise distances between the predicted outputs. For a rugged function, there is no correlation between its inputs and outputs. The Tikhonov regularisation, which takes norm operators of the function in a Hilbert space, measures the smoothness of GP models from different aspects such as extreme values, ruggedness and wiggliness [146]. Another one is the Vapnik–Chervonenkis (VC) dimension of the function represented by GP models [147], [148]. VC dimension measures the complexity of model by its ability to learn from any given data. The Hessian complexity of GP models [149] considers the form of the ratio of the non-constant entries in the Hessian matrix of GP models. The partial derivatives of the model with respect to each feature in the matrix is a good indicator of whether and how it influences the output of the model. A further one is Rademacher complexity [150], [151], which measures how well the model correlates with randomly generated labels. In its simplest form, Rademacher complexity measures the ability of a learning model to fit random noise [150]. It can also be as sophisticated as possible in approximating the supremum of the correlation between the model and Rademacher random variable [151]. Rademacher complexity has a tighter bound than VC dimension on the generalisation error, thus can potentially avoid oversimple models. Although the VC dimension-based model complexity measure is precise, it is usually complicated to calculate and very computationally

expensive for GP with varying model structures. Another simple but interesting idea is to measure the complexity of GP models by the computational time required for evaluation [152]. The evaluation time is a sensible indicator of both the structural and functional complexities since it usually takes longer for evaluating models made up of computationally expensive functions and/or large structures.

3) Indirect Methods to Generate Simpler Models: Another group of GP methods generate simpler models by reducing overtraining. The sampling technique has shown to be effective for mitigating overtraining. At every generation, GP individuals are evaluated on a subset of sampled training data only. This implies only individuals that fit various subsets well will remain in the population. These individuals are less likely to be overtrained nor capture the noise via overcomplex structures and/or functions. Various sampling techniques have been employed to generate simpler models [153]–[156]. A statically random sampling technique has been used in [153]. At every generation, GP learns from a subset of randomly sampled training data, which contains half of the training data while keeping the same distribution as that of the entire training set. This was extended in [154] by adding two flexible configurations: —the size of the sampled subsets and the sampling frequency. The work in [155] further developed the idea of randomly sampling a single instance for evaluation while balancing it with periodically using the entire set of training data. Bootstrapping is employed in GP [156] to prefer simpler individuals. GP individuals are evaluated on a number of bootstrap replicates, each of which is formed by repeatedly sampling from the training data with replacement. Introducing bootstrapping into GP gives simpler individuals a high priority for breeding and survival, since simpler individuals usually vary slightly on the bootstrap errors.

C. Fewer Distinct Features in GP Models

In XAI, a good explanation should be *selected* [50] to reduce the causes/features to relevant ones only, which can be achieved via *feature selection*. Furthermore, the original features collected could be low-level raw information, which requires further learning/transformation to discover a good explanation. This can be achieved by *feature construction*. Both feature selection and feature construction are challenging because of the large search space and the complex interactions between features, where feature construction is even harder since it also needs to explore ways to create new high-level features. Both the selected and constructed feature sets are expected to be much smaller and provide insights between features that facilitate interpretability.

GP has built-in feature selection and feature construction ability, and implicitly and automatically takes feature interactions during the learning process. Tran at al. [158] performed an extensive investigation on this using high-dimensional classification data. The results show that GP has the ability to select a very small number of features (e.g., less than 1%) while maintaining the classification performance. GP can also construct a single new high-level feature by using only a few original features, which were selected from a large feature set,

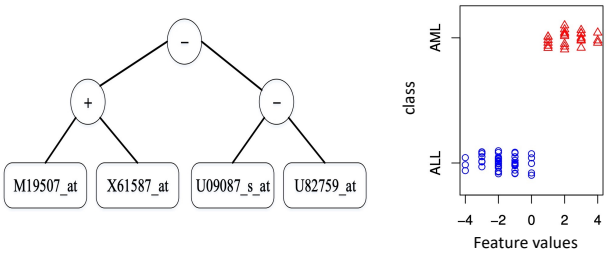

Fig. 7. An example GP tree generated from the Leukemia dataset [157].

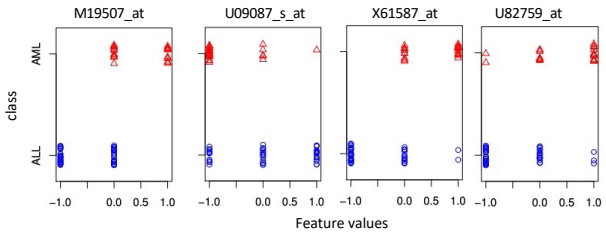

Fig. 8. Visualisation of feature values and the class labels on the Leukemia dataset [157].

e.g., with thousands of features, to obtain high classification accuracy. An example GP tree is shown in Fig. 7 [157], where the leaf nodes are the selected features. This GP tree can also be seen as a constructed feature (i.e., M19507_at + X61587_at - U09087_s_at -U82759_at), which can separate the two classes much better than the original four features as shown in Fig. 8. Furthermore, the GP tree in Fig. 7 is also a classification model, showing how the classification decision is made (from input features/data to the output class label), which can be potentially interpreted by users.

1) Feature Selection: Different methods have been introduced to achieve feature selection in GP to facilitate model interpretability, such as incorporating the permutation-based feature importance measures [87], [137], [159]–[164] to evaluate the quality of the feature subsets, partial derivative to evaluate feature importance [165], and adaptive terminal selection and bloat control methods [159]. For example, an interaction-transformation representation is employed in [165], where the partial derivative can be calculated to evaluate the partial effect of a feature in a symbolic regression model (i.e., GP individual) to avoid complex models. By restricting the search space to simple mathematical expressions, this approach can facilitate model explanation, shown as providing the closest explanations to the ground-truth and a close approximation to Shapley values (i.e., how much a feature contributes to the deviation from a reference point).

Feature selection can also be achieved by feature ranking. However, most existing feature ranking methods lead to redundancy in the obtained feature subset due to ignoring interactions between features. GP implicitly considers interactions between features when performing feature ranking. A simple method to measure the importance of features is based on the frequency of a feature appearing in good GP individuals. Ahmed at al. [166] measured the importance of features based on their frequency of appearance in good GP models. Experiments on biomarker detection datasets show that a small number of top ranked features can significantly improve the classification performance, which effectively select almost

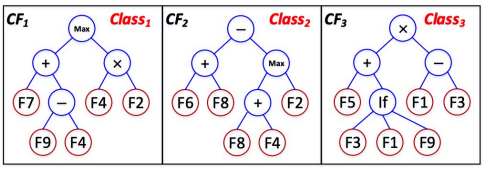

Fig. 9. Multiple feature construction with a multi-tree representation for a 3-class dataset, where CF_1, CF_2 and CF_3 are the three constructed features from the three trees [174], and each corresponds to one class.

all of the predefined biomarkers, which are key for domain experts to explain the data. Another approach is to consider the fitness values of the GP models that features appeared [167], where the results show how different numbers of top ranked features affect the classification performance, which can help understand and explain the data and the generated GP model. Hu [168] developed a linear GP method, where the final model was learned by using the most frequently occurred features from the a previous one as input, i.e., using a significantly reduced feature set to build the model. The results showed that the selected features can help linear GP obtain significantly better prediction performance and more interpretable models.

2) Feature Construction: GP has been used to achieve feature construction in many studies. Icke and Rosenberg [169] proposed to use GP to construct a few features for improving the classification accuracy, and their interpretability is achieved by controlling the complexity of the features constructed. Later, Virgolin et al. [170] proposed a similar idea and evolve a group of crucial and compact (and thus interpretable) features to maximise the accuracy. Nag and Pal [171] used GP to construct features for classification, so that the classes become linearly separable in the constructed feature space, which facilitates the interpretability of the learned classifiers. Peng et al. [172] applied GP to rolling bearing fault diagnosis. Detailed analysis and visualisation show that GP effectively constructs diverse features to capture different patterns in the data to improve the diagnosis accuracy with explainable tree-based models. Furthermore, most GP-based feature construction methods use a single-tree approach [169]–[171], which work well when the number of classes is small, e.g., binary classification, but their accuracy or interpretability might be limited when the number of classes is large [173]. Multi-tree GP is used to construct multiple class-dependent features for multi-class classification [174]. As shown in Fig. 9, one constructed feature or tree corresponding to one class, taking only features most relevant to this class as input, and therefore can better explain how a classifier separates instances of different classes.

Later, Ain et al. [175], [176] performed multiple feature construction using multi-tree GP with different types of features for melanoma detection. By keeping the same type of features in the same tree, the features selected and constructed by GP have good interpretability, since they can potentially match the way dermatologists make a decision. For example, the lesion colour features selected correspond to the geometrical shape characteristics of melanoma, such as irregularity indices, corners and the major asymmetry index, which are commonly used by dermatologists in clinics in the real-world situations. Most GP methods construct features based on the *root* of the tree. In addition, the internal nodes (sub-trees) are also utilised as constructed features to improve the performance [177], [178]. These constructed features can potentially be interpreted as meaningful information in the domain. In particular, the GP method in [177] can successfully find biomarkers that are indicators of a particular disease state or other physiological state of an organism, which are exactly what a doctor or biologist is looking for. Lensen [179] proposed to use GP to automatically construct the *relationship* between features, which can particularly help with the interpretable association rule mining. It showed that GP can evolve interpretable relationships between features.

D. Interpretable GP (Sub-)Model Structures

If a GP model has interpretable model structures or modules, then it has a better decomposibility. The existing strategies can be grouped into three categories: (1) interpretable function nodes, (2) interpretable combinations between nodes, and (3) problem decomposition.

1) Interpretable function nodes: A main reason for the poor interpretability of deep neural networks is that the operations (weighted sum and activation such as the Sigmoid and $\tanh$ functions) are not interpretable. Similarly, traditional GP function nodes mainly include basic arithmetic and logic functions (e.g., $+$, $-$, $\exp$, $\min$, `AND`, `OR`) with no domain-specific meaning. To address this issue, interpretable problem-specific function nodes can be designed to replace the less interpretable basic functions. A typical example is GP for image processing, where existing studies have designed various interpretable image processing functions, such as region detection. Compared with deep neural networks (DNNs), although GP-based methods cannot easily achieve better accuracy than state-of-the-art DNNs particularly on large-scale image datasets, they have already achieved similar or better performance than DNNs when the number of images is small [180]. However, DNNs often have tens or hundreds of layers in the network with millions of parameters, which makes it almost impossible to interpret the learned models. In contrast, the tree structure in GP often has a much smaller number of layers, e.g., less than 10.

The interpretable image processing functions generally require specific inputs and outputs. Strongly-Typed GP (STGP) [32], [181]–[183] has been used to satisfy the input-output constraints of the operators. The use of STGP for image classification started many years ago [184], when the interpretability of models was not in a high demand as nowadays. But those earlier works have shown potential interpretability of GP for image classification via the tree-based model by decomposing the complex image classification task to relatively smaller tasks, being addressed by different nodes of the tree.

A typical example of GP tree for image classification is shown in Fig. 10, where the left part shows an example of a multi-layer model structure [185]. The input of the tree is the original image and attributes for locating a region of interest in the image. The internal function nodes include *automatically selected* image-specific operators, such as the region detection

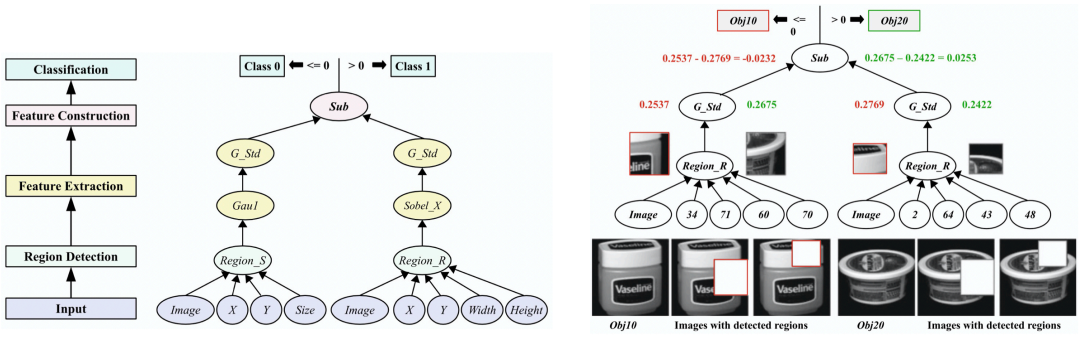

Fig. 10. An example multi-layer model structure [185].

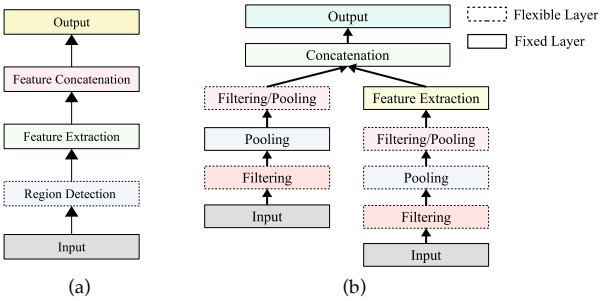

Fig. 11. Examples GP programs for Image Classification [184].

operators (e.g., $Region_R$ to detect a rectangle region whose top-left point has a coordinate (X, Y) and with size defined by $Width$ and $Height$), feature extraction operators (e.g., the Gaussian smooth filtering operation shown as *Gaul*, G_Std for calculating the standard deviation or any other appropriate operators), and feature construction operator (e.g., subtraction Sub). The right part in Fig. 10 shows an evolved tree with the decision making process predicting the class label of the input image (red for the example image of the *Obj10* class and green for the *Obj20* class). This example GP model can effectively detect two squares of different sizes from the input image, which are key regions that can distinguish these two classes. The right branch captures a top right region of the lid, which shows to be a white rectangle in the *Obj10* class and a round corner in the *Obj20* class. By selecting such key regions to perform classification, together with other feature extraction and construction operators, it is easier to explain the models to end-users.

Al-Sahaf at al. [186] developed a two-tier GP method for image classification, where the two tiers are the aggregation and classification tiers. The former tier takes raw pixels as input, selects regions from the image and then performs feature extraction to convert each region to a single value using simple operations like mean or maximum. This facilitates the interpretability by allowing the input-to-output process being easily shown in a single GP tree. Later, a GP method with explicit region detection, feature extraction, feature construction and classification functions are developed [187], where experiments on facial expression data shows the model can be explained by selecting the key regions in faces, e.g. the eye and corners of the mouth. Almeida et al. [188] applied GP to the automatic identification of regions of interest in remote sensing images by accessing the similarity of associated time series. The GP method can find better combinations of time series similarity functions, and shows clearly how they are combined to achieve promising performance. Shao et al. [189] developed a multi-objective GP method for image classification by extracting domain-adaptive global feature descriptors. The two objectives are minimising the classification error and the tree complexity. The learned GP models not only improved the classification accuracy over hand-crafted features, but also have potentially better explainability, since the model complexity is minimised and the function nodes are required to extract meaningful information from the images. Complex image classification tasks often require various types of features to be extracted and combined together to achieve promising performance. Therefore, a new representation (shown as Fig. 10) is proposed in [190], which allows GP to take raw images as input, and automatically select and combine various functions to extract various types of global and/or local features. Fig. 11 is more complex than Fig. 10, but compared with DNNs, Fig. 11 still has much better potential to be interpretable, as briefly discussed in [184].

Another example is to use GP to automatically learn decision trees, where each terminal is a class label, and each function node is a possible feature test. Early studies [36] defined each function node as a feature, whose arity equals the number of possible values of the feature. This has been further extended to consider both continuous and nominal features [38]. The function nodes can be easily interpreted from the meaning of their features.

STGP has been used to learn more flexible oblique decision trees [37], where each function node is defined as a test on a linear combination of features. Specifically, a function `Cond`k`Var` is defined to test on the linear combination of k features. Its arity is $2k + 3$, where the first $2k$ elements represent the weighted sum of the k features. Among the last three elements, the first is the threshold, the second and third are the class labels of the two branches. The number of features in the function is limited to at most three to improve its interpretability.

With the help of grammar, GP has also been successfully used to evolve effective fuzzy pattern trees [43]–[47], which

is another important inherently interpretable machine learning model. A fuzzy pattern tree is a tree-based model, whose inner nodes are generalised (fuzzy) logical and arithmetic operators, and leaf nodes are the fuzzy variables. It combines the interpretability of both decision trees and fuzzy logic, and can be interpreted by recursively parsing from the root node.

On the other hand, grammars with domain-specific interpretable functions have been used to learn interpretable decision trees with composite feature tests (multiple single-feature tests concatenated by logic operators) [191], association rules [192], rule induction systems [193], [194] and multi-view learning classification rules [195], [196].

2) Interpretable combinations between nodes: In addition to designing interpretable functions, we can still use the basic functions but only allow interpretable combinations between nodes. A typical example is GP for automatic natural law discovery [197], where the collected data are physical observations accompanied by their units of measurement (e.g., $\mathtt{m}$ for length, $\mathtt{s}$ for time, $\mathtt{kg}$ for mass). Although the data are all floating numbers and can be combined in any way, some combinations (e.g., length+mass) are much less interpretable than others (e.g., time+time), if not interpretable at all.

We can design type constraints based on the dimensions of the nodes to avoid less interpretable combinations. Specifically, the type of a node is defined as a vector, where each element represents the exponent on the corresponding unit of measurement. For example, for the measurement vector $[\mathtt{kg}, \mathtt{m}, \mathtt{s}]$, then the types of a mass variable ($\mathtt{kg}$) and a time variable ($\mathtt{s}$) are $[1, 0, 0]$, and $[0, 0, 1]$, respectively.

From domain knowledge in physics, interpretable addition/subtraction requires the two variables to have the same type. Moreover, the type of the output depends on that of the inputs and the function, e.g., $\mathtt{m} \times \mathtt{m} \rightarrow \mathtt{m}^2$. We can define corresponding type constraints as below:

- $+$: $([m, l, t], [m, l, t]) \rightarrow [m, l, t]$
- $-$: $([m, l, t], [m, l, t]) \rightarrow [m, l, t]$
- $\times$: $([m_1, l_1, t_1], [m_2, l_2, t_2]) \rightarrow [m_1 + m_2, l_1 + l_2, t_1 + t_2]$
- $/$: $([m_1, l_1, t_1], [m_2, l_2, t_2]) \rightarrow [m_1 - m_2, l_1 - l_2, t_1 - t_2]$

The GP respecting the constraints on dimensions is called dimensionally aware GP [198]. It penalises the combinations violating the constraints during the search process, and has successfully evolved more interpretable models for energy conservation law [198], automated innovisation [199], scheduling dispatching rules [200], crowd behaviour modelling [201], and Feynman Equations [202].

In contrast with defining type constraints, a more generic approach is to use *context-free grammar* [203] to restrict the model structure, and use grammar-guided GP [33] or grammatical evolution [34] to learn GP models following the grammar. Interpretable grammars can help GP obtain more interpretable models. Different domain-specific grammars have been designed to learn laws in high-energy physics [204] and scheduling heuristics [205].

Even for general symbolic regression with no domain knowledge, grammar may still be used to improve the model interpretability. For example, a grammar is proposed in [206]–[208] to restrict the structure of the evolved model to be the so-called *canonical form*. The model alters between "L" layers and "NL" layers. A "L" layer is the sum of a polynomial/rational and possibly a "NL" layer. A "NL" layer is the product of a polynomial/rational and a non-linear function, which takes another "L" layer as the input. An example model is shown in Fig. 12. The canonical form can well balance between the interpretability (through "L" layers) and capability (by "NL" layers).

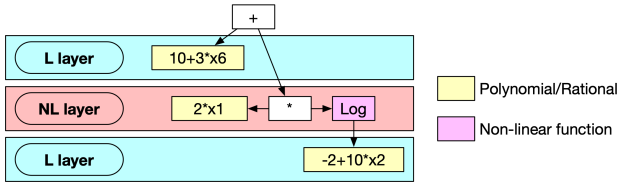

Fig. 12. A canonical form of GP model.

3) Problem decomposition: Instead of solving the problem as a whole and obtaining a huge and complex model, we can decompose the problem and learn interpretable modules or sub-models to solve the decomposed sub-problems.

Problem decomposition can be *sequential* or *parallel*. In sequential problem decomposition, the sub-problems have hierarchical relationship, and the solutions of lower-level sub-problems serve as modules for solving higher-level sub-problems or the original problem. Automatically Defined Functions (ADFs) [209] is an earliest approach for sequential problem decomposition. ADFs evolve building blocks along with the main population, which can then be reused as modules in the GP individuals. The layered-learning GP approaches [210]–[216] decompose the problem into hierarchical sub-problems with increasing complexities, and solve them sequentially (the original problem is solved last). The solutions of the former easier sub-problems are reused as initial individuals to solve the latter more complex sub-problems. The run transferable library approaches [217]–[220] follow the same sequential run process, but keep a library of functions rather than the raw GP individuals. Each element in the library is referred to as a tag addressable function. It is updated after each GP run, and used for initialising the individuals for solving the next sub-problem.

In parallel problem decomposition, the sub-problems complement with each other (e.g., fitting different subsets of data points), and their solutions work in a team to solve the original problem. Ensemble GP is a common approach in this category, where each sub-model in the ensemble fits a subset of the data points. As each sub-problem is simpler than the original problem, the sub-model should also be more interpretable. Existing studies [221], [222] have shown that the conventional bagging and boosting ensemble learning approaches have limitations for GP, as they are susceptible to noise and can have large biases, thus may not be better than any simple ensemble in some cases. The N-Version (island-based) GP and cooperative co-evolution GP [223], [224] have shown better performance than bagging and boosting. They maintain multiple populations, each for evolving a sub-model. The ensemble is formed by all or a random subset of the sub-models. The orthogonal evolution of teams approach [225] improves the performance by reducing the correlation between

the team members. As a further improvement, tangled program graphs [226]–[239] employ graph-based representations to learn the teams. Specifically, it consists of a "node" population and a "program" population. The two populations cooperate to evolve good teams in the node population (pointing to the sub-models in the program population) and good sub-models in the program population.

In addition to cooperative co-evolution, competitive co-evolution can also be used for problem decomposition. The Pareto-coevolutionary GP [240] and symbiotic bid-based GP [241]–[243] approaches maintain populations of GP sub-models, their teams and points. The teams, consisting of the sub-models in the model population, aim to fit the points as much as possible, while the points are evolved to better distinguish the teams.

E. Learning GP Model Interpretability Measures

Interpretability is subjective, and there is no universally accepted interpretability measure. Most intrinsic interpretability XGP approaches simply optimises an interpretability proxy measure, but without real interpretability assessment from human users. To address this issue, Virgolin et al. [244] proposed to learn the interpretability measure based on the data generated from a survey. The survey was conducted to collect human feedback on randomly generated formulas in terms of simulatability and decomposability as defined in [48]. The survey data consists of four model features: the number of non-arithmetic operations, the number of consecutive compositions of non-arithmetic operations, the number of operations, and the size of the formula. The output is the product between the correctness ratio of an response's answer and this person's confidence, which is treated as the estimate of interpretability. Then, a weighted sum function of the four model features is learnt from the survey data as a new model interpretability measure. By optimising this new interpretability measure, GP can obtain more accurate models than simply minimising model size, and the evolved models are also arguably more interpretable. As discussed in the paper, the major limitation of the work is the limited responses and the feedback is mainly from students and faculty members from universities. Then, Virgolin et al. [245] proposed another approach called "Model Learning with Personalized Interpretability Estimation (ML-PIE)". This approach is a human-in-the-loop system, which asks for human feedback during the evolutionary process. Specifically, the system regularly shows a pair of evolved models to users and asks for their opinion on which one is more interpretable. Then, the internal GP adjusts the interpretability objective function based on user feedback.

F. Summary

This section reviews intrinsic interpretability XGP approaches that consider model interpretability from the aspects of (1) model complexity (including size), (2) number of features, (3) meaningful combinations between features, and (4) learning interpretability measures from questionnaires.

The first three measures do not rely on user preference, and focus on different aspects of model interpretability. They may not be sufficient to obtain interpretable models by themselves. For example, a small model with many non-linear functions might not interpretable, and a model with few features but meaningless combinations among them can be hard to interpret. The model complexity, structure and number of features should be considered simultaneously to obtain more interpretable models.

In contrast, the last measure can better reflect model interpretability. However, different applications/domains may need different interpretability measures, and it can be time consuming and costly to collect data for learning an accurate interpretability measure.

IV. Post-hoc Interpretability XGP Methods

The post-hoc interpretability methods aim to approximate the behaviour of a pre-trained black-box model. In contrast to the intrinsic methods, there is very limited work on post-hoc XGP methods. They can be grouped into (1) post-hoc *global* interpretability by GP that explains the global behaviour on all instances; and (2) post-hoc *local* interpretability by GP that explains the local behaviour on a specific instance.

A. Post-hoc Global Interpretability by GP

Due to its good interpretability, GP can be directly used to achieve post-hoc global interpretability. Specifically, a dataset is first generated from the explained model, where the target output is the prediction of the explained model. Then, an interpretable GP model is trained to fit the generated dataset. This process is almost the same as intrinsic XGP approaches, except that we use the dataset generated from the explained model rather than the original one. Note that the GP models must be interpretable (e.g., have interpretable structures and small size) to be able to explain the black box model. In [246], a multi-objective GP method is developed to learn an interpretable decision tree by simultaneously maximising the reconstruction ability of the model and minimising the model complexity. The results on a wide range of classification datasets show that the proposed method achieves at least the same accuracy but with lower model complexity and better interpretability than the compared methods. For example, it produces a 4-layer decision tree that can replicate the predictions of the DNN with 200 hidden layers.

In [247], [248], two novel post-hoc interpretability GP algorithms are developed to balance the accuracy and comprehensibility of the learned models. The first algorithm named GEMS is used for ensemble creation, and the second one named G-REX is used for rule extraction from opaque models. The main property of GEMS is the ability to combine smaller ensembles and individual models in a completely random way. Moreover, GEMS can use base models of any kind and the optimisation function is very flexible, easily permitting inclusion of, for instance, diversity measures. The experiments show that GEMS can obtain higher accuracy than both straightforward design choices and random forests and AdaBoost. The key quality of G-REX is the inherent ability to explicitly control the accuracy vs comprehensibility trade-off. Compared to the standard tree inducers C5.0 and CART, and some well-known

rule extraction algorithms, the rules extracted by G-REX are much more accurate and compact.

B. Post-hoc Local Interpretability by GP

The post-hoc local interpretability aims to explain the local behaviour of the given model on a specific instance. In contrast with the global interpretability, which requires a comprehensive dataset with a wide range of instances, the local interpretability requires only a local dataset with instances similar to the given instance.

For classification and regression tasks, an instance is a feature vector $\mathbf{x}$, and the local dataset can be generated by sampling around $\mathbf{x}$ in the feature space (e.g., following a multivariate Gaussian distribution [249]). The work in [249] developed a GP method to explain the local behaviours of pre-trained black box models such as random forests and DNNs, and the results show that GP can obtain better balance between accuracy on the generated local dataset and interpretability than the other post-hoc XAI methods (e.g., LIME [18] and decision trees). Note that if the dataset itself is complex, to fit the dataset well, the (intrinsic interpretability) GP models might need to be large and complex, and less interpretable. To address this issue, we can use an even simpler model to explain the complex GP model. For example, Filho et al. [250] proposed to use linear model to approximate the local predictions of a complex GP model around a given datapoint for symbolic regression. The local dataset contains a number of nearest neighbours from the training set (rather than randomly sampling). Experimental results and extensive analyses show strong approximation, but this local explanation method was only tested on very low-dimensional datasets, and more complex problems need to be considered in the future.

For sequential decision making problems, an instance is a $\langle \texttt{state}, \texttt{action} \rangle$ pair rather than a datapoint in the feature space, and the local explanation becomes "why the action (instead of other candidate actions) is selected by the model in this state". The work in [251] proposed to use linear model to explain the local behaviour of a complex GP-evolved policy in terms of ranking the candidate actions in a given state. Instead of generating a local dataset around the state, the linear model aims to replicate the ranking (i.e., relative order of the outputs) of the candidate actions given by the explained GP policy in the state. The proposed method can find interpretable linear models with highly consistent local ranking behaviours with the explained GP models in a wide range of states.

V. Visualisation XGP Methods

The visualisation of data, algorithmic processes and results is critical in developing XAI systems. The relationships within data of relatively high dimensionalities (e.g., 10 or more features) is extremely difficult for even experienced practitioners to understand. The use of visualisation — which represents high-dimensional data in two- or three-dimensional space — is a powerful tool for explainable exploratory data analysis in today's big data era. GP is uniquely positioned to produce explainable visualisations due to its functional model structure; most visualisation methods do not provide a mapping between features in the data and the resulting visualisation. In addition to understanding the structure of data, one must also be able to understand the function of AI algorithms and models that are used on that data. Recently, visualisation of GP models and the GP process itself have helped users to better understand and explain the evolved GP models and how they were constructed [252]. In this section, we will survey (1) visualisation *by* GP, e.g., using GP for explainable data visualisation, and (2) visualisation *for* GP, e.g., using visualisation techniques to explain GP models and learning process.

A. Visualisation by GP

Data visualisation is typically performed by projecting a high-dimensional space into a two-dimensional (or, sometimes, three-dimensional) space that can be directly plotted for human interpretation. This can be performed in either an unsupervised or supervised manner.

Early work proposed the application of GP to exploring benchmark medical data [253] by evolving functional mappings using a three-tree representation. This was an unsupervised approach: the authors did not use class information despite the data being labelled, giving a "true" visualisation of the data structure. In contrast, supervised visualisation methods will purposefully skew the feature space to better separate classes. An unsupervised approach is best for understanding the natural topology of the data, whereas a supervised approach lends insight into how well-structured the data is for supervised learning purposes.

GP has been used for manifold learning [254]–[256], which learns the inherent structure within a high-dimensional dataset and represented/visualise it in a much lower-dimensional space. The proposed GP-MaL (GP for Manifold Learning) approach aims to preserve the ordering of neighbours for each instance across the original high-dimensional and mapped low-dimensional spaces. Experiments show that GP-MaL can achieve competitive performance to existing manifold learning methods including t-SNE, but produce interpretable models evidenced by further analysis on the generated trees and the visualisation results.

More recently, GP-tSNE [257] was proposed to perform unsupervised feature visualisation based on the well-established t-Distributed Stochastic Neighbour Embedding (t-SNE) method [258]. GP-tSNE uses the same objective function for measuring visualisation quality as t-SNE, but employs GP to produce an interpretable functional mapping rather than only an embedding (as in regular t-SNE). While visualisation quality was lower in some cases than standard t-SNE, the authors showed that an EMO approach could produce a range of visualisations at different levels of model (tree) complexity. Recent work has further improved the performance of GP in producing such functional mappings [256], [259].

Icke et al. [260], [261] proposed the multi-objective genetic programming projection pursuit (MOG3P) algorithm to improve the visualisation quality of feature construction in supervised (classification) problems. Their method considers three objectives: classifiability, visual interpretability, and semantic interpretability. Classifiability measures the quality of

the features for supervised learning; visual interpretability is measured using a proxy of class compactness and separability; and semantic interpretability considers the size and functional complexity of the two evolved GP trees. A user study [262] was also performed, which investigated the consistency between automated measures and human perspectives of visual and semantic interpretability. A later approach focused on classifiability and visual interpretability only, but used a number of different measures for each of these objectives [263]. When using GP for feature construction tasks, it is also possible to use the constructed features themselves as axes of a scatterplot to explain their relationships. Virgolin et al. [264] used such an approach to explaining the performance of the Gene-pool Optimal Mixing Evolutionary Algorithm (GP-GOMEA) on machine learning tasks.

B. Visualisation for GP

From an academic research perspective, understanding the evolutionary process (i.e., *how* the good GP models are found) is often even more important than understanding the models themselves. Convergence plots, population diversity plots, and other analysis methods are commonly used in GP and in other EA methods [265].

In addition to this, there have been GP-specific extensions proposed that use measures such as edit distance [266] and structural diversity [267] to measure the diversity of GP programs. Medvet et al. [268] used diversity and usage maps (DU maps), which are heat maps that visualise both the diversity of the population and the contribution of genes (e.g., sub-trees) to the overall model. Rather than taking a population-wide approach, Daida et al. [269] focused on tracking the diversity of individual GP programs across the evolutionary process. This allowed them to understand and improve the evolutionary search process. PushGP (a version of stack-based GP) was analysed by constructing the full ancestry graph for an entire algorithm run [270]. By performing a deep analysis of the ancestry graph, the authors were able to learn more about the role of different genetic operators in finding a solution, which can allow for further refinements of the PushGP algorithm.

Visualisation of the GP process has also been performed in a problem-specific manner. The Computational Evolution System (CES) — an extension to GP that incorporates greater biological realism — was applied to cancer problems, with visualisation used to identify and explore interesting patterns in the results [271]. Nguyen et al. proposed new visualisation methods for understanding the performance of GP on job shop scheduling problems using growing neural gas [272] and people-centric approaches [273]. While these methods were tested on specific tasks, they have clear potential for being applied more widely. Indeed, XGP has been applied to a plethora of real-world applications; we survey a selection of applications in the next section.

VI. Applications of XGP

XGP has successfully been applied to improve the interpretability of GP models in many real-world applications. The application domain areas include

- *primary industry* such as agriculture, aquaculture, water resources seafood and open ocean industry;
- *(bio)medical and medicine domains* such as disease diagnosis, biomarker detection and drug discovery;
- *environment and climate changes* such as global temperature change, and natural disaster prediction;
- *high-tech and high-value industry* and manufacturing such as control and automation, games and graphics, robotics, cybersecurity and sustainable energy; and
- *social and economic aspects* such as education, CPI prediction and council resource management.

With the further development, it is clear that XGP will be used much more widely to almost all major application domains and our daily life over the next ten years. Table III shows a representative sample of such real-world applications.

TABLE III
Real-world applications of XGP.

Application Domain	References
Fish weight prediction	[274]
Prediction/Detection of phytoplankton species	[275], [276]
Parkinson disease diagnosis	[68], [277], [278]
Diabetes diagnosis	[246], [249]
Skin cancer detection	[175], [176]
Disease prediction from biomedical data	[29], [157], [158], [166], [174], [177], [279]
Drug discovery and formulation	[280], [281]
Cropland field extraction assessment	[282]
Modelling global temperature change	[283]
Condition monitoring and fault analysis	[172], [284]
Dynamic system control	[285]
Waste collection routing decisions	[87], [88], [223], [224]
Industrial control	[12], [286]
Predicting pipe failures in water distribution systems	[287]
Forecasting solar power production	[288]
Control of synchronisation in the oscillator networks	[289]
Student performance prediction	[290]
Dynamic portfolio trading	[291]
Modelling cosmic structures	[292]
Learning transformation functions in computer graphics imaging	[293]

VII. Challenges and Future Directions

The area of XGP is still in its infancy, and there are many challenges to be addressed, leading to future research directions in this area.

The notion of "interpretability" is inherently subjective: defining and understanding this concept requires insights from psychology and the wider social sciences. The limited existing XGP approaches typically define quantitative proxy interpretability metrics such as the number of nodes, tree depth, model complexity, and the number of meaningful building blocks. However, there is no robust literature on how well these proxy metrics reflect the "real" interpretability for human users. The tendency of many GP researchers to equate "small"

with "interpretable" lacks nuance [294]. The interpretability of an XGP (or any XAI) system depends not only on one's technical expertise, but also on their cultural background, world views, and personal priorities in understanding different factors in decision making. A benchmark of explanatory methods was developed to facilitate the evaluation of quality of post-hoc explanations [295]. Some initial work has sought to learn proxy metrics based on questionnaires (e.g., [244]), but significantly more work is needed — especially in automating user-specific explanations.

There is also a trade-off between a model's accuracy/effectiveness and interpretability [51]; there is no single model that can be both maximally effective and optimally interpretable for most reasonably difficult problems. Attempts to balance these two objectives are easily thwarted by local optima: it is, in general, much harder to optimise a small (interpretable) function to be more performant than a large function with many free parameters. The post-hoc simplification of complex GP trees (at the end of the evolutionary process) is reasonably widespread in the literature, but such techniques are still ultimately constrained by the semantic tree structure. Furthermore, if the interpretability is defined by a range of different factors (e.g., program size, number of features used, and model structure) that are potentially conflicting with each other, the problem will become a complex many-objective optimisation problem, which is very challenging to solve.

VIII. Conclusions

This survey reviews a recent emerging area in GP and evolutionary machine learning: XAI by GP (XGP). This paper discusses both the recent studies that explicitly consider improving the interpretability of the GP-evolved models as well as earlier studies that can potentially evolve more interpretable GP models as a byproduct of their algorithm design. The surveyed papers have also shown the clear potential of GP for contributing to the wider XAI area. Firstly, GP itself has a relatively interpretable program structure that combines symbolic and computational AI (both logical and algebraic operators in the function set). Secondly, as an evolutionary computation approach, GP uses a population-based search process to evolve programs, which is ideal to tackle the challenges of the trade-off between effectiveness and interpretability. Finally, the fitness landscape of many formulations of interpretability is poorly-defined and is likely to be discontinuous and/or non-differentiable. GP does not rely on the availability of gradient information or any other strong mathematical assumptions: thus it is capable of searching in the space of interpretability effectively. This is a promising research area with great potential in contributing XAI and their real-world applications.

References

[1] W. Banzhaf, F. D. Francone, R. E. Keller, and P. Nordin, *Genetic Programming: An Introduction: On the Automatic Evolution of Computer Programs and Its Applications*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1998.

[2] J. R. Koza and J. R. Koza, *Genetic programming: on the programming of computers by means of natural selection*. MIT press, 1992, vol. 1.

[3] H. Al-Sahaf, Y. Bi, Q. Chen, A. Lensen, Y. Mei, Y. Sun, B. Tran, B. Xue, and M. Zhang, "A survey on evolutionary machine learning," *J. R. Soc. N. Z.*, vol. 49, no. 2, pp. 205–228, Apr. 2019.

[4] P. G. Espejo, S. Ventura, and F. Herrera, "A survey on the application of genetic programming to classification," *IEEE Trans. Syst. Man Cybern. Part C Appl. Rev.*, vol. 40, no. 2, pp. 121–144, 2010.

[5] U. Bhowan, M. Johnston, M. Zhang, and X. Yao, "Evolving diverse ensembles using genetic programming for classification with unbalanced data," *IEEE Trans. Evol. Comput.*, vol. 17, no. 3, pp. 368–386, 2013.

[6] N. Q. Uy, N. X. Hoai, M. O'Neill, R. I. McKay, and E. Galván-López, "Semantically-based crossover in genetic programming: Application to real-valued symbolic regression," *Genetic Programming and Evolvable Machines*, vol. 12, no. 2, pp. 91–119, Jun. 2011.

[7] E. Vladislavleva, G. Smits, and D. den Hertog, "Order of Nonlinearity as a Complexity Measure for Models Generated by Symbolic Regression via Pareto Genetic Programming," *IEEE Transactions on Evolutionary Computation*, vol. 13, no. 2, pp. 333–349, Apr. 2009.

[8] E. Naredo and L. Trujillo, "Searching for novel clustering programs," in *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO)*, 2013, pp. 1093–1100.

[9] A. Lensen, B. Xue, and M. Zhang, "Genetic programming for evolving similarity functions for clustering: Representations and analysis," *Evol. Comput.*, vol. 28, no. 4, pp. 531–561, Oct. 2019.

[10] H. D. Mettler, M. Schmidt, W. Senn, M. A. Petrovici, and J. Jordan, "Evolving neuronal plasticity rules using cartesian genetic programming," in *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, 2021, pp. 285–286.

[11] D. Hein, S. Udluft, and T. A. Runkler, "Interpretable policies for reinforcement learning by genetic programming," *Engineering Applications of Artificial Intelligence*, vol. 76, pp. 158–169, 2018.

[12] ——, "Interpretable policies for reinforcement learning by genetic programming," *Engineering Applications of Artificial Intelligence*, vol. 76, pp. 158–169, Nov. 2018.

[13] J. D. Co-Reyes, Y. Miao, D. Peng, E. Real, S. Levine, Q. V. Le, H. Lee, and A. Faust, "Evolving reinforcement learning algorithms," *arXiv preprint arXiv:2101.03958*, 2021.

[14] J. Branke, S. Nguyen, C. W. Pickardt, and M. Zhang, "Automated Design of Production Scheduling Heuristics: A Review," *IEEE Transactions on Evolutionary Computation*, vol. 20, no. 1, pp. 110–124, Feb. 2016.

[15] S. Nguyen, Y. Mei, and M. Zhang, "Genetic programming for production scheduling: A survey with a unified framework," *Complex & Intelligent Systems*, vol. 3, no. 1, pp. 41–66, Mar. 2017.

[16] W. Samek, G. Montavon, A. Vedaldi, L. K. Hansen, and K.-R. Müller, Eds., *Explainable AI: Interpreting, Explaining and Visualizing Deep Learning*. Cham: Springer International Publishing, 2019, vol. 11700.

[17] S. Lapuschkin, S. Wäldchen, A. Binder, G. Montavon, W. Samek, and K.-R. Müller, "Unmasking clever hans predictors and assessing what machines really learn," *Nature communications*, vol. 10, no. 1, pp. 1–8, 2019.

[18] M. T. Ribeiro, S. Singh, and C. Guestrin, ""why should I trust you?": Explaining the predictions of any classifier," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*. ACM, 2016, pp. 1135–1144.

[19] J. Larson, S. Mattu, L. Kirchner, and J. Angwin, "How we analyzed the compas recidivism algorithm," *ProPublica*, 2016.

[20] Council of European Union, "Council regulation (EU) no 679/2016," 2016, https://eur-lex.europa.eu/eli/reg/2016/679/oj.

[21] R. Hamon, H. Junklewitz, I. Sanchez, G. Malgieri, and P. De Hert, "Bridging the gap between ai and explainability in the gdpr: towards trustworthiness-by-design in automated decision-making," *IEEE Computational Intelligence Magazine*, vol. 17, no. 1, pp. 72–85, 2022.

[22] R. Confalonieri, L. Coba, B. Wagner, and T. R. Besold, "A historical perspective of explainable Artificial Intelligence," *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, vol. 11, no. 1, p. e1391, Jan. 2021.

[23] F. K. Došilović, M. Brčić, and N. Hlupić, "Explainable artificial intelligence: A survey," in *2018 41st International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*, May 2018, pp. 0210–0215.

[24] A. Adadi and M. Berrada, "Peeking Inside the Black-Box: A Survey on Explainable Artificial Intelligence (XAI)," *IEEE Access*, vol. 6, pp. 52 138–52 160, 2018.

[25] M. Du, N. Liu, and X. Hu, "Techniques for interpretable machine learning," *Communications of the ACM*, vol. 63, no. 1, pp. 68–77, Dec. 2019.

[26] A. Barredo Arrieta, N. Díaz-Rodríguez, J. Del Ser, A. Bennetot, S. Tabik, A. Barbado, S. Garcia, S. Gil-Lopez, D. Molina, R. Benjamins, R. Chatila, and F. Herrera, "Explainable Artificial Intelligence

(XAI): Concepts, taxonomies, opportunities and challenges toward responsible AI," *Information Fusion*, vol. 58, pp. 82–115, Jun. 2020.
[27] P. Linardatos, V. Papastefanopoulos, and S. Kotsiantis, "Explainable ai: A review of machine learning interpretability methods," *Entropy*, vol. 23, no. 1, p. 18, 2021.
[28] M. Brameier and W. Banzhaf, *Linear genetic programming*. Springer, 2007, vol. 1.
[29] T. Hu, "Can genetic programming perform explainable machine learning for bioinformatics?" in *Genetic Programming Theory and Practice XVII*, W. Banzhaf, E. Goodman, L. Sheneman, L. Trujillo, and B. Worzel, Eds. East Lansing, MI, USA: Springer, 16-19 May 2019, pp. 63–77.
[30] R. Poli, "Evolution of graph-like programs with parallel distributed genetic programming," in *International Conference on Genetic Algorithms*, 1997, pp. 346–353.
[31] J. F. Miller, "Cartesian genetic programming: its status and future," *Genetic Programming and Evolvable Machines*, vol. 21, no. 1, pp. 129–168, 2020.
[32] D. J. Montana, "Strongly Typed Genetic Programming," *Evolutionary Computation*, vol. 3, no. 2, pp. 199–230, Jun. 1995.
[33] R. I. McKay, N. X. Hoai, P. A. Whigham, Y. Shan, and M. O'Neill, "Grammar-based Genetic Programming: A survey," *Genetic Programming and Evolvable Machines*, vol. 11, no. 3-4, pp. 365–396, Sep. 2010.
[34] M. O'Neill and C. Ryan, "Grammatical evolution," *IEEE Transactions on Evolutionary Computation*, vol. 5, no. 4, pp. 349–358, 2001.
[35] J. R. Quinlan, *C4.5: programs for machine learning*. Elsevier, 2014.
[36] J. R. Koza, "Concept formation and decision tree induction using the genetic programming paradigm," in *Parallel Problem Solving from Nature*, H.-P. Schwefel and R. Männer, Eds. Berlin/Heidelberg: Springer-Verlag, 1991, vol. 496, pp. 124–128.
[37] M. C. J. Bot and W. B. Langdon, "Application of Genetic Programming to Induction of Linear Classification Trees," in *Genetic Programming*, R. Poli, W. Banzhaf, W. B. Langdon, J. Miller, P. Nordin, and T. C. Fogarty, Eds. Berlin, Heidelberg: Springer, 2000, pp. 247–258.
[38] H. Zhao, "A multi-objective genetic programming approach to developing Pareto optimal decision trees," *Decision Support Systems*, vol. 43, no. 3, pp. 809–826, Apr. 2007.
[39] A. Shali, M. R. Kangavari, and B. Bina, "Using genetic programming for the induction of oblique decision trees," in *International Conference on Machine Learning and Applications*. IEEE, 2007, pp. 38–43.
[40] I. De Falco, A. Della Cioppa, and E. Tarantino, "Discovering interesting classification rules with genetic programming," *Applied Soft Computing*, vol. 1, no. 4, pp. 257–269, May 2002.
[41] R. Mazouni and A. Rahmoun, "Agge: A novel method to automatically generate rule induction classifiers using grammatical evolution," in *Intelligent Distributed Computing VIII*. Springer, 2015, pp. 279–288.
[42] H. Shahrzad, B. Hodjat, and R. Miikkulainen, "Evolving explainable rule sets," in *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, 2022, pp. 1779–1784.
[43] A. S. Koshiyama, M. M. Vellasco, and R. Tanscheit, "Gpfis-class: A genetic fuzzy system based on genetic programming for classification problems," *Applied Soft Computing*, vol. 37, pp. 561–571, 2015.
[44] A. R. dos Santos, J. L. M. do Amaral, C. A. R. Soares, and A. V. de Barros, "Multi-objective fuzzy pattern trees," in *IEEE International Conference on Fuzzy Systems*. IEEE, 2018, pp. 1–6.
[45] A. Murphy, M. S. Ali, D. M. Dias, J. L. Amaral, E. Naredo, and C. Ryan, "Grammar-based fuzzy pattern trees for classification problems." in *IJCCI*, 2020, pp. 71–80.
[46] A. Murphy, G. Murphy, J. Amaral, D. MotaDias, E. Naredo, and C. Ryan, "Towards incorporating human knowledge in fuzzy pattern tree evolution," in *European Conference on Genetic Programming*. Springer, 2021, pp. 66–81.
[47] A. Murphy, G. Murphy, D. M. Dias, J. Amaral, E. Naredo, and C. Ryan, "Human in the loop fuzzy pattern tree evolution," *SN Computer Science*, vol. 3, no. 2, pp. 1–14, 2022.
[48] Z. C. Lipton, "The mythos of model interpretability," *Commun. ACM*, vol. 61, no. 10, pp. 35–43, 2018.
[49] F. Doshi-Velez and B. Kim, "Towards A Rigorous Science of Interpretable Machine Learning," *arXiv:1702.08608 [cs, stat]*, Feb. 2017.
[50] T. Miller, "Explanation in artificial intelligence: Insights from the social sciences," *Artificial Intelligence*, vol. 267, pp. 1–38, Feb. 2019.
[51] D. Gunning and D. Aha, "Darpa's explainable artificial intelligence (xai) program," *AI Magazine*, vol. 40, no. 2, pp. 44–58, 2019.
[52] W. J. Murdoch, C. Singh, K. Kumbier, R. Abbasi-Asl, and B. Yu, "Definitions, methods, and applications in interpretable machine learning," *PNAS*, vol. 116, no. 44, pp. 22 071–22 080, Oct. 2019.
[53] C. Molnar, *Interpretable machine learning*. Lulu. com, 2020.
[54] C. Rudin, "Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead," *Nature Machine Intelligence*, vol. 1, no. 5, pp. 206–215, 2019.
[55] L. Semenova, C. Rudin, and R. Parr, "On the existence of simpler machine learning models," in *ACM Conference on Fairness, Accountability, and Transparency*, 2022, pp. 1827–1858.
[56] S. Luke and L. Panait, "A Comparison of Bloat Control Methods for Genetic Programming," *Evol. Comput.*, vol. 14, no. 3, pp. 309–344, Sep. 2006.
[57] W. B. Langdon, "Quadratic bloat in genetic programming," in *Proceedings of the 2nd Annual Conference on Genetic and Evolutionary Computation*. Citeseer, 2000, pp. 451–458.
[58] S. Gustafson, A. Ekart, E. Burke, and G. Kendall, "Problem difficulty and code growth in genetic programming," *Genetic Programming and Evolvable Machines*, vol. 5, no. 3, pp. 271–290, 2004.
[59] M. J. Streeter, "The root causes of code growth in genetic programming," in *European Conference on Genetic Programming*, 2003, pp. 443–454.
[60] S. Dignum and R. Poli, "Generalisation of the limiting distribution of program sizes in tree-based genetic programming and analysis of its effects on bloat," in *Proceedings of the 9th annual conference on Genetic and evolutionary computation*, 2007, pp. 1588–1595.
[61] S. Silva and E. Costa, "Dynamic limits for bloat control in genetic programming and a review of past and current bloat theories," *Genet Program Evolvable Mach*, vol. 10, no. 2, pp. 141–179, Jun. 2009.
[62] S. Silva and J. Almeida, "Dynamic maximum tree depth — a simple technique for avoiding bloat in tree- based gp," in *Genetic and evolutionary computation conference*. Springer, 2003, pp. 1776–1787.
[63] S. Silva and E. Costa, "Dynamic limits for bloat control — variations on size and depth," in *Genetic and Evolutionary Computation Conference*. Springer, 2004, pp. 666–677.
[64] M. Virgolin, T. Alderliesten, C. Witteveen, and P. A. Bosman, "Scalable genetic programming by gene-pool optimal mixing and input-space entropy-based building-block learning," in *Proceedings of the Genetic and Evolutionary Computation Conference*, 2017, pp. 1041–1048.
[65] M. Virgolin, T. Alderliesten, C. Witteveen, and P. A. N. Bosman, "Improving Model-Based Genetic Programming for Symbolic Regression of Small Expressions," *Evolutionary Computation*, vol. 29, no. 2, pp. 211–237, 06 2021.
[66] J. Miller, "What bloat? cartesian genetic programming on boolean problems," in *ACM Genetic and Evolutionary Computation Conference Late Breaking Papers*, 2001, pp. 295–302.
[67] A. J. Turner and J. F. Miller, "Cartesian genetic programming: Why no bloat?" in *European Conference on Genetic Programming*. Springer, 2014, pp. 222–233.
[68] A. Parziale, R. Senatore, A. Della Cioppa, and A. Marcelli, "Cartesian genetic programming for diagnosis of parkinson disease through handwriting analysis: Performance vs. interpretability issues," *Artificial Intelligence in Medicine*, vol. 111, p. 101984, 2021.
[69] R. Poli, "A simple but theoretically-motivated method to control bloat in genetic programming," in *European Conference on Genetic Programming*. Springer, 2003, pp. 204–217.
[70] T. Soule and J. A. Foster, "Effects of code growth and parsimony pressure on populations in genetic programming," *Evolutionary computation*, vol. 6, no. 4, pp. 293–309, 1998.
[71] C. Ryan, "Pygmies and civil servants," in *Advances in Genetic Programming*, 1994, pp. 243–263.
[72] H. Iba, H. de Garis, and T. Sato, "Genetic programming using a minimum description length principle," in *Advances in genetic programming*, 1994, pp. 265–284.
[73] P. Nordin and W. Banzhaf, "Complexity compression and evolution," in *Proceedings of the 6th International Conference on Genetic Algorithms*, 1995, p. 310–317.
[74] M. J. Cavaretta and K. Chellapilla, "Data mining using genetic programming: The implications of parsimony on generalization error," in *Proceedings of the 1999 Congress on Evolutionary Computation*, vol. 2. IEEE, 1999, pp. 1330–1337.
[75] B.-T. Zhang and H. Mühlenbein, "Balancing accuracy and parsimony in genetic programming," *Evolutionary Computation*, vol. 3, no. 1, pp. 17–38, 1995.
[76] T. Kalganova and J. Miller, "Evolving more efficient digital circuits by allowing circuit layout evolution and multi-objective fitness," in *Proceedings of the first NASA/DoD workshop on evolvable hardware*. IEEE, 1999, pp. 54–63.

[77] S. Bleuler, M. Brack, L. Thiele, and E. Zitzler, "Multiobjective genetic programming: Reducing bloat using SPEA2," in *Proceedings of the 2001 Congress on Evolutionary Computation*, vol. 1. Seoul, South Korea: IEEE, 2001, pp. 536–543.

[78] A. Ekárt and S. Z. Németh, "Selection Based on the Pareto Nondomination Criterion for Controlling Code Growth in Genetic Programming," *Genetic Programming and Evolvable Machines*, vol. 2, no. 1, pp. 61–73, Mar. 2001.

[79] E. D. de Jong and J. B. Pollack, "Multi-Objective Methods for Tree Size Control," *Genetic Programming and Evolvable Machines*, vol. 4, no. 3, pp. 211–233, Sep. 2003.

[80] Y. Bernstein, Xiaodong Li, V. Ciesielski, and A. Song, "Multiobjective parsimony enforcement for superior generalisation performance," in *Proceedings of the 2004 Congress on Evolutionary Computation (IEEE Cat. No.04TH8753)*, vol. 1, Jun. 2004, pp. 83–89 Vol.1.

[81] D. Parrott, X. Li, and V. Ciesielski, "Multi-objective techniques in genetic programming for evolving classifiers," in *2005 IEEE Congress on Evolutionary Computation*, vol. 2, pp. 1141–1148 Vol. 2.

[82] S. Luke and L. Panait, "A comparison of bloat control methods for genetic programming," *Evol. Comput.*, vol. 14, no. 3, pp. 309–344, 2006.

[83] F. Neumann, "Computational complexity analysis of multi-objective genetic programming," in *Proceedings of the 14th annual conference on Genetic and evolutionary computation*, 2012, pp. 799–806.

[84] J. Liang, Y. Liu, and Y. Xue, "Preference-driven pareto front exploitation for bloat control in genetic programming," *Applied Soft Computing*, vol. 92, p. 106254, 2020.

[85] J. Liang and Y. Xue, "Multi-objective memetic algorithms with tree-based genetic programming and local search for symbolic regression," *Neural Processing Letters*, vol. 53, no. 3, pp. 2197–2219, 2021.

[86] S. Wang, Y. Mei, J. Park, and M. Zhang, "A two-stage genetic programming hyper-heuristic for uncertain capacitated arc routing problem," in *Proceedings of IEEE Symposium Series on Computational Intelligence*. IEEE, dec 2019, pp. 1606–1613.

[87] S. Wang, Y. Mei, and M. Zhang, "Towards interpretable routing policy: A two stage multi-objective genetic programming approach with feature selection for uncertain capacitated arc routing problem," in *Proceedings of the IEEE Symposium Series on Computational Intelligence*. IEEE, dec 2020, pp. 2399–2406.

[88] ——, "A two-stage multi-objective genetic programming with archive for uncertain capacitated arc routing problem," in *Proceedings of the ACM Genetic and Evolutionary Computation Conference*. ACM, jul 2021, pp. 287–295.

[89] ——, "A multi-objective genetic programming hyper-heuristic approach to uncertain capacitated arc routing problems," in *Proceedings of the IEEE World Congress on Computational Intelligence*. IEEE, mar 2020, pp. 1–8.

[90] ——, "A multi-objective genetic programming approach with self-adaptive alpha dominance to uncertain capacitated arc routing problem," in *Proceedings of the IEEE Congress on Evolutionary Computation*. IEEE, jul 2021, pp. 636–643.

[91] ——, "A multi-objective genetic programming algorithm with α dominance and archive for uncertain capacitated arc routing problem," *IEEE Transactions on Evolutionary Computation*, jul 2022.

[92] S. Luke and L. Panait, "Lexicographic parsimony pressure," in *Proceedings of the 4th Annual Conference on Genetic and Evolutionary Computation*, 2002, pp. 829–836.

[93] T. Kötzing, J. G. Lagodzinski, J. Lengler, and A. Melnichenko, "Destructiveness of lexicographic parsimony pressure and alleviation by a concatenation crossover in genetic programming," *Theoretical Computer Science*, vol. 816, pp. 96–113, 2020.

[94] S. Luke and L. Panait, "Fighting Bloat with Nonparametric Parsimony Pressure," in *Parallel Problem Solving from Nature — PPSN VII*, J. J. M. Guervós, P. Adamidis, H.-G. Beyer, H.-P. Schwefel, and J.-L. Fernández-Villacañas, Eds. Springer Berlin Heidelberg, 2002, pp. 411–421.

[95] R. Poli and W. B. Langdon, "Genetic programming with one-point crossover," in *Soft Computing in Engineering Design and Manufacturing*. Springer, 1998, pp. 180–189.

[96] ——, "Schema theory for genetic programming with one-point crossover and point mutation," *Evolutionary Computation*, vol. 6, no. 3, pp. 231–252, 1998.

[97] W. B. Langdon, "Size fair and homologous tree genetic programming crossovers," *Genetic programming and evolvable machines*, vol. 1, no. 1/2, pp. 95–119, 2000.

[98] P. Nordin, W. Banzhaf, and F. D. Francone, "Efficient evolution of machine code for cisc architectures using instruction blocks and homologous crossover," *Advances in genetic programming*, vol. 3, pp. 275–299, 1999.

[99] M. D. Platel, M. Clergue, and P. Collard, "Maximum homologous crossover for linear genetic programming," in *European Conference on Genetic Programming*. Springer, 2003, pp. 194–203.

[100] J. Page, R. Poli, and W. B. Langdon, "Mutation in genetic programming: a preliminary study," in *European Conference on Genetic Programming*. Springer, 1999, pp. 39–48.

[101] W. B. Langdon, T. Soule, R. Poli, and J. A. Foster, "The evolution of size and shape," *Advances in genetic programming*, vol. 3, pp. 163–190, 1999.

[102] E. Alfaro-Cid, J. Merelo, F. F. de Vega, A. I. Esparcia-Alcázar, and K. Sharman, "Bloat control operators and diversity in genetic programming: A comparative study," *Evolutionary Computation*, vol. 18, no. 2, pp. 305–332, 2010.

[103] M. I. Heywood and A. N. Zincir-Heywood, "Dynamic page based crossover in linear genetic programming," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 32, no. 3, pp. 380–388, 2002.

[104] W. Ashlock and D. Ashlock, "Single parent genetic programming," in *2005 IEEE Congress on Evolutionary Computation*, vol. 2, 2005, pp. 1172–1179.

[105] S. Wang, Y. Mei, M. Zhang, and X. Yao, "Genetic programming with niching for uncertain capacitated arc routing problem," *IEEE Transactions on Evolutionary Computation*, pp. 1–1, 2021.

[106] P. A. Whigham and G. Dick, "Implicitly Controlling Bloat in Genetic Programming," *IEEE Transactions on Evolutionary Computation*, vol. 14, no. 2, pp. 173–190, Apr. 2010.

[107] G. Dick and P. A. Whigham, "Controlling bloat through parsimonious elitist replacement and spatial structure," in *European Conference on Genetic Programming*. Springer, 2013, pp. 13–24.

[108] L. Trujillo, L. Muñoz, E. Galván-López, and S. Silva, "Neat Genetic Programming: Controlling bloat naturally," *Information Sciences*, vol. 333, pp. 21–43, Mar. 2016.

[109] P. Juárez-Smith, L. Trujillo, M. García-Valdez, F. Fernández de Vega, and F. Chávez, "Local search in speciation-based bloat control for genetic programming," *Genet. Program. Evolvable Mach.*, vol. 20, no. 3, pp. 351–384, 2019.

[110] F. F. d. Vega, G. Olague, F. Chávez, D. Lanza, W. Banzhaf, and E. Goodman, "It is time for new perspectives on how to fight bloat in gp," in *Genetic Programming Theory and Practice XVII*. Springer, 2020, pp. 25–38.

[111] S. Silva, S. Dignum, and L. Vanneschi, "Operator equalisation for bloat free genetic programming and a survey of bloat control methods," *Genetic Programming and Evolvable Machines*, vol. 13, no. 2, pp. 197–238, 2012.

[112] S. Dignum and R. Poli, "Operator equalisation and bloat free gp," in *Genetic Programming*, M. O'Neill, L. Vanneschi, S. Gustafson, A. I. Esparcia Alcázar, I. De Falco, A. Della Cioppa, and E. Tarantino, Eds., 2008, pp. 110–121.

[113] S. Silva and L. Vanneschi, "The Importance of Being Flat–Studying the Program Length Distributions of Operator Equalisation," in *Genetic Programming Theory and Practice IX*, R. Riolo, E. Vladislavleva, and J. H. Moore, Eds. New York, NY: Springer New York, 2011, pp. 211–233.

[114] S. Silva and S. Dignum, "Extending operator equalisation: Fitness based self adaptive length distribution for bloat free gp," in *European Conference on Genetic Programming*. Springer, 2009, pp. 159–170.

[115] S. Silva and L. Vanneschi, "Operator equalisation, bloat and overfitting: a study on human oral bioavailability prediction," in *Proceedings of the 11th Annual conference on Genetic and evolutionary computation*, 2009, pp. 1115–1122.

[116] N. Javed, F. Gobet, and P. Lane, "Simplification of genetic programs: a literature survey," *Data Mining and Knowledge Discovery*, pp. 1–22, 2022.

[117] D. C. Hooper and N. S. Flann, "Improving the accuracy and robustness of genetic programming through expression simplification," in *Proceedings of the 1st annual conference on genetic programming*, 1996, pp. 428–428.

[118] A. Ekart, "Shorter fitness preserving genetic programs," in *European Conference on Artificial Evolution*. Springer, 1999, pp. 73–83.

[119] M. Zhang and P. Wong, "Explicitly simplifying evolved genetic programs during evolution," *International Journal of Computational Intelligence and Applications*, vol. 07, no. 02, pp. 201–232, Jun. 2008.

[120] A. Song, D. Chen, and M. Zhang, "Bloat control in genetic programming by evaluating contribution of nodes," in *Proceedings of the 11th*

Annual conference on Genetic and evolutionary computation, 2009, pp. 1893–1894.

[121] Q. U. Nguyen and T. H. Chu, "Semantic approximation for reducing code bloat in genetic programming," *Swarm and Evolutionary Computation*, vol. 58, p. 100729, 2020.

[122] S. Panda and Y. Mei, "Genetic programming with algebraic simplification for dynamic job shop scheduling," in *2021 IEEE Congress on Evolutionary Computation*. IEEE, 2021, pp. 1848–1855.

[123] T. Helmuth, N. F. McPhee, E. Pantridge, and L. Spector, "Improving generalization of evolved programs through automatic simplification," in *Proceedings of the Genetic and Evolutionary Computation Conference*, 2017, p. 937–944.

[124] P. Rockett, "Pruning of genetic programming trees using permutation tests," *Evolutionary Intelligence*, vol. 13, no. 4, pp. 649–661, 2020.

[125] M. Zhang, Y. Zhang, and W. Smart, "Program simplification in genetic programming for object classification," in *International Conference on Knowledge-Based and Intelligent Information and Engineering Systems*. Springer, 2005, pp. 988–996.

[126] D. Kinzett, M. Johnston, and M. Zhang, "Numerical simplification for bloat control and analysis of building blocks in genetic programming," *Evolutionary Intelligence*, vol. 2, no. 4, pp. 151–168, Dec. 2009.

[127] M. Johnston, T. Liddle, and M. Zhang, "A relaxed approach to simplification in genetic programming," in *European Conference on Genetic Programming*. Springer, 2010, pp. 110–121.

[128] N. Javed and F. Gobet, "On-the-fly simplification of genetic programming models," in *Proceedings of the 36th Annual ACM Symposium on Applied Computing*. New York, NY, USA: Association for Computing Machinery, Mar. 2021, pp. 464–471.

[129] J. F. Smith, "Genetic program based data mining for fuzzy decision trees," in *International Conference on Intelligent Data Engineering and Automated Learning*. Springer, 2004, pp. 464–470.

[130] K. Murano, S. Yoshida, T. Harada, and R. Thawonmas, "A study on multimodal genetic programming introducing program simplification," in *Joint 10th international conference on soft computing and intelligent systems and 19th international symposium on advanced intelligent systems*. IEEE, 2018, pp. 109–114.

[131] B. Burlacu, L. Kammerer, M. Affenzeller, and G. Kronberger, "Hash-based tree similarity and simplification in genetic programming for symbolic regression," in *International conference on computer aided systems theory*. Springer, 2019, pp. 361–369.

[132] L. Spector and A. Robinson, "Genetic programming and autoconstructive evolution with the push programming language," *Genetic Programming and Evolvable Machines*, vol. 3, no. 1, pp. 7–40, 2002.

[133] M. Naoki, B. McKay, N. Xuan, E. Daryl, and S. Takeuchi, "A new method for simplifying algebraic expressions in genetic programming called equivalent decision simplification," in *International workconference on artificial neural networks*. Springer, 2009, pp. 171–178.

[134] T. H. Chu and Q. U. Nguyen, "Reducing code bloat in genetic programming based on subtree substituting technique," in *Asia Pacific symposium on intelligent and evolutionary systems*. IEEE, 2017, pp. 25–30.

[135] T. H. Chu, Q. U. Nguyen, and V. L. Cao, "Semantics based substituting technique for reducing code bloat in genetic programming," in *Proceedings of the ninth international symposium on information and communication technology*, 2018, pp. 77–83.

[136] D. Jackson, "The identification and exploitation of dormancy in genetic programming," *Genetic Programming and Evolvable Machines*, vol. 11, no. 1, pp. 89–121, 2010.

[137] Y. Mei, M. Zhang, and S. Nyugen, "Feature selection in evolving job shop dispatching rules with genetic programming," in *Proceedings of the Genetic and Evolutionary Computation Conference*. ACM, 2016, pp. 365–372.

[138] G. F. Smits and M. Kotanchek, "Pareto-front exploitation in symbolic regression," in *Genetic programming theory and practice II*. Springer, 2005, pp. 283–299.

[139] M. Keijzer and J. Foster, "Crossover bias in genetic programming," in *Genetic Programming*. Springer Berlin Heidelberg, 2007, pp. 33–44.

[140] M. Kommenda, G. Kronberger, M. Affenzeller, S. M. Winkler, and B. Burlacu, "Evolving simple symbolic regression models by multi-objective genetic programming," in *Genetic Programming Theory and Practice XIII*. Springer, 2016, pp. 1–19.

[141] C. Raymond, Q. Chen, B. Xue, and M. Zhang, "Adaptive weighted splines: A new representation to genetic programming for symbolic regression," in *Proceedings of the 2020 Genetic and Evolutionary Computation Conference*, 2020, p. 1003–1011.

[142] W. La Cava, K. Danai, and L. Spector, "Inference of compact nonlinear dynamic models by epigenetic local search," *Engineering Applications of Artificial Intelligence*, vol. 55, pp. 292–306, 2016.

[143] E. J. Vladislavleva, G. F. Smits, and D. den Hertog, "Order of nonlinearity as a complexity measure for models generated by symbolic regression via pareto genetic programming," *IEEE Transactions on Evolutionary Computation*, vol. 13, no. 2, pp. 333–349, 2009.

[144] L. Vanneschi, M. Castelli, and S. Silva, "Measuring bloat, overfitting and functional complexity in genetic programming," in *Proceedings of the 12th annual conference on Genetic and evolutionary computation*, 2010, pp. 877–884.

[145] L. Vanneschi and M. Castelli, "Soft target and functional complexity reduction: A hybrid regularization method for genetic programming," *Expert Systems with Applications*, vol. 177, p. 114929, 2021.

[146] J. Ni and P. Rockett, "Tikhonov regularization as a complexity measure in multiobjective genetic programming," *IEEE Transactions on Evolutionary Computation*, vol. 19, no. 2, pp. 157–166, 2015.

[147] Q. Chen, B. Xue, L. Shang, and M. Zhang, "Improving generalisation of genetic programming for symbolic regression with structural risk minimisation," in *Proceedings of the Genetic and Evolutionary Computation Conference 2016*, 2016, pp. 709–716.

[148] Q. Chen, M. Zhang, and B. Xue, "Structural risk minimization-driven genetic programming for enhancing generalization in symbolic regression," *IEEE Transactions on Evolutionary Computation*, vol. 23, no. 4, pp. 703–717, 2019.

[149] B. Al-Helali, Q. Chen, B. Xue, and M. Zhang, "Hessian complexity measure for genetic programming-based imputation predictor selection in symbolic regression with incomplete data," in *Genetic Programming*, 2020, pp. 1–17.

[150] C. Raymond, Q. Chen, B. Xue, and M. Zhang, "Genetic programming with rademacher complexity for symbolic regression," in *2019 IEEE Congress on Evolutionary Computation*, 2019, pp. 2657–2664.

[151] Q. Chen, B. Xue, and M. Zhang, "Rademacher complexity for enhancing the generalization of genetic programming for symbolic regression," *IEEE Transactions on Cybernetics*, pp. 1–14, 2020.

[152] A. S. Sambo, R. M. A. Azad, Y. Kovalchuk, V. P. Indramohan, and H. Shah, "Time control or size control? reducing complexity and improving accuracy of genetic programming models," in *Genetic Programming*, 2020, pp. 195–210.

[153] Y. Liu and T. Khoshgoftaar, "Reducing overfitting in genetic programming models for software quality classification," in *Eighth IEEE International Symposium on High Assurance Systems Engineering, 2004. Proceedings.* IEEE Computer Society, 2004, pp. 56–65.

[154] I. Gonçalves, S. Silva, J. B. Melo, and J. M. B. Carreiras, "Random sampling technique for overfitting control in genetic programming," in *Genetic Programming*, 2012, pp. 218–229.

[155] I. Gonçalves and S. Silva, "Balancing learning and overfitting in genetic programming with interleaved sampling of training data," in *European Conference on Genetic Programming*, 2013, pp. 73–84.

[156] J. Fitzgerald, R. M. A. Azad, and C. Ryan, "A bootstrapping approach to reduce over-fitting in genetic programming," in *Proceedings of the 15th Annual Conference Companion on Genetic and Evolutionary Computation*, 2013, p. 1113–1120.

[157] B. Tran, B. Xue, and M. Zhang, "Using feature clustering for gp-based feature construction on high-dimensional data," in *European Conference on Genetic Programming*. Springer, 2017, pp. 210–226.

[158] ——, "Genetic programming for feature construction and selection in classification on high-dimensional data," *Memetic Computing*, vol. 8, no. 1, pp. 3–15, 2016.

[159] G. Dick, "Sensitivity-like analysis for feature selection in genetic programming." New York, NY, USA: ACM, 2017, p. 401–408.

[160] Q. Chen, M. Zhang, and B. Xue, "Feature selection to improve generalization of genetic programming for high-dimensional symbolic regression," *IEEE Transactions on Evolutionary Computation*, vol. 21, no. 5, pp. 792–806, 2017.

[161] Y. Mei, S. Nguyen, B. Xue, and M. Zhang, "An efficient feature selection algorithm for evolving job shop scheduling rules with genetic programming," *IEEE Transactions on Emerging Topics in Computational Intelligence*, vol. 1, no. 5, pp. 339–353, 2017.

[162] D. Yska, Y. Mei, and M. Zhang, "Feature construction in genetic programming hyper-heuristic for dynamic flexible job shop scheduling," in *Proceedings of the Genetic and Evolutionary Computation Conference Companion*. ACM, jul 2018, pp. 149–150.

[163] F. Zhang, Y. Mei, and M. Zhang, "A two-stage genetic programming hyper-heuristic approach with feature selection for dynamic flexible job shop scheduling," in *Proceedings of the Genetic and Evolutionary Computation Conference*. IEEE, jul 2019, pp. 347–355.

[164] F. Zhang, Y. Mei, S. Nguyen, and M. Zhang, "Evolving scheduling heuristics via genetic programming with feature selection in dynamic flexible job shop scheduling," *IEEE Transactions on Cybernetics*, vol. 51, no. 4, pp. 1797–1811, apr 2021.

[165] G. S. I. Aldeia and F. O. de França, *Measuring Feature Importance of Symbolic Regression Models Using Partial Effects*. New York, NY, USA: Association for Computing Machinery, 2021, p. 750–758.

[166] S. Ahmed, M. Zhang, and L. Peng, "Improving feature ranking for biomarker discovery in proteomics mass spectrometry data using genetic programming," *Connection Science*, vol. 26, no. 3, pp. 215–243, 2014.

[167] K. Neshatian, M. Zhang, and P. Andreae, "Genetic programming for feature ranking in classification problems," in *Asia-Pacific Conference on Simulated Evolution and Learning*. Springer, 2008, pp. 544–554.

[168] T. Hu, "Can genetic programming perform explainable machine learning for bioinformatics?" *Genetic Programming Theory and Practice XVII*, p. 63, 2020.

[169] I. Icke and A. Rosenberg, "Dimensionality reduction using symbolic regression," in *Proceedings of the 12th annual conference companion on Genetic and evolutionary computation*, 2010, pp. 2085–2086.

[170] M. Virgolin, T. Alderliesten, and P. A. Bosman, "On explaining machine learning models by evolving crucial and compact features," *Swarm and Evolutionary Computation*, vol. 53, p. 100640, 2020.

[171] K. Nag and N. R. Pal, "Feature extraction and selection for parsimonious classifiers with multiobjective genetic programming," *IEEE Transactions on Evolutionary Computation*, vol. 24, no. 3, pp. 454–466, 2019.

[172] B. Peng, Y. Bi, B. Xue, M. Zhang, and S. Wan, "Multi-view feature construction using genetic programming for rolling bearing fault diagnosis [application notes]," *IEEE Computational Intelligence Magazine*, vol. 16, no. 3, pp. 79–94, 2021.

[173] B. Tran, B. Xue, and M. Zhang, "Genetic programming for multiple-feature construction on high-dimensional classification," *Pattern Recognition*, vol. 93, pp. 404–417, 2019.

[174] ——, "Class dependent multiple feature construction using genetic programming for high-dimensional data," in *Australasian Joint Conference on Artificial Intelligence*. Springer, 2017, pp. 182–194.

[175] Q. U. Ain, H. Al-Sahaf, B. Xue, and M. Zhang, "Generating knowledge-guided discriminative features using genetic programming for melanoma detection," *IEEE Transactions on Emerging Topics in Computational Intelligence*, 2020.

[176] ——, "Genetic programming for automatic skin cancer image classification," *Expert Systems with Applications*, vol. 197, p. 116680, 2022.

[177] S. Ahmed, M. Zhang, L. Peng, and B. Xue, "Multiple feature construction for effective biomarker identification and classification using genetic programming," in *Proceedings of the 2014 Annual Conference on Genetic and Evolutionary Computation*, 2014, pp. 249–256.

[178] Q. Fan, Y. Bi, B. Xue, and M. Zhang, "Genetic programming with a new representation and a new mutation operator for image classification," in *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, 2021, p. 249–250.

[179] A. Lensen, "Mining feature relationships in data," in *Genetic Programming*, T. Hu, N. Lourenço, and E. Medvet, Eds. Cham: Springer International Publishing, 2021, pp. 247–262.

[180] Q. Fan, Y. Bi, B. Xue, and M. Zhang, "Genetic programming for image classification: A new program representation with flexible feature reuse," *IEEE Transactions on Evolutionary Computation*, 2022.

[181] T. Haynes, R. L. Wainwright, S. Sen, and D. A. Schoenefeld, "Strongly typed genetic programming in evolving cooperation strategies," in *International Conference on Genetic Algorithms*, vol. 95, 1995, pp. 271–278.

[182] S. Wappler and J. Wegener, "Evolutionary unit testing of object-oriented software using strongly-typed genetic programming," in *Proceedings of the annual conference on Genetic and evolutionary computation*, 2006, pp. 1925–1932.

[183] K. Michell and W. Kristjanpoller, "Generating trading rules on us stock market using strongly typed genetic programming," *Soft Computing*, vol. 24, no. 5, pp. 3257–3274, 2020.

[184] Y. Bi, B. Xue, and M. Zhang, *Genetic Programming for Image Classification*. Springer, 2021.

[185] ——, "An automatic feature extraction approach to image classification using genetic programming," in *International Conference on the Applications of Evolutionary Computation*. Springer, 2018, pp. 421–438.

[186] H. Al-Sahaf, A. Song, K. Neshatian, and M. Zhang, "Extracting image features for classification by two-tier genetic programming," in *2012 IEEE Congress on Evolutionary Computation*. IEEE, 2012, pp. 1–8.

[187] A. Lensen, H. Al-Sahaf, M. Zhang, and B. Xue, "Genetic programming for region detection, feature extraction, feature construction and classification in image data," in *European conference on genetic programming*. Springer, 2016, pp. 51–67.

[188] A. E. Almeida and R. d. S. Torres, "Remote sensing image classification using genetic-programming-based time series similarity functions," *IEEE Geoscience and Remote Sensing Letters*, vol. 14, no. 9, pp. 1499–1503, 2017.

[189] L. Shao, L. Liu, and X. Li, "Feature learning for image classification via multiobjective genetic programming," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 25, no. 7, pp. 1359–1371, 2014.

[190] Y. Bi, B. Xue, and M. Zhang, "An effective feature learning approach using genetic programming with image descriptors for image classification [research frontier]," *IEEE Computational Intelligence Magazine*, vol. 15, no. 2, pp. 65–77, 2020.

[191] Jin Li, Xiaoli Li, and Xin Yao, "Cost-sensitive classification with genetic programming," in *2005 IEEE Congress on Evolutionary Computation*, vol. 3, Sep. 2005, pp. 2114–2121 Vol. 3.

[192] F. Padillo, J. M. Luna, and S. Ventura, "A Grammar-Guided Genetic Programing Algorithm for Associative Classification in Big Data," *Cognitive Computation*, vol. 11, no. 3, pp. 331–346, Jun. 2019.

[193] G. L. Pappa and A. A. Freitas, "Automatically evolving rule induction algorithms," in *European Conference on Machine Learning*. Springer, 2006, pp. 341–352.

[194] ——, "Evolving rule induction algorithms with multi-objective grammar-based genetic programming," *Knowledge and Information Systems*, vol. 19, no. 3, pp. 283–309, Jun. 2009.

[195] C. Garcia-Martinez and S. Ventura, "Multi-view semi-supervised learning using genetic programming interpretable classification rules," in *2017 IEEE Congress on Evolutionary Computation*. IEEE, 2017, pp. 573–579.

[196] C. García-Martínez and S. Ventura, "Multi-view Genetic Programming Learning to Obtain Interpretable Rule-Based Classifiers for Semi-supervised Contexts. Lessons Learnt," *International Journal of Computational Intelligence Systems*, vol. 13, no. 1, pp. 576–590, May 2020.

[197] M. Schmidt and H. Lipson, "Distilling Free-Form Natural Laws from Experimental Data," *Science*, vol. 324, no. 5923, pp. 81–85, Apr. 2009.

[198] M. Keijzer and V. Babovic, "Dimensionally aware genetic programming," in *Proceedings of the Annual Conference on Genetic and Evolutionary Computation*, 1999, pp. 1069–1076.

[199] S. Bandaru and K. Deb, "A dimensionally-aware genetic programming architecture for automated innovization," in *International conference on evolutionary multi-criterion optimization*. Springer, 2013, pp. 513–527.

[200] Y. Mei, S. Nguyen, and M. Zhang, "Constrained dimensionally aware genetic programming for evolving interpretable dispatching rules in dynamic job shop scheduling," in *Proceedings of the Asia-Pacific Conference on Simulated Evolution and Learning (SEAL)*. Springer, 2017, pp. 435–447.

[201] D. Li and J. Zhong, "Dimensionally Aware Multi-Objective Genetic Programming for Automatic Crowd Behavior Modeling," *ACM Transactions on Modeling and Computer Simulation*, vol. 30, no. 3, pp. 19:1–19:24, Jul. 2020.

[202] M. Durasevic, D. Jakobovic, M. S. R. Martins, S. Picek, and M. Wagner, "Fitness landscape analysis of dimensionally-aware genetic programming featuring feynman equations," in *International Conference on Parallel Problem Solving from Nature*. Springer, 2020, pp. 111–124.

[203] P. A. Whigham *et al.*, "Grammatically-based genetic programming," in *Proceedings of the workshop on genetic programming: from theory to real-world applications*, vol. 16, no. 3. Citeseer, 1995, pp. 33–41.

[204] N. Cherrier, J.-P. Poli, M. Defurne, and F. Sabatié, "Consistent Feature Construction with Constrained Genetic Programming for Experimental Physics," in *2019 IEEE Congress on Evolutionary Computation*, Jun. 2019, pp. 1650–1658.

[205] S. Nguyen, M. Zhang, M. Johnston, and K. C. Tan, "A Computational Study of Representations in Genetic Programming to Evolve Dispatching Rules for the Job Shop Scheduling Problem," *IEEE Transactions on Evolutionary Computation*, vol. 17, no. 5, pp. 621–639, Oct. 2013.

[206] T. McConaghy and G. Gielen, "Ibmg: Interpretable behavioral model generator for nonlinear analog circuits via canonical form functions and genetic programming," in *Proceedings of the IEEE International Symposium on Circuits and Systems*. IEEE, 2005, pp. 5170–5173.

[207] ——, "Canonical Form Functions As a Simple Means for Genetic Programming to Evolve Human-interpretable Functions," in *Proceedings of the 8th Annual Conference on Genetic and Evolutionary Computation*. New York, NY, USA: ACM, 2006, pp. 855–862.

[208] T. McConaghy and G. G. Gielen, "Template-free symbolic performance modeling of analog circuits via canonical-form functions and genetic programming," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 28, no. 8, pp. 1162–1175, 2009.

[209] J. R. Koza, *Genetic programming II: automatic discovery of reusable programs*. MIT press, 1994.

[210] S. M. Gustafson and W. H. Hsu, "Layered learning in genetic programming for a cooperative robot soccer problem," in *European Conference on Genetic Programming*. Springer, 2001, pp. 291–301.

[211] W. H. Hsu and S. M. Gustafson, "Genetic programming and multi-agent layered learning by reinforcements," in *Proceedings of the Genetic and Evolutionary Computation Conference*, 2002, pp. 764–771.

[212] D. Jackson and A. P. Gibbons, "Layered learning in boolean gp problems," in *European Conference on Genetic Programming*. Springer, 2007, pp. 148–159.

[213] J.-Y. Lin, H.-R. Ke, B.-C. Chien, and W.-P. Yang, "Designing a classifier by a layered multi-population genetic programming approach," *Pattern Recognition*, vol. 40, no. 8, pp. 2211–2225, 2007.

[214] ——, "Classifier design with feature selection and feature extraction using layered genetic programming," *Expert Systems with Applications*, vol. 34, no. 2, pp. 1384–1393, 2008.

[215] N. T. Hien, N. X. Hoai, and B. McKay, "A study on genetic programming with layered learning and incremental sampling," in *Proceedings of the IEEE Congress of Evolutionary Computation*. IEEE, 2011, pp. 1179–1185.

[216] L. Rodriguez-Coayahuitl, A. Morales-Reyes, and H. J. Escalante, "Structurally layered representation learning: Towards deep learning through genetic programming," in *European Conference on Genetic Programming*. Springer, 2018, pp. 271–288.

[217] M. Keijzer, C. Ryan, and M. Cattolico, "Run transferable libraries—learning functional bias in problem domains," in *Proceedings of the Genetic and Evolutionary Computation Conference*. Springer, 2004, pp. 531–542.

[218] C. Ryan, M. Keijzer, and M. Cattolico, "Favourable biasing of function sets using run transferable libraries," in *Genetic Programming Theory and Practice II*. Springer, 2005, pp. 103–120.

[219] M. Keijzer, C. Ryan, G. Murphy, and M. Cattolico, "Undirected training of run transferable libraries," in *European Conference on Genetic Programming*. Springer, 2005, pp. 361–370.

[220] G. Murphy and C. Ryan, "Seeding methods for run transferable libraries," in *Proceedings of the Genetic and Evolutionary Computation Conference*, 2007, pp. 1755–1755.

[221] K. Imamura, R. B. Heckendorn, T. Soule, and J. A. Foster, "N-version genetic programming via fault masking," in *European Conference on Genetic Programming*. Springer, 2002, pp. 172–181.

[222] K. Imamura, T. Soule, R. B. Heckendorn, and J. A. Foster, "Behavioral diversity and a probabilistically optimal gp ensemble," *Genetic Programming and Evolvable Machines*, vol. 4, no. 3, pp. 235–253, 2003.

[223] S. Wang, Y. Mei, J. Park, and M. Zhang, "Evolving ensembles of routing policies using genetic programming for uncertain capacitated arc routing problem," in *Proceedings of IEEE Symposium Series on Computational Intelligence*. IEEE, dec 2019, pp. 1628–1635.

[224] S. Wang, Y. Mei, and M. Zhang, "Novel ensemble genetic programming hyper-heuristics for uncertain capacitated arc routing problem," in *Proceedings of the Genetic and Evolutionary Computation Conference*. ACM, jul 2019, pp. 1093–1101.

[225] T. Soule and P. Komireddy, "Orthogonal evolution of teams: A class of algorithms for evolving teams with inversely correlated errors," in *Genetic Programming Theory and Practice IV*. Springer, 2007, pp. 79–95.

[226] S. Kelly and M. I. Heywood, "Multi-task learning in atari video games with emergent tangled program graphs," in *Proceedings of the Genetic and Evolutionary Computation Conference*, 2017, pp. 195–202.

[227] ——, "Emergent tangled graph representations for atari game playing agents," in *European Conference on Genetic Programming*. Springer, 2017, pp. 64–79.

[228] R. J. Smith and M. I. Heywood, "Scaling tangled program graphs to visual reinforcement learning in vizdoom," in *European Conference on Genetic Programming*. Springer, 2018, pp. 135–150.

[229] S. Kelly and M. I. Heywood, "Emergent tangled program graphs in multi-task learning," in *IJCAI*, 2018, pp. 5294–5298.

[230] D. G. Wilson, S. Cussat-Blanc, H. Luga, and J. F. Miller, "Evolving simple programs for playing atari games," in *Proceedings of the genetic and evolutionary computation conference*, 2018, pp. 229–236.

[231] S. Kelly, R. J. Smith, and M. I. Heywood, "Emergent policy discovery for visual reinforcement learning through tangled program graphs: A tutorial," *Genetic programming theory and practice XVI*, pp. 37–57, 2019.

[232] R. J. Smith and M. I. Heywood, "A model of external memory for navigation in partially observable visual reinforcement learning tasks," in *European Conference on Genetic Programming*. Springer, 2019, pp. 162–177.

[233] ——, "Evolving a dota 2 hero bot with a probabilistic shared memory model," in *Genetic Programming Theory and Practice XVII*. Springer, 2020, pp. 345–366.

[234] R. J. Smith, R. Amaral, and M. I. Heywood, "Evolving simple solutions to the cifar-10 benchmark using tangled program graphs," in *IEEE Congress on Evolutionary Computation*. IEEE, 2021, pp. 2061–2068.

[235] S. Kelly, R. J. Smith, M. I. Heywood, and W. Banzhaf, "Emergent tangled program graphs in partially observable recursive forecasting and vizdoom navigation tasks," *ACM Transactions on Evolutionary Learning and Optimization*, vol. 1, no. 3, pp. 1–41, 2021.

[236] K. Desnos, N. Sourbier, P.-Y. Raumer, O. Gesny, and M. Pelcat, "Gegelati: Lightweight artificial intelligence through generic and evolvable tangled program graphs," in *Workshop on Design and Architectures for Signal and Image Processing*, 2021, pp. 35–43.

[237] C. Bayer, R. Amaral, R. J. Smith, A. Ianta, and M. I. Heywood, "Finding simple solutions to multi-task visual reinforcement learning problems with tangled program graphs," in *Genetic Programming Theory and Practice XVIII*. Springer, 2022, pp. 1–19.

[238] N. Sourbier, K. Desnos, T. Guyet, F. Majorczyk, O. Gesny, and M. Pelcat, "Secure-gegelati always-on intrusion detection through gegelati lightweight tangled program graphs," *Journal of Signal Processing Systems*, pp. 1–18, 2022.

[239] N. Sourbier, J. Bonnot, F. Majorczyk, O. Gesny, T. Guyet, and M. Pelcat, "Imbalanced classification with tpg genetic programming: impact of problem imbalance and selection mechanisms," in *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, 2022, pp. 608–611.

[240] M. Lemczyk and M. Heywood, "Pareto-coevolutionary genetic programming classifier," in *Proceedings of the Genetic and Evolutionary Computation Conference*, 2006, pp. 945–946.

[241] P. Lichodzijewski and M. I. Heywood, "Managing team-based problem solving with symbiotic bid-based genetic programming," in *Proceedings of the Genetic and Evolutionary Computation Conference*, 2008, pp. 363–370.

[242] A. R. McIntyre, *Novelty Detection + Coevolution = Automatic problem decomposition: A framework for scalable Genetic Programming classifiers*. Dalhousie University, 2008.

[243] P. Lichodzijewski, "A symbiotic bid-based framework for problem decomposition using genetic programming," Ph.D. dissertation, Faculty of Computer Science, Dalhousie University, 2011.

[244] M. Virgolin, A. De Lorenzo, E. Medvet, and F. Randone, "Learning a formula of interpretability to learn interpretable formulas," in *International Conference on Parallel Problem Solving from Nature*. Springer, 2020, pp. 79–93.

[245] M. Virgolin, A. De Lorenzo, F. Randone, E. Medvet, and M. Wahde, "Model learning with personalized interpretability estimation (ml-pie)," in *Proceedings of the Genetic and Evolutionary Computation Conference Companion*. ACM, 2021, p. 1355–1364.

[246] B. P. Evans, B. Xue, and M. Zhang, "What's inside the black-box? a genetic programming method for interpreting complex machine learning models," in *Proceedings of the Genetic and Evolutionary Computation Conference*, 2019, pp. 1012–1020.

[247] U. Johansson, L. Niklasson, and R. König, "Accuracy vs. comprehensibility in data mining models," 2004, pp. 295–300.

[248] U. Johansson, "Obtaining accurate and comprehensible data mining models: An evolutionary approach," Ph.D. dissertation, Institutionen för datavetenskap, 2007.

[249] L. A. Ferreira, F. G. Guimarães, and R. Silva, "Applying genetic programming to improve interpretability in machine learning models," in *2020 IEEE Congress on Evolutionary Computation (CEC)*. IEEE, 2020, pp. 1–8.

[250] R. M. Filho, A. Lacerda, and G. L. Pappa, "Explaining symbolic regression predictions," in *2020 IEEE Congress on Evolutionary Computation*, 2020, pp. 1–8.

[251] S. Wang, Y. Mei, and M. Zhang, "Local ranking explanation for genetic programming evolved routing policies for uncertain capacitated arc routing problems," in *Proceedings of the ACM Genetic and Evolutionary Computation Conference (GECCO)*. ACM, jul 2022, pp. 314–322.

[252] N. Boukhelifa and E. Lutton, "Guest editorial: Special issue on genetic programming, evolutionary computation and visualization," *Genetic Programming and Evolvable Machines*, vol. 19, no. 3, pp. 313–315, 2018.

[253] J. J. Valdés, R. Orchard, and A. J. Barton, "Exploring medical data using visual spaces with genetic programming and implicit functional mappings," in *Genetic and Evolutionary Computation Conference*, D. Thierens, Ed. ACM, 2007, pp. 2953–2960.

[254] A. Lensen, B. Xue, and M. Zhang, "Can genetic programming do manifold learning too?" in *European Conference on Genetic Programming*. Springer, 2019, pp. 114–130.

[255] A. Lensen, M. Zhang, and B. Xue, "Multi-objective genetic programming for manifold learning: Balancing quality and dimensionality," *Genetic Programming and Evolvable Machines*, vol. 21, pp. 399–431, 2020.

[256] A. Lensen, B. Xue, and M. Zhang, "Genetic programming for manifold learning: Preserving local topology," *IEEE Transactions on Evolutionary Computation*, pp. 1–15, 2021.

[257] ——, "Genetic Programming for Evolving a Front of Interpretable Models for Data Visualization," *IEEE Transactions on Cybernetics*, pp. 1–15, 2020.

[258] L. van der Maaten and G. Hinton, "Visualizing Data using t-SNE," *Journal of Machine Learning Research*, vol. 9, no. 86, pp. 2579–2605, 2008.

[259] F. Schofield and A. Lensen, "Using genetic programming to find functional mappings for UMAP embeddings," in *IEEE Congress on Evolutionary Computation*. IEEE, 2021, pp. 704–711.

[260] I. Icke, "Multi-objective genetic programming for data visualization and classification," Ph.D. dissertation, Computer Science, City University of New York, USA, 2011.

[261] I. Icke and A. Rosenberg, "Multi-objective Genetic Programming for Visual Analytics," in *Proceedings of the European Conference on Genetic Programming*, 2011, pp. 322–334.

[262] ——, "Automated measures for interpretable dimensionality reduction for visual classification: A user study," in *6th IEEE Conference on Visual Analytics Science and Technology*. IEEE, 2011, pp. 281–282.

[263] A. Cano, S. Ventura, and K. J. Cios, "Multi-objective genetic programming for feature extraction and data visualization," *Soft Comput.*, vol. 21, no. 8, pp. 2069–2089, 2017.

[264] M. Virgolin, T. Alderliesten, and P. A. N. Bosman, "On explaining machine learning models by evolving crucial and compact features," *Swarm Evol. Comput.*, vol. 53, p. 100640, Mar. 2020.

[265] H. Pohlheim, "Visualization of evolutionary algorithms - set of standard techniques and multidimensional visualization," in *Proceedings of the Genetic and Evolutionary Computation Conference*, vol. 1, 1999, pp. 533–540.

[266] E. Burke, S. Gustafson, and G. Kendall, "Diversity in Genetic Programming: An Analysis of Measures and Correlation With Fitness," *IEEE Transactions on Evolutionary Computation*, vol. 8, no. 1, pp. 47–62, Feb. 2004.

[267] D. Jackson, "Promoting phenotypic diversity in genetic programming," in *Proceedings of the 11th International Conference on Parallel Problem Solving from Nature (PPSN) Part II*, ser. Lecture Notes in Computer Science, vol. 6239. Springer, 2010, pp. 472–481.

[268] E. Medvet, M. Virgolin, M. Castelli, P. A. N. Bosman, I. Gonçalves, and T. Tusar, "Unveiling evolutionary algorithm representation with DU maps," *Genetic Programming and Evolvable Machines*, vol. 19, no. 3, pp. 351–389, 2018.

[269] J. M. Daida, D. J. Ward, A. M. Hilss, S. L. Long, M. R. Hodges, and J. T. Kriesel, "Visualizing the loss of diversity in genetic programming," in *Proceedings of the 2004 IEEE Congress on Evolutionary Computation*. Portland, Oregon: IEEE Press, 20-23 Jun. 2004, pp. 1225–1232.

[270] N. F. McPhee, M. D. Finzel, M. M. Casale, T. Helmuth, and L. Spector, "A detailed analysis of a pushgp run," in *Genetic Programming Theory and Practice XIV*. Springer, 2018, pp. 65–83.

[271] J. H. Moore, D. P. Hill, J. M. Fisher, N. Lavender, and L. C. Kidd, "Human-computer interaction in a computational evolution system for the genetic analysis of cancer," in *Genetic Programming Theory and Practice IX*. Springer, 2011, pp. 153–171.

[272] S. Nguyen, M. Zhang, D. Alahakoon, and K. C. Tan, "Visualizing the evolution of computer programs for genetic programming," *IEEE Computational Intelligence Magazine*, vol. 13, no. 4, pp. 77–94, 2018.

[273] ——, "People-centric evolutionary system for dynamic production scheduling," *IEEE Transactions on Cybernetics*, vol. 51, no. 3, pp. 1403–1416, 2021.

[274] Y. Yang, B. Xue, L. Jesson, and M. Zhang, "Genetic programming for symbolic regression: A study on fish weight prediction," in *2021 IEEE Congress on Evolutionary Computation*. IEEE, 2021, pp. 588–595.

[275] D.-K. Kim, H. Cao, K.-S. Jeong, F. Recknagel, and G.-J. Joo, "Predictive function and rules for population dynamics of microcystis aeruginosa in the regulated nakdong river (south korea), discovered by evolutionary algorithms," *Ecological Modelling*, vol. 203, no. 1-2, pp. 147–156, 2007.

[276] A. Lensen, H. Al-Sahaf, M. Zhang, and B. Verma, "Genetic Programming for algae detection in river images," in *Proceedings of the IEEE Congress on Evolutionary Computation*, May 2015, pp. 2468–2475.

[277] R. Senatore, A. Della Cioppa, and A. Marcelli, "Automatic diagnosis of parkinson disease through handwriting analysis: A cartesian genetic programming approach," in *IEEE International Symposium on Computer-Based Medical Systems*, 2019, pp. 312–317.

[278] F. Cavaliere, A. D. Cioppa, A. Marcelli, A. Parziale, and R. Senatore, "Parkinson's disease diagnosis: Towards grammar-based explainable artificial intelligence," in *2020 IEEE Symposium on Computers and Communications*, 2020, pp. 1–6.

[279] B. N. Tran, "Evolutionary computation for feature manipulation in classification on high-dimensional data," Ph.D. dissertation, 2018.

[280] W. Langdon and S. Barrett, "Genetic programming in data mining for drug discovery," in *Evolutionary computation in data mining*. Springer, 2005, pp. 211–235.

[281] P. Barmpalexis, K. Kachrimanis, A. Tsakonas, and E. Georgarakis, "Symbolic regression via genetic programming in the optimization of a controlled release pharmaceutical formulation," *Chemometrics and Intelligent Laboratory Systems*, vol. 107, no. 1, pp. 75–82, 2011.

[282] C. Wen, M. Lu, Y. Bi, S. Zhang, B. Xue, M. Zhang, Q. Zhou, and W. Wu, "An object-based genetic programming approach for cropland field extraction," *Remote Sensing*, vol. 14, no. 5, p. 1275, 2022.

[283] K. Stanislawska, K. Krawiec, and Z. W. Kundzewicz, "Modeling global temperature changes with genetic programming," *Computers & Mathematics with Applications*, vol. 64, no. 12, pp. 3717–3728, 2012.

[284] B. Xu, G. Wen, Z. Zhang, and F. Chen, "Genetic programming-based classification of ferrograph wear particles," in *International Conference on Ubiquitous Robots and Ambient Intelligence*, 2016, pp. 842–847.

[285] M. Quade, T. Isele, and M. Abel, "Machine learning control - explainable and analyzable methods," *Physica D: Nonlinear Phenomena*, vol. 412, p. 132582, 2020.

[286] D. Hein, S. Limmer, and T. A. Runkler, "Interpretable control by reinforcement learning," *IFAC-PapersOnLine*, vol. 53, no. 2, pp. 8082–8089, 2020.

[287] L. Berardi, O. Giustolisi, Z. Kapelan, and D. Savic, "Development of pipe deterioration models for water distribution systems using epr," *Journal of Hydroinformatics*, vol. 10, no. 2, pp. 113–126, 2008.

[288] M. Quade, M. Abel, K. Shafi, R. K. Niven, and B. R. Noack, "Prediction of dynamical systems by symbolic regression," *Physical Review E*, vol. 94, no. 1, p. 012214, 2016.

[289] J. Gout, M. Quade, K. Shafi, R. K. Niven, and M. Abel, "Synchronization control of oscillator networks using symbolic regression," *Nonlinear Dynamics*, vol. 91, no. 2, pp. 1001–1021, 2018.

[290] W. Xing, R. Guo, E. Petakovic, and S. Goggins, "Participation-based student final performance prediction model through interpretable genetic programming: Integrating learning analytics, educational data mining and theory," *Computers in Human Behavior*, vol. 47, pp. 168–181, 2015.

[291] S. Mousavi, A. Esfahanipour, and M. H. Fazel Zarandi, "MGP-INTACTSKY: Multitree Genetic Programming-based learning of INTerpretable and ACcurate TSK sYstems for dynamic portfolio trading," *Applied Soft Computing*, vol. 34, pp. 449–462, Sep. 2015.

[292] M. Cranmer, A. Sanchez Gonzalez, P. Battaglia, R. Xu, K. Cranmer, D. Spergel, and S. Ho, "Discovering symbolic models from deep learning with inductive biases," *Advances in Neural Information Processing Systems*, vol. 33, pp. 17 429–17 442, 2020.

[293] K. Debattista, "Application-specific tone mapping via genetic programming," *Computer Graphics Forum*, vol. 37, no. 1, pp. 439–450, 1 Nov. 2018.

[294] S. T. Mueller, R. R. Hoffman, W. Clancey, A. Emrey, and G. Klein, "Explanation in Human-AI Systems: A Literature Meta-Review, Synopsis of Key Ideas and Publications, and Bibliography for Explainable AI," *arXiv:1902.01876 [cs]*, Feb. 2019.

[295] G. S. I. Aldeia and F. O. de França, "Interpretability in symbolic regression: a benchmark of explanatory methods using the feynman data set," *Genetic Programming and Evolvable Machines*, vol. 23, p. 309–349, 2022.