Applied Soft Computing 143 (2023) 110427

Contents lists available at ScienceDirect

Applied Soft Computing

journal homepage: www.elsevier.com/locate/asoc

Review article

A comprehensive review of automatic programming methods

Sibel Arslan [a,*], Celal Ozturk [b]

[a] Department of Software Engineering, Sivas Cumhuriyet University, Sivas, 58140, Turkey
[b] Department of Software Engineering, Erciyes University, Kayseri, 38280, Turkey

ARTICLE INFO

Article history:
Received 22 March 2022
Received in revised form 20 February 2023
Accepted 10 May 2023
Available online 18 May 2023

Keywords:
Artificial intelligence
Evolutionary computation
Automatic programming
Genetic programming

ABSTRACT

Automatic programming (AP) is one of the most attractive branches of artificial intelligence because it provides effective solutions to problems with limited knowledge in many different application areas. AP methods can be used to determine the effects of a system's inputs on its outputs. Although there is increasing interest in solving many problems using these methods for a variety of applications, there is a lack of reviews that address the methods. Therefore, the goal of this paper is to provide a comprehensive literature review of AP methods. At the same time, we mention the main characteristics of the methods by grouping them according to how they represent solutions. We also try to give an outlook on the future of the field by highlighting possible bottlenecks and perspectives for the benefit of the researchers involved.

Contents

* Corresponding author.
E-mail addresses: sibelarslan@cumhuriyet.edu.tr (S. Arslan), celal@erciyes.edu.tr (C. Ozturk).

https://doi.org/10.1016/j.asoc.2023.110427
1568-4946/

1. Introduction

Automatic Programming (AP) is a subfield of artificial intelligence (AI) and soft computing that attempts to accurately describe systems. It has been expressed in various definitions over time. One of these describes AP as the ability to automatically generate machine code and similarly refers to the creation of compilers [1]. Another definition was defined by Arthur Samuel as “telling the machine what to do, not how to do it”. In this definition, “telling the machine what to do” is now interpreted as high-level language [2]. In future studies, AP is expected to evolve over time to higher level programming languages for telling the machine what to do in human–computer interaction. The theory and methodology of genetic programming (GP) has increased its potential for AP as well as for solving various machine learning problems. Many studies have described GP as an evolutionary computation-based AP method [3,4]. In addition, inspired by GP (representation of solutions, operators, etc.), various methods have been proposed and applied in different engineering fields. Numerous studies have been published on AP methods that show relatively better performance than conventional algorithms. As interest in this area grows, researchers are working to develop better models because many of the problems have limited information, coupled with the fact that the methods rely on random processes and lead to successful conclusions more quickly. For this reason, there are always new publications and applications in the literature related to solving many complex problems in engineering, economics, medicine, construction and other fields using AP methods.

The most accurate and safest way to search for solutions to many problems in different application areas is to try all possibilities. Unfortunately, it takes an unreasonable amount of time to try all acceptable solutions. In recent years, AP has emerged strongly as a solution with methods that can generate the best models by trying many fewer possibilities. As a result, more and more different methods and AP versions have been presented at prestigious conferences and journals in recent years. Despite numerous review articles on traditional algorithms inspired by AP methods and their applications, to the best of our knowledge, there is no survey of the literature on AP and its methods from the past to the present. In our research, we only came across one review article on swarm-based AP methods published in 2014 [5]. Therefore, we decided to prepare a comprehensive review focusing on AP methods and versions for different problems. Our motivation is not only to explain the important and special characteristics of each of AP methods reviewed so that researchers can make satisfactory choices when solving different problems, but also to take a look at these methods from an academic perspective. We also wanted to provide an outlook on where the AP field is going and what future directions are possible.

Our main motivation is to answer the following research questions:

- Which AP methods are used for which problem solutions?
- With which conventional algorithms can the performance of AP methods be compared?
- In what direction will the field of AP develop?

We focus on reviewing the applications of AP methods to a variety of problems from their appearance to the present. The encoding (representation of solutions), inspiration, and operators for each method are presented in Table 1. The operators of the methods also differ depending on the encoding. For example, in GP, since the solutions are represented by parse trees, the crossover operator can be used, which allows swapping of subtrees; in GE, since the solutions are represented by linear arrays, the genotype–phenotype mapping operator can be used. As shown in Table 1, the characteristics of the methods are grouped by encoding similarity. It should be noted that the encoding schemes used in the standard versions of the methods are included in the table. In the rest of the study, the methods are categorized according to these groupings.

For our research, we primarily included journals, conferences, and notable studies that came up in web searches. In reviewing the studies since the day of introduction, we compiled statistics on the field, especially considering the AP applications in the last 5 years.

Based on our findings, this study is the first comprehensive review to address AP. There are many reviews of the GP method in the literature, but we did not come across studies of the AP methods. Therefore, we aimed to fill this gap by analyzing AP and its applications accordingly. We hope that, as a result of this study, researchers will have a better idea of the main features and applications of AP methods in their studies.

The remainder of this paper is organized as follows. Section 2 discusses the current AP methods, considering the grouping in Table 1. Section 3 mentions recent studies dealing with AP methods.

Table 1
AP methods grouped by encoding.

Method	Encoding	Inspiration	Operators	Publication
GP	Parse tree	GA	Crossover and mutation	[6]
AntP		ACO	Evaporation process, Reinforcement of the components	[7]
AISP		AIS	Clonal selection principle	[8]
ABCP		ABC	Information sharing mechanism	[9]
BBP		BBO	Migration, mutation	[10]
DoME		Can be adapted any algorithm	Simple expression tree, limit complexity	[11]
GE	Linear array	GA	BNF and genotype–phenotype mapping	[12]
AnP		Can be adapted any algorithm	GFS, DSH, and SPs	[13]
PSOP		PSP	Mapping schema	[14]
CSP		AIS	Clone selection principle, Removal of agents based on their affinity	[15]
HP		Horse heard	Social hierarchy, Positioning by weighting The quality of the surrounding areas	[16]
FP	Gene expression tree	GFA	GEP schema, crossover, Hamming distance	[17]
AFSP		AFSO	Penalty-based fitness function	[18]
EP	Follows for the problem	EA	Mutation	[19]
IP	Stack-based framework	AIS	Replacement, cloning, Hypermutation	[20]
PME	Parse matrix with integer entries	Can be adapted any algorithm	Mapping rules	[21]
BIBP	Seperable sample set	D&C method	Block and factor detection	[22]

Section 4 presents the overall statistical results of our findings, including histograms and heatmap of the distributions of AP methods by problem topics. Section 5 discusses the bottlenecks that need improvement and their future directions. Finally, we conclude this paper in Section 6 by summarizing our findings.

2. AP methods

This section presents information on the general characteristics of AP methods grouped in Table 1. With this grouping, the methods are examined in 4 basic subsections, from the most commonly used solution representation to the least commonly used. In the last subsection, 4 different representations are listed as "Other representations of solutions". However, since GP is the pioneer of these methods, this algorithm is explained in more detail.

2.1. The representation of solutions as parse trees

2.1.1. Genetic Programming (GP)

Cramer applied a tree approach similar to the first modern application to Markov decision processes, and Koza inspired this approach and proposed GP for complex optimization and search problems [6,23]. The standard GP aims to produce the best results by applying operators such as selection, crossover, and mutation to the solutions in the population generated in the parse trees.

GP uses hierarchical trees of variable size, similar to the tree structures in the Lisp programming language. The smallest unit of trees is called a node. Nodes are selected from the terminal set (variables and constants such as x, y) and the function set (arithmetic operators, logical functions, mathematical functions) defined specifically for problems. Fig. 1 illustrates a commonly used solution example in the standard GP (parse tree structure). The mathematical equation of the solution GP in the figure is expressed in Eq. (1). In these representations, the symbols x_1 and x_2 are used to represent the independent variable and the symbol $f(x_1, x_2)$ is used to represent the dependent variable.

$$f(x_1, x_2) = \frac{\sin(x_1)}{x_2} + x_1 - 0.8 \tag{1}$$

The flowchart GP is shown in Fig. 2. The first step of the flowchart is to generate the initial population. The individuals of the population can be generated using *full method*, *grow method* or *ramped half and half method* [24]. In the *full method*, nodes are randomly selected from the function set until they reach the maximum tree depth. The nodes at the maximum tree depth are selected from the terminal set. The *grow method* allows the creation of trees of different sizes and shapes. The nodes are randomly selected from the function and terminal set until they reach the maximum tree depth. When the maximum tree depth is reached, only the terminals are selected, as in the *full method*. To generate trees of different size and shape and a diverse population, Koza [6] has proposed a method called *ramped half and half* as a combination of these two methods. In this method, half of the initial population is *full method* generated and the other half is generated using the *grow method*.

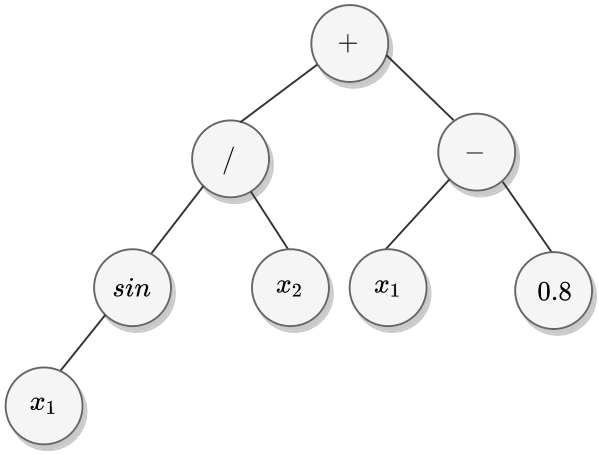

Fig. 1. GP parse tree solution example.

The work of natural selection, crossover, and mutation in Genetic Algorithm (GA) is adapted to the GP to improve existing solutions and search for new ones. The natural selection operator evaluates the fitness of each individual in the population for the problem. In all cases, the suitability of the individual is determined by performing a predetermined fitness function and analyzing the entire tree [15]. The operator selects the chromosomes to be transferred to the next generation by using a selection strategy after evaluating the individuals [25]. The operator aims to obtain better individuals for future generations by

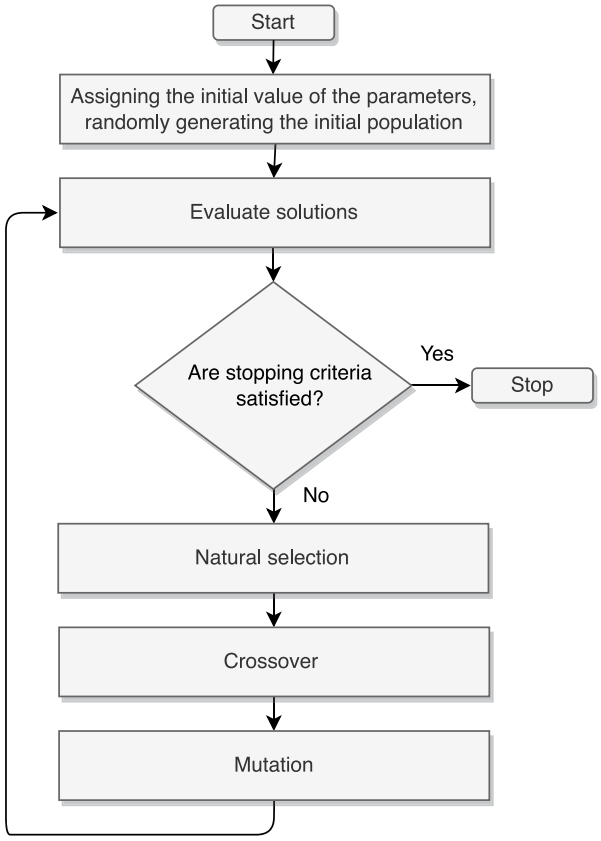

Fig. 2. GP flowchart.

selecting individuals with high fitness values. There are several selection operators in the literature with different approaches to evaluating selection probability [26]. The preferred selection methods, tournament selection and roulette selection operators, are shown in Fig. 3.

The crossover operator is a genetic processing operator that provides new solutions by exchanging information between individuals and increasing diversity in the population. The individuals selected by the tournament and roulette wheel are parental candidates. Fig. 4 shows how the crossover operator works. The mutation operator provides diversity to the population after the selection and crossover operators are used. Mutation attempts to prevent the similarity of individuals that results in individuals not searching the entire solution space by attaching to the local maximum/minimum in the search space. It also allows finding new, unseen and unexplored solutions [27]. The two most commonly used mutation operators are subtree and single node.

In sub-tree mutation, an individual and a random node are selected and the new randomly generated sub-tree is replaced by that node. The randomly selected node can be from the function or terminal set. In single node mutation, a randomly selected node from the terminal set is replaced by a node from the terminal set. If the node is from the function set, it is replaced by the node from the function set. These mutation operators are shown in Fig. 5. The best solutions of the generation are transferred to the current generation in elitism. The program terminates when the given termination criteria (such as the specific fitness value of the solutions, the number of iterations) are met.

2.1.2. Ant Programming (AntP)

Roux and Fonlupt proposed AntP as the first swarm-based AP method [7]. It follows Ant Colony Optimization (ACO) and GP, in optimizing the shortest distance between ant colony nests and food sources.

As with GP, AntP first generates trees using the *ramped half and half method*. The method uses the ACO to search for trees. Each node in the trees has a table that stores the pheromone ratio of the different possible elements. A representative tree example in AP is shown in Fig. 6. Initially, the pheromone table is assigned a value of 0.5 to ensure that the probability of selecting either the function or the terminal is equal. The higher this value, the greater the probability that the function will be selected. The pheromone table is updated for each element as the pheromone ratio decreases with the evaporation process or as the elements become stronger depending on the suitability of the trees. The flowchart of the standard AntP is shown in Fig. 7.

The most commonly used versions of AntP in the literature are listed below:

- *Parallel AntP [28]*: It introduces genetic operators to improve trees to prevent premature convergence.
- *AntP and its versions [29,30]*: They are based on the classical ant colony algorithm for solving symbolic regression (SR) problems. These algorithms generate the tree structure in the form of a graph.
- *Grid Ant Colony Programming (GACP) [31]*: It generates grid-like solutions instead of graphs. In this algorithm, the nodes visited by the ants can be stored. Thus, the amount of pheromone released by a particular ant on different visits to the same node can be adjusted.
- *Grammar-based AntP versions*: Ant Tree Adjunct Grammar (AntTAG) [32], Generalized Ant Programming (GAP) [33], grammar-based AntP (GBAP) [34]. These algorithms generate programs by combining different algorithms with ACO.
- *Cartesian Ant Programming (CAP) [35]*: It represents programs as a graph addressed in the Cartesian coordinate system.
- *Dynamic Ant Programming (DAP) [36]*: It uses dynamically changing pheromone table.

2.1.3. Artificial Immune System Programming (AISP)

From the past to the present, there are many AP methods inspired by the immune system. Artificial Immune System Programming (AISP) proposed by Johnson for the SR problem [8]. The algorithm is inspired by systems in which organisms can learn to identify and distinguish antigens and harmful cells. The other method is the immune version of GP with a dynamic fitness function implemented by an immune network [37]. Inductive genetic programming (IGP) has achieved more successful results on many machine learning problems than the standard method GP.

2.1.4. Artificial Bee Colony Programming (ABCP)

ABCP is an AP method developed by Karaboga et al. and inspired by the foraging behavior of honeybees [9]. The position of a food source in the algorithm, a possible solution to the problem; the amount of nectar of the food source relates to the quality of the solution. Similar to GP, each solution is represented by parse trees. Moreover, the information sharing mechanism for improving the solutions was developed following the crossover operator (Fig. 4) in GP. The flowchart of the ABCP algorithm is shown in Fig. 8.

In the algorithm, there are bees with three different tasks in the colony: employed bees, onlooker bees, and scout bees. In the initial phase, all bees in the colony are in scout status. Random solutions are generated for each scout bee. Employed bees try to find a better food source in their memory by using another food source through an information sharing mechanism. Employed

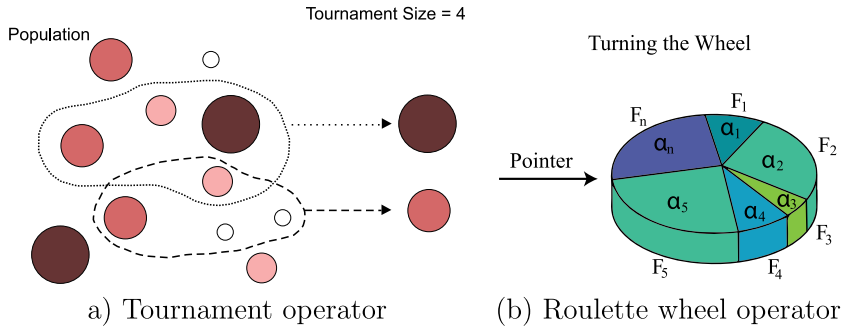

Fig. 3. Natural selection operators.

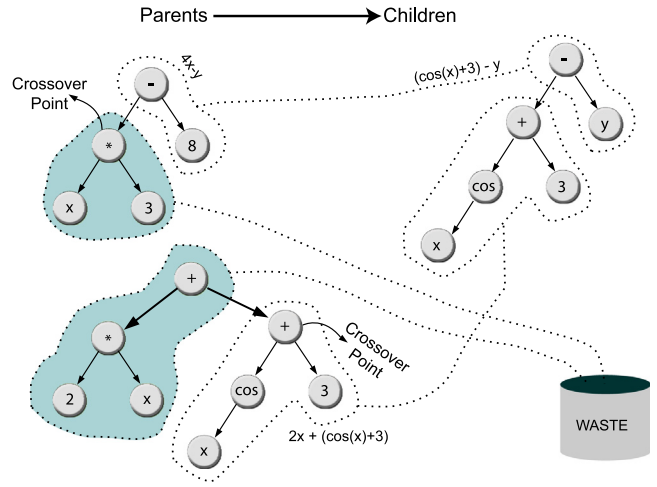

Fig. 4. Crossover operator.

bees share information about the quality of their food sources with onlooker bees. The onlooker bees choose a different food source depending on the quality of employed bees' food source and try to improve the chosen source as in the employed bees phase. A parameter called *limit* controls the exhaustion of the food sources. The number of improvement attempts is recorded for each source, and in each iteration it is checked if the number of attempts exceeds the value of the *limit* parameter. If it does, the food source is considered exhausted and is abandoned. The assigned bee of this source becomes a scout bee and continues working with a new, randomly generated source.

2.1.5. Biogeography-Based Programming (BBP)

Biogeography is a subdiscipline of physical geography that studies the distribution of plant and animal species and the causes of that distribution. BBP is a method inspired by the science of biogeography. In the algorithm, a habitat corresponds to a solution, and a habitat with a Habitat Suitability Index (HSI) is called a good habitat that tends to host a large number of species. BBP aims to randomly generate habitats in the initial population and improve them with migration and mutation operators. The migration operator is the same as the crossover in GP (Fig. 4). The mutation operator is the same as the subtree mutation in GP (Fig. 5). To generate new habitats, it tries to select habitats with high HSI values. The details of the algorithm can be found in [10].

2.1.6. Deterministic Technique for Equation Development (DoME)

To best our knowledge, this is the last AP method found in the literature. DoME uses tree structure solutions like GP, but can compute the error directly from the output of a node [11]. Thus, the target model is created by computing the best constant for the node and the error of the node output. Another difference with GP is that the trees proposed by the method can generate a suitable model without using a protected function.

2.2. The representation of solutions as linear arrays

2.2.1. Grammatical Evolution (GE)

GE is a method that can generate grammatical programs independently of the target programming language. The main point that distinguishes GE from other methods is the mapping of genotype to phenotype. Genotypes are randomly generated sequences of integers, each of which is called a codon. The phenotype is a running program generated by this mapping process. A Backus Naur Form (BNF) grammar is used for mapping. It contains four basic structures: non-terminals, terminals, start generation, and rules for each production of non-terminals. Genotype refinement is performed using genetic operators. As with other methods, the fitness value of the phenotype is evaluated depending on the objective function. During the algorithm process, the codons of the individuals are iteratively changed depending on the rules. Details of the algorithm can be found in [12].

2.2.2. Analytic Programming (AnP)

AnP was first proposed by Zelinka [13] for SR problems, but has since been successfully applied to many engineering problems [38–43]. AnP is a very powerful method, as it can be used with any arbitrary-based algorithm on individual representation.

AnP can create nonlinear mathematical expressions with simple mathematical elements. These elements can be mathematical operators, functions, variables, or constants and are contained in the General Functional Set (GFS). In AnP, a set of indices is first assigned to the inputs. Then the mathematical expression (output) is obtained from the elements referenced by the indices. This process of AnP is shown in Fig. 9.

Other core functionality of AP besides GFS are Discrete Set Handling (DSH) and Security Procedures (SPs). DSH is used to map the output programs of individuals. SPs are applied to the cost function to avoid critical situations such as an infinite loop in the mapping process. Detailed information about AnP can be found in [38,44].

2.2.3. Particle Swarm Optimization Programming (PSOP)

O'Neill and Brabazon named the first AP method based on Particle Swarm Programming (PSO), Grammatical Swarm (GS) [14]. GS can map candidate solutions to programs by using a grammar, as in GE. PSOP, another PSO-based method, treats solutions as expression trees and attempts to improve solutions using the crossover mutation-based discrete particle swarm algorithm [45].

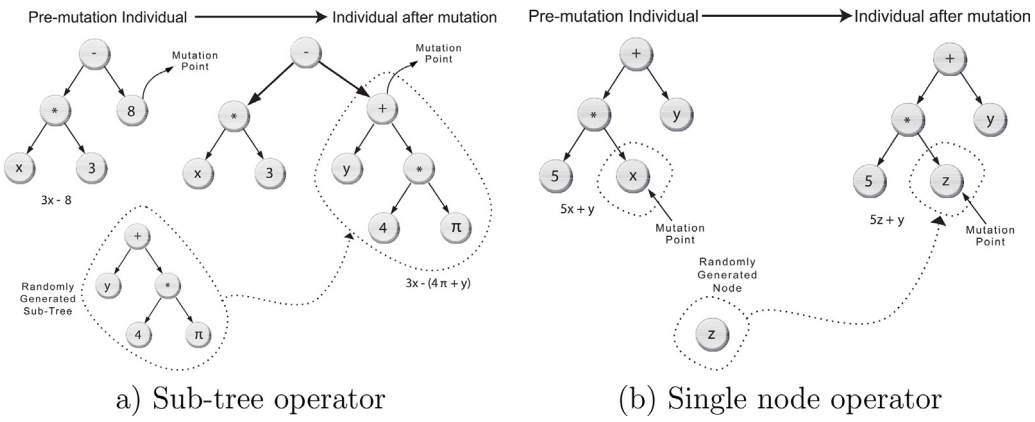

Fig. 5. Mutation operators.

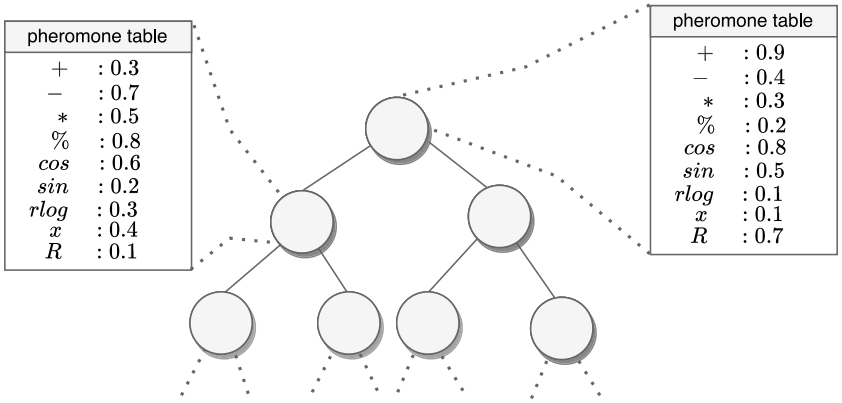

Fig. 6. Representation tree with a pheromone table associated with each node (only two nodes are shown).

2.2.4. Clone Selection Programming (CSP)

Clone Selection Programming (CSP), which develops the principle of clone selection as a search strategy and encoding schema [15]. In their experiments, they showed that the performance of the method is superior to IP and Gen Expression Programming (GEP).

It is randomly generated as the initial population in the initial phase of the CSP algorithm. In the algorithm, each antibody represents a candidate solution. Antibodies are fixed length strings of one or more genes. It consists of two basic components: Antibody coding and expression trees. The symbols of the antibodies have a one-to-one relationship with the programs they represent. Translation refers to the process of decoding information from the antibody encoding expression tree. The arrangement of functions and terminals in these programs is determined by the rules. The translation of an algebraic expression from a string array to an expression tree is shown in Fig. 10. Q is the square-root function, E is the exponential function, and S is the sinusoidal function. Improving antibodies and ensuring diversity are achieved by cloning and hypermutation operators. The quality of antibodies is based on their affinity value. Therefore, antibodies with high affinity values are selected for these operators to produce better antibodies. The algorithm continues until the stopping criteria are met.

2.2.5. Herd Programming (HP)

Swarm intelligence-based algorithms aim to find the global max, inspired by the collective behavior of individuals in a herd. However, these algorithms were mainly inspired by the natural foraging and hunting behavior of fish, birds, and insect swarms. Moreover, all individuals in the swarm exhibit stochastic behavior due to their neighborhood perception [46]. Individuals can interact locally with different behaviors, such as dancing (bees dance to share nectar information about the food source), chemicals (pheromones in ants), and transmission abilities (movement to benefit the population).

HP is also called Grammatical Herding (GH), the first AP method based on the movements and social hierarchy of horse herds [5,16]. The algorithm uses an encoding scheme like GE and tries to match the proposed solutions with surrounding domains known to produce high quality solutions.

2.3. The representation of solutions as gen expression trees

2.3.1. Firefly Programming (FP)

In the literature, there are two AP methods inspired by Firefly Algorithm (FA). The first of these is the geometric firefly algorithm (GFA), which expresses solutions as gen expression programming (GEP) schemes [17]. The algorithm uses the crossover operator with multiparent recombination. It is also measured by the distance between candidates and the Hamming distance. The other method proposed by Aliwi et al. is FP for solving SR problems [47]. In addition to FA, the authors developed the solutions using the sharing operation mechanism, which they adopted from the crossover operator in GP.

2.3.2. Artificial Fish Swarm Programming (AFSP)

AFSP was developed to solve the SR problem, inspired by the foraging, hunting and breeding behavior of fish [18]. Each artificial fish is encoded as a GEP scheme. In addition, the authors proposed a new penalty-based fitness function.

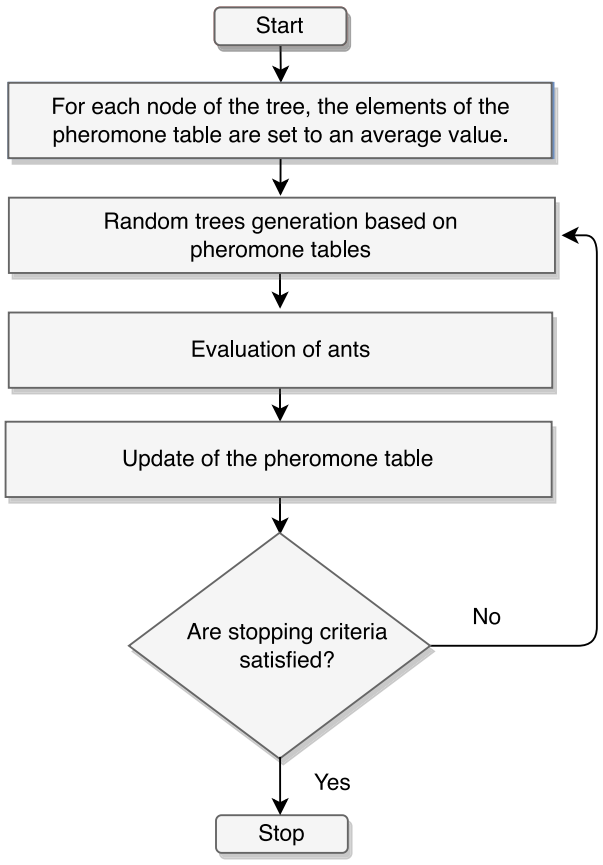

Fig. 7. AntP flowchart.

2.4. Other representations of solutions

2.4.1. Evolutionary Programming (EP)

Evolution is an optimization technique that can be used to solve various engineering problems and is inspired by many scientific methods. Fogel summarizes Evolutionary Computation (EC) into three main areas: (1) GA, (2) Evolutionary Strategies (ES), and (3) EP [48]. EP aims only to improve solutions in the population using the mutation operator [19]. Therefore, it can be called a version of the generation and testing method. It can be used with many different algorithms, such as Simulated Annealing (SA), Tabu Search (TS), Differential Evolution (DE), and PSO. The main drawback of the standard EP is its slow convergence to the optimum [49].

The algorithm of EP proceeds in the following order:

1. The initial population is randomly generated.
2. The fitness value of each individual is calculated using the objective function.
3. The candidate individual is generated using the mutation operator.
4. The fitness value of each candidate is calculated.
5. Greedy selection is used to decide which solutions are passed to the next generation.
6. Stop if termination criterion(s) are met, otherwise proceed to Step 3.

2.4.2. Immune Programming (IP)

Immune Programming (IP), which aims to improve programs by using cloning and hypermutation together [20]. In the algorithm, an antibody is used to identify candidate solutions. Similar to gene libraries in immune systems, each antibody has an array that is a set of stack-based assembly instructions.

2.4.3. Parse Matrix Evolution (PME)

PME can be implemented in any programming language, as in AnP [21]. Since the algorithm expresses the solutions as piecewise matrices with integer inputs, it does not require any additional functions related to the fragmentation process. Mapping tables are used to transform solutions into regression models.

2.4.4. Block Building Programming (BlBP)

BlBP [22] and its improved version Multilevel Block Building (MBB) [50] decompose target models into minimal factors containing one or two variables. These decomposed blocks are to be combined according to the target model. Variables are divided into two classes: repeated and non-repeated.

3. Literature review

For this review, preliminary research was first conducted using the Google Scholar search engine to determine the research questions. Then, these questions were used to identify keywords and search terms/strings related to AP field. Elsevier, Springer, IEEExplore, Web of Science, Scopus, and Google Scholar scientific databases, which contain a variety of the most crucial and highest-impact journals and conference proceedings, were selected for an advanced search using the questions. Title, abstract, and keywords were used as a search scheme to perform a systematic search of each scientific database. Relevant journal papers, conference proceedings, book chapters, and technical reports published in databases were collected. In addition, publications proposing AP methods were also included in the review to briefly introduce the methods.

The keywords and search terms/strings used to search databases for AP are listed below:

- “analytic programming”
- “ant programming” or “ant colony programming”
- “applications” or “utilization or implementation or practices”
- “artificial bee colony programming”
- “automatic programming”
- “data mining” or “data analysis”
- “evolutionary computation” or “evolutionary computing”
- “evolutionary programming”
- “feature selection” or “feature extraction”
- “genetic programming” or “genetic algorithm”
- “grammar evolution” or “grammatical evolution”
- “optimization methods” or “optimization techniques” or “optimization”
- “swarm intelligence” or “swarm programming” or “swarm optimization” or “herd programming”
- “symbolic regression” or “regression”

After completing the database search, we assessed the collected studies and decided which of them should be included or excluded from the systematic literature of the review. In selecting studies related to AP field and answering the research questions, we excluded review articles, book chapters, and other publications that were unrelated to the field.

Algorithms for analyzing application trends and comparing methods as a function of application are important to researchers. In evaluating all reviewed publications, we examined the problems to which AP methods were applied under 7 main problem topics. These are: symbolic regression, classification, prediction, feature selection, image processing, job scheduling, and complex problems not included. We have grouped the work of the

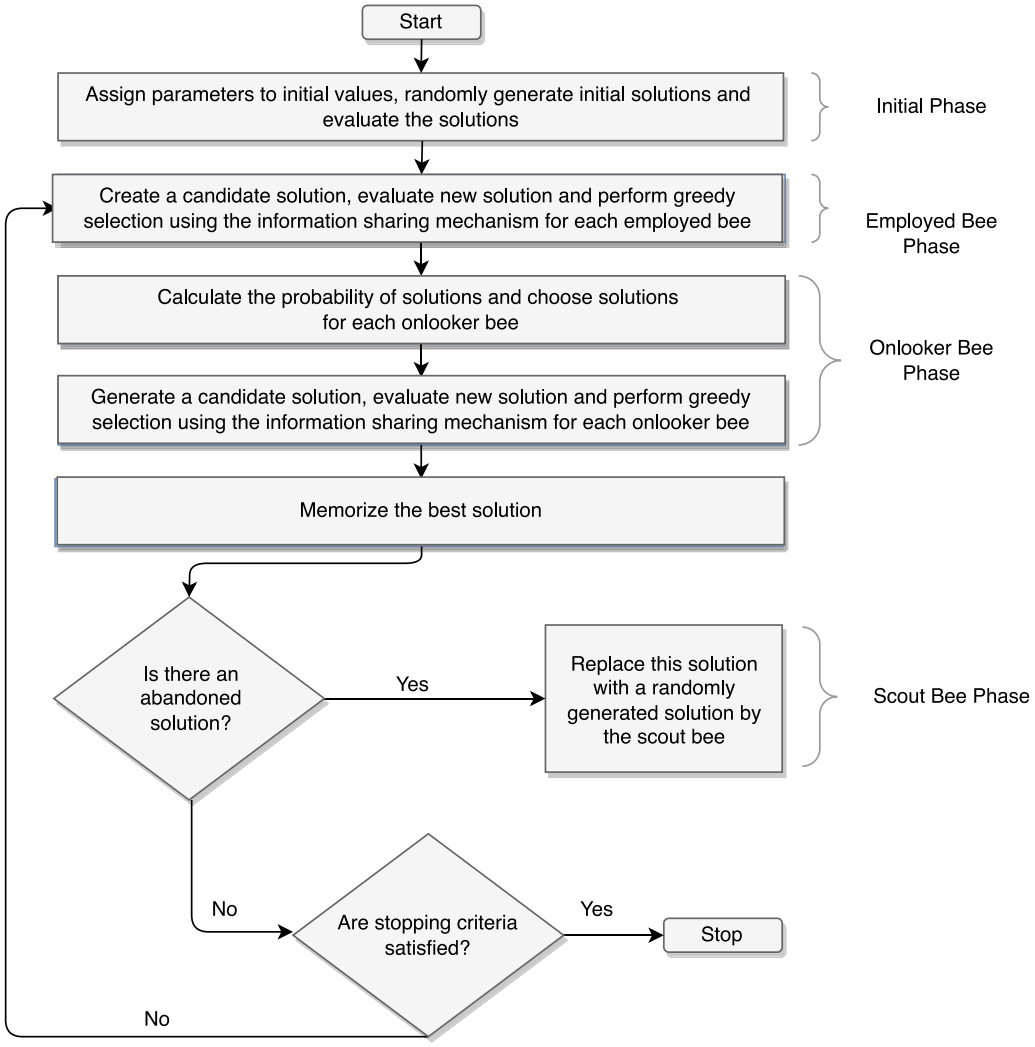

Fig. 8. ABCP flowchart.

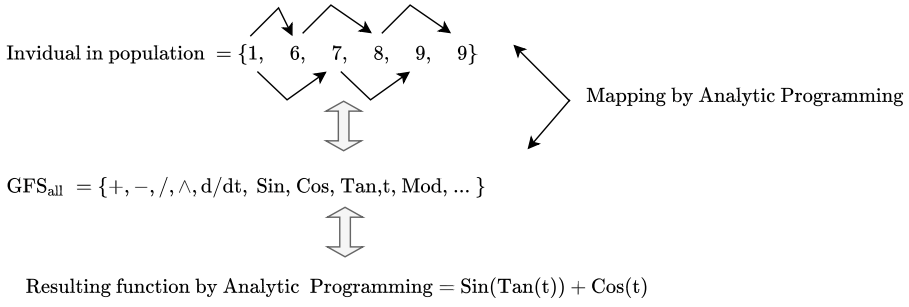

Fig. 9. The main process of AnP [38].

methods in these topics according to how the solutions are presented in Table 1. We have also summarized which benchmark algorithms are most commonly used for which types of encoding.

3.1. Studies of methods whose encoding schemes are parse trees

3.1.1. Studies using genetic programming

Symbolic regression aims to build a mathematical model of the relationships between independent variables and dependent variables by using symbolic expressions [9]. AP methods are used in SR problems because they can formulate these relationships with the solutions they generate without a preliminary model.

GP and its versions have proven themselves in many SR problems. These versions include hybrid grammar-based GP [51], which combines grammar-based GP (GGP) and ES, and multifactorial GP (MFGP) [52], which aims to optimize multiple tasks simultaneously, can be cited as examples. Many studies aimed at improving the generalization ability in solving SR problems [53,54]. In addition, various applications such as fault diagnosis [55], soccer game attendance prediction [56] have been solved as SR problems. Structural approaches [53,54,57] and semantic-based approaches [53,58] have also been proposed to successfully solve such problems

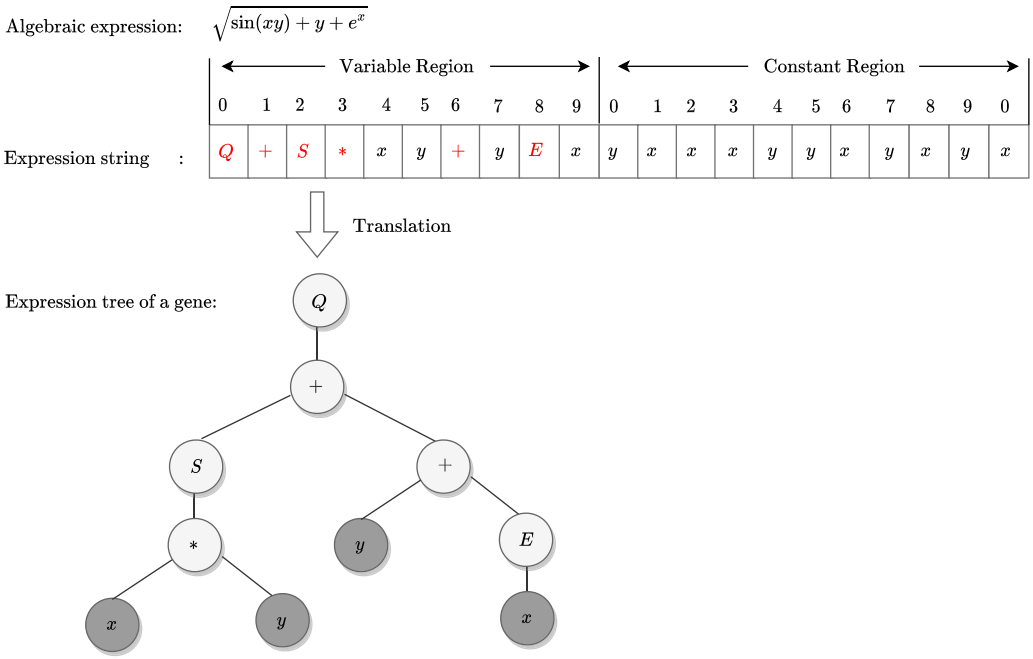

Fig. 10. CSP translation process [15].

After a predictive algorithm is trained on a dataset, it generates the predictor or model based on the results of that data in the presence of unseen data [59]. Unlike classification, there is no class label for the predictor, so unseen data can be fitted to problems such as regression-fitted time series.

AP methods are used as predictors in engineering problems in many different fields. GP and its versions have been used for prediction in various fields such as earthquakes [60], lion battery life [61], heat transfer coefficient [62], stream flow [63], breast cancer [64], backbreak in open-pit blasting [65]. The most commonly used GP version in prediction problems is multi-gene GP (MGGP) [64,66,67]. GP is mostly compared to adaptive neuro-fuzzy inference system (ANFIS) [62,68], except for conventional algorithms in such problems.

Classification determines to which data class new observations belong based on a training dataset that contains the observations and whose category is known [69]. GP and its versions (genetic texture signature (GTS) [70], stacking-GP [71]) are used for texture classification [70], document classification [72], stacked generalization [71], and exploiting semantic effects of subtrees [73]. In addition, Majeed al. improved the performance of GP, by proposing the impact-aware crossover operator [73].

Feature selection is another method of data analysis that aims to reduce computational complexity by identifying the required or relevant features for the model [74]. With successful feature selection, predictive performance and generalization ability can be improved. An attempt to identify significant/relevant features for high-dimensional test datasets was made using GP [75–77]. However, it has been adapted to feature selection problems in many different domains, e.g., microprocessor design [78], Alzheimer's diagnosis [79]. Ma and Gao have proposed two different multi-objective versions GP (genetic programming multi Objective (GPMO) and feature selection genetic programming multi objective (FSGPMO)) for these problems [77].

Job scheduling is about prioritizing jobs so that companies can use their available resources more efficiently [80]. GP has been used in extracting rules specifically for scheduling problems [81,82].

Various techniques such as digital imaging, object tracking/diagnosis, rotation, classification, restoration are used in many different fields such as biology, geology, meteorology. Classification [83–85], restoration [86], recognition [87] were performed on digital images using GP. It should be noted that GP has been compared with convolutional neural network (CNN) in many studies [83–85,87]. In addition to the test datasets, the studies also used [83,84,86] face datasets [85,87].

All applications not included in other problem topics were considered "complex problems". Although GP is the first proposed method in AP methods, it is still up-to-date and usable with different versions. For this reason, it is still used to solve many real-world problems in different areas and to compare with newly proposed AP methods. Burned area estimation [88], quality development of pharmaceutical mini-tablets [89], design of hash functions for ipv6 [90] are some of these problems. GP also hybridizes with various algorithms such as genetic programming and reinforcement learning (GPRL) [91], fuzzy inference system strong typed GP (FISTGP) [92], model selection criteria approximated GP (MSC-GP) [93], program synthesis via model finder 2 (PSMF2) [94]. There are also several studies to improve the performance of GP, such as avoiding overfitting by reducing complexity [95], measuring genotypic and phenotypic convergence [96], estimating sampling error based on population size [97].

3.1.2. Studies using ant programming

AntP [7], generalized AntP (GAP) [33], ant colony programming (ACP) [98] methods were developed using GP and ACO. Among the other AntP-based versions, the most commonly used are dynamic AntP-DAP [36,99], Enhanced generalized AntP EGAP [100,101], grammar-based AntP-GBAP [34,102], cartesian AntP-CAP [35]. There are also parallel versions of AntP [28,103] and multi objective versions [103,104]. One of the benchmarking algorithms in all studies with GBAP is Ant-Miner [34,102,105].

When AntP studies have been investigated, it has mostly been used for optimal control of fuzzy systems [104,106–108], solving differential equations [109,110], for developing node release mechanisms [111,112]. Santa Fe ant trail and multiplexer problems are also solved using different versions of AntP [100,101].

Table 2
Literature review of methods whose encoding schemes are parse trees.

Primary	Method	Problem topic	Publications
GP-based	Hybrid GGP, ADGSGP, SRM-Driven GP, GP, MGGP, SLP-CHC, SBGP, MFGP	Symbolic regression	[51–58,125]
	EP-GPBoost, MGGP, NARX, GP,	Prediction	[60–68,126,127]
	DNN, GP-SR, MSGP, MGGP	Classification	[70–72]
	GP, MOGP, PMO, FSGPMO	Feature selection	[75–79]
	MGP, GP	Job scheduling	[81,82,128,129]
	GP-FER, ConvGP, GPRM, MOGP, GP-SD, GP-PD, GP-SN, GP-PN	Image processing	[83–87]
	GPRL, GP, MSC-GP, NSGA-II, PSMF2, MOGP, STGP, CGP, softGP, Nested, Hybrid	Complex problems	[73,88–97,130]
AntP-based	ACP, GBAP, DAP, CAP, AntP AC-CGP, ACOP, ACP, GBAP-RARM, MOGBAP-RARM	Symbolic regression	[7,34–36,98,99,101,102,110–112,131–135]
	GBAP, MOGBAP, APIC, GPU-MOGBAP	Classification	[104,105,136]
	GAP, ACP, EGAP, Parallel AntP	Complex problems	[29,33,100,104,107,108,136]
ABCP-based	ABCP, Scaled ABCP, ABCPWeb, MHABCP qABCP, sABCP, qsABCP, ABCEP	Symbolic regression	[4,9,74,118–120]
	ABCP	Feature selection	[113,114]
	ABCP, shABCP, ABCP-descriptor	Complex problems	[3,69,115–117]
BBP-based	BBP, Linear BBP, Constrained BBP	Prediction	[59,121–124]

3.1.3. Studies using artificial bee colony programming

The standard version of ABCP was used to solve various problems, such as feature selection [113,114], classification [115], artificial ant problem [116], image processing [69], definition of intensity-duration-frequency relationship [117]. At the same time, the cloud-based web application ABCPWeb was also developed [118]. The application aims to solve various problems with ABCP without having any programming knowledge.

ABCP has major shortcomings such as convergence rate and high localization [4,119]. Attempts have been made to address these shortcomings by modifying the parse tree structure of ABCP or adding new operators. One of the settings in the tree structure is proposed as Scaled ABCP. In this method, bias and scaling factors are added to the trees [120]. Another setting is proposed as multi-hive ABCP (MHABCP) [74]. In the studies, the maximum tree depth of the standard ABCP is 15 [4,9,119]. In MHBCP, this parameter is chosen lower, e.g., 4 or 5. A solution is not represented by a single tree, but by a linear combination of several trees with low depth. Thus, the trees grow horizontally and not vertically. In artificial bee colony expression programming (ABCEP), solutions are expressed as linear string arrays consisting of heads and tails [4]. By computing the distance between the trees using qABCP, the algorithm tries to optimize the quality of the best neighbor solution [119]. Moreover, in this study, a semantic operator was adapted to ABCP for the first time. In another study, the convergence rate was improved by periodically decreasing the colony size of ABCP [3]. Almost all studies using ABCP were compared with GP-based methods.

3.1.4. Studies using biogeography-based programming

BBP and its versions have been used in civil engineering to solve prediction problems in various subfields [59,121–124]. BBP is most compared to ABCP [10,122], artificial neural network (ANN) [59,121,123], and GP [10,123].

3.1.5. Studies using deterministic technique for equation development

Since DoME [11] is a new algorithm from 2022, to the best of our knowledge, there is no study yet that solves problems other than SR or proposes its version. The literature review of methods whose encoding schemes are parse trees is shown in Table 2. Except for the SR problem, AISP [8] and DoME [11], which have not yet been adapted to problems, are not included in this table.

3.2. Studies of methods whose encoding schemes are linear arrays

3.2.1. Studies using grammatical evolution

With its numerous versions, GE has solved both complex world problems and SR problems. SR with attribute GE [137] and christiansen GE (CGE) [138], pseudorandom number generator design with shared GE (SGE) [139], maximizing symmetry of building facades with split GE (SpGE) [140] problems solved. Unlike other AP methods, GE [141] is used to generate automatic composition of musical melodies. In another study, GE automatically generated basic plagiarism techniques from students [142]. These studies also show that AP methods can solve complex problems in many different domains.

3.2.2. Studies using analytic programming

In the literature review, most studies have solved SR problems with AnP. There are also 2 important versions called variational AnP (VAP) [143] and binary AnP (BAP) [43]. AnP has been compared or used in 11 studies self-organizing migration algorithms (SOMA) [38–42,144–149], 11 studies Differential Evolution (DE) [38–41,144–146,148,150–152], 4 studies genetic algorithm (GA) [38,143,153,154]. Therefore, although the scope of the studies varies, it should be noted that they are generally successful against algorithms that have been previously compared, such as SOMA. Santa fe trail (SFT), one of the artificial ant problems [39,40,145] and time series prediction [151,152] are among the problems studied with AnP.

3.2.3. Studies using other methods that use linear arrays

Apart from the studies proposing PSOP [45], CSP [15], HP [16] methods, no other problem has been solved by these methods. The literature review of other methods whose encoding schemes are linear arrays (GE [12] and AnP [13]) is presented in Table 3.

3.3. Studies of methods whose encoding schemes are gen expression trees

Two methods (AFSP and FP) that encode their solutions as gene expression trees in their standard versions are listed in Table 1. As mentioned earlier, AFSP [18] models the movement, prey, pursuit, and avoidance behaviors of fish flocks using the GEP scheme. GFA, the first proposed method [17] of two FA-based AP methods, represents solutions with gene expression trees. FP [47], the other method, represents solutions as parse trees. Neither method has any application other than the SR problem. Due to the lack of studies, a summary table on these methods is not given.

3.4. Studies of methods whose encoding schemes are other schemes

3.4.1. Studies using evolutionary programming

Using EP, various problems such as the unit commitment problem [160], robotic path planning [161], scheduling problems [162] have been solved. Several studies have used fast immunized EP (FI-EP) [163] inspired by immune system and

Table 3
Literature review of methods whose encoding schemes are linear arrays.

Primary	Method	Problem topic	Publications
GE-based	Attribute GE, CGE, Semantic filter, Dynamic, Depth-aware dynamic GE, SGE, Split GE	Symbolic regression Complex problems	[137,138,155] [139–142,156–159]
AnP-based	AnP, VAP, BAP AnP	Symbolic regression Complex problems	[38–43,143–149,152–154] [150,151]

Table 4
Literature review of methods whose encoding schemes are other schemes.

Primary	Method	Problem topic	Publications
EP-based	FI-EP, MCEP, ABC-EP MOEP EP	Symbolic regression Job scheduling Complex problems	[161,163,164] [162] [160,173–175]
IP-based	ISP Chaos IP, ILNEP, IP, MIIGA, MOCMIEP, RankIP	Image processing Complex problems	[168] [165–167,170,176]
PME-based	PME, EBR PME	Symbolic regression Complex problems	[21,171] [172]
BIBP-based	MBB	Symbolic regression	[50]

artificial bee colony-evolutionary programming (ABC-EP) [161] inspired by ABC to improve the performance of EP. Many studies have compared EP with metaheuristics such as ABC, GA, PSO [161,162,164]. There is also a multi-objective version of EP, multi-objective evolutionary programming (MOEP) [162] in the literature.

3.4.2. Studies using immune programming

Similarly, AP methods inspired by immune system mechanisms have solved SR problems [8,20,37]. IP-inspired rank IP is used in web search [165], chaos IP in clustering [166], immune log-normal evolutionary programming (ILNEP) in distribution problems [167], immune system programming (ISP) in medical image segmentation [168]. There are also multi-objective versions of IP based on GA [169] and EP [170].

3.4.3. Studies using parse matrix evolution and block building programming

Methods using PME apply to SR [21,171] and invariant detection [172]. Although BIBP [22] is a newer algorithm compared to other methods, its improved version using the same mathematical method is available in Multilevel Block Building (MBB) [50]. However, these methods have not yet been applied to real-world problems. Literature review of methods whose encoding schemes are other schemes is given Table 4.

3.5. Analysis of benchmark algorithms according to encoding schemes

In this subsection, the algorithms are numerically evaluated in comparison with AP methods in studies according to encoding schemes. The number of times which algorithm is used for comparison is given in Table 5. The total value of the counting column differs from the number of publications discussed in this review. There are two reasons for this. The first reason is that algorithms that were used only once for comparison are not included in the table. The second reason is that one or more algorithms are compared in one study.

When the publications in the review were grouped according to Table 1; there were 99 papers with methods whose encoding schemes are parse trees, 32 papers with methods whose encoding schemes are linear arrays, 3 papers with methods whose encoding schemes are gene expression trees, and 21 papers with methods whose encoding schemes are other encoding schemes in the standard version. The number of comparison algorithms is higher than the other groupings, which is due to the abundance of studies with parse trees. In addition, it can be seen from Table 5 that, GP, which maintains its modernity among all methods, is the most commonly used comparison algorithm. Although comparison algorithms such as DE (11), SOMA (11), SA (5) are compared with the methods that use linear array encoding schemes, these algorithms were not used in other groupings. In addition to ANN, CNN, a newly developed deep learning algorithm, was also used in comparisons for methods whose encoding schemes are parse trees.

Table 5
Benchmark algorithms according to encoding schemes.

Encoding	Benchmark algorithm	Counting
Parse tree	ABCP	7
	ANFIS	3
	ANN	11
	Ant-Miner	3
	CAP	3
	CNN	4
	Conventional CAP	3
	GP	26
	JRIP	3
	KNN	4
	LBP	2
	MLR	2
	NB	5
	PSO/ACO	3
	RK	3
	SVM	4
Linear array	DE	11
	GA	4
	GE	3
	GP	3
	PSO	2
	SA	5
	SOMA	11
Gen expression tree	GEP	3
Other encoding schemes	ANN	2
	EP	3
	GP	2
	LDSE	2

4. Current metrics of automatic programming

Now that we have reviewed all the research papers that deal specifically with the methods and applications of AP, we can present some general statistics on the current state of the field. The number of papers we have included in our review is 155.

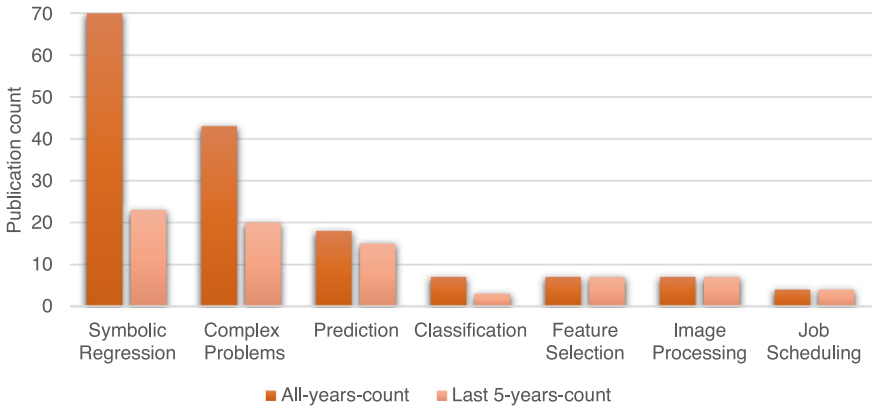

Fig. 11. The histogram of publication count in problem types.

We analyzed the studies according to important criteria such as the year of publication, the application domain, the benchmark algorithms they were compared to, and the application goals. We will now summarize our remarkable observations to present the main highlights at AP.

Fig. 11 shows the various problems that researchers have solved using the AP methods. As expected, SR problems are most often solved using methods. The fact that these problems have been studied extensively is due to the nature of AP methods. However, other types of problems seem to be less explored, and it is seen as an opportunity to work in these areas. Other important topics in the engineering applications are complex problems and prediction. Optimal control of systems, disease detection, estimation, solution of differential equations, and design problems can be cited as examples of complex engineering problems. As for problem topics, the most interesting area in recent years has been SR. Our review collected 69 papers on SR, 43 on complex problems, and 18 on prediction. These studies account for more than 80% of all studies and have the most research areas. In addition, there are 7 papers each on classification, feature selection, and image processing problems, and 4 papers on job scheduling problems. All papers in these topics were published in the last 5 years.

Fig. 12 provides more details on the selection of AP methods for the various problem topics. As can be seen in Fig. 12, symbolic regression and complex problems were solved using the majority of AP methods. In addition, it can be seen that GP is used for all other problem topics. Except for complex and symbolic regression problems, AntP was mainly used for classification problems and not for other problems. It can be seen that ABCP is not used in classification, job scheduling and prediction problems, but it can be assumed that it can be applied to these problem topics due to its structural similarity with GP. From the map, it can be interpreted that AnP has been studied a lot in SR problems, but hardly in other problems. Therefore, applying AnP to these problems and comparing it with other algorithms can be recommended in future studies to fill the gap in this area. Compared to other methods, the newly developed BBP mainly solves prediction problems and has the least number of publications among all methods. For this reason, BBP is expected to contribute to the literature by applying it to other problems. In the context of the discussed problems, it can be seen that the methods GE, IP and EP are similarly applied to complex problems with SR and their applications to other problems are almost nonexistent.

Fig. 13 shows the distribution of methods used in the studies collected over the last 5 years. From the figure, it can be seen that most of the studies were performed on GP, with ABCP in second place. As shown in the figure, GP is the most active among all methods still in use. In recent years, more studies have been conducted with ABCP than with any other of the proposed methods. There are studies on MBB and DOME, proposed in the last 5 years, in which only methods are presented. For this reason, it can be said that there is a lack of study related to the methods.

5. Discussion and open issues

5.1. Open issues and future work

From the beginning of AP, it has increasingly attracted the attention of AI researchers, and there are several methods that attempt to solve many problems. GP is steadily growing among AP methods as it inspires researchers to develop new algorithms. It also presents methods that are superior to GP in many areas, such as solution quality and convergence speed. Although each method attempts to be developed using different methodologies, the representation of individuals is similar in issues such as improvement operators. Therefore, O'Neil and Spector stated that there are several obstacles to the successful implementation of their AP method [177]. If we wish to summarize these obstacles, we arrive at the following basic headings:

- *Development of robust methods*: In developing the methodology of AP methods, researchers should answer the following questions:
  - How are the individuals represented? In tree form, grammar-based, matrix-based, etc. With different representations of individuals, the methods may perform differently depending on the application and the problem. The performance of grammar-based and matrix-based solutions can be compared to tree-based solutions. In the case of tree-form structures, it should be checked whether the tree can generate a defined mathematical equation. For example, if an expression in one part of the tree attempts to divide by zero, then the equation generated from the tree is undefined. In such cases, the researcher can express functions such as division and logarithm functions as protected functions. This means that in case the value to be processed returns an undefined value (for example, if the divisor value is zero), it will be returned as another number, otherwise a normal division will be performed [115]. In addition, in some cases there may be structures in the calculation of the numeric value expressed by the subtrees that may cause problems in the calculation of the numeric values expressed by the subtrees. For example, the value determined in a subtree may be so large that it exceeds the maximum value of the data

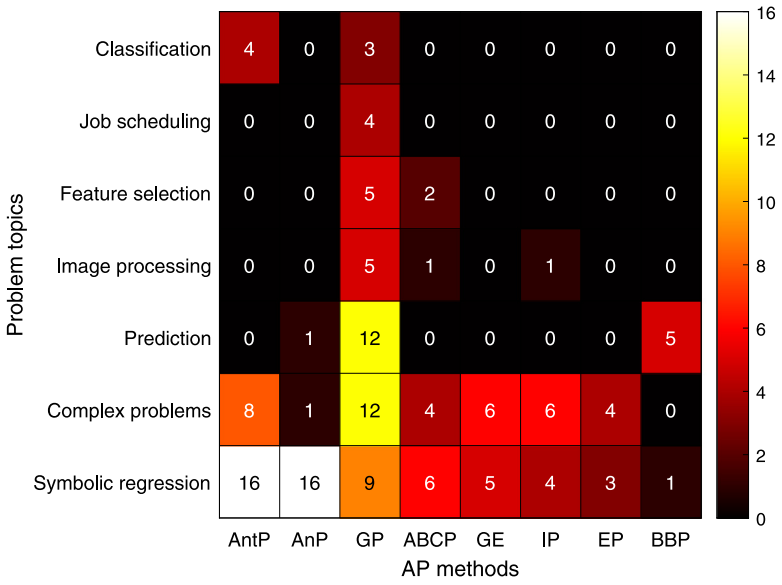

Fig. 12. Problem topics-AP methods heatmap.

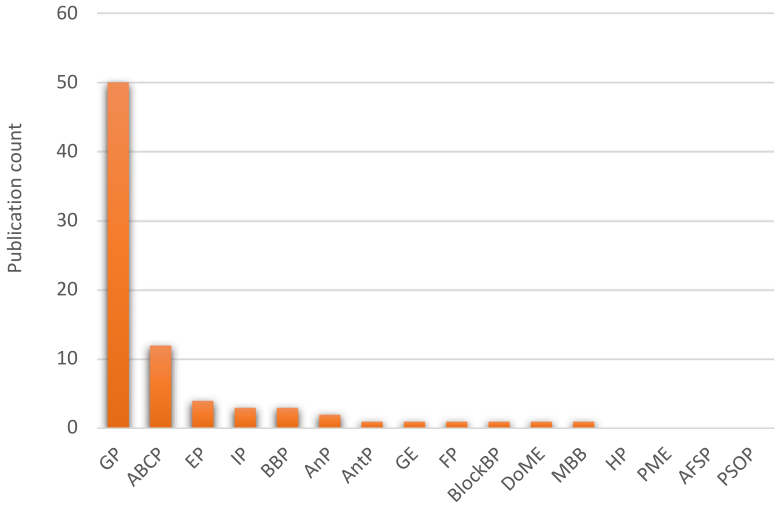

Fig. 13. According to problem topic-AP method distributions.

type used in coding the algorithm. Koza explained that in such cases, operations can be continued by assigning a suitable value to the subtree [6].

In AP solutions are usually generated by selecting from the terminal set and the function set. In the terminal set are constants and dependent variables such as x, y. The function set includes arithmetic, trigonometric and logical functions, conditional expressions and loops, including recursive structures. The iterative use of these functions in the algorithm and the fact that their use is open to evolutionary mechanisms leads to programs that cannot be stopped [177]. The halting problem is another important problem that researchers are trying to solve and is open for development.

- What could be the main source of inspiration? Depending on the source of inspiration, one can determine which steps of the algorithm are developed that are similar to the steps of the algorithm. However, although the steps are similar, some of the equations and solution steps used in the algorithm may not be used depending on the representation of the individuals. For example, the equations used in the ABC algorithm for solution generation and improvement are not used in ABCP. As mentioned in Section 2.1.4, the solutions in ABCP are generated in the form of parse trees, and their improvement is done through the information-sharing mechanism.
- Which operators/mechanisms will be used? Researchers can either adapt the operators of the known methods or develop their own operators for the new method they want to develop without affecting the basic operations of the algorithm they are inspired by.
- What is the success of the developed method according to different performance criteria? It should be compared whether the method achieves the optimal value for different comparison problems and what is the convergence rate. The presented methods are often compared with the standard version of the method and/or the standard GP, if available. In addition, the Big O(n) notation of the presented algorithm should be calculated.

Considering the above, it is quite difficult to develop a robust AP method that can handle the various possibilities that occur during execution. According to our study, many swarm intelligence algorithms do not have adaptation as AP method. Therefore, developing such adaptations could be interesting for researchers in new studies. However, combining the strengths of AP methods and proposing hybrid methods could provide promising performance improvements for various applications.

- *Determining the AI ratio*: Koza defined the ratio between the intelligence provided by the automatic operation of a developed method and the intelligence provided by the human using the method as the AI ratio [178]. The AI ratio is specific to each problem. He tried to determine the AI ratio of GP for different problems by qualitatively evaluating the quantities "A" and "I". The execution steps of the method are independent of the problem and the human contribution, which are automatically executed during the execution of the method. Therefore, the results of the execution steps provide the amount of "A". The set of "I" is expressed by the human as the preparatory steps before starting the execution of the method. For example, the preparatory steps for GP can be expressed as terminal and function sets, evaluation criteria, parameters, and stopping criteria used in generating solutions, as we mentioned in "Development of robust methods". Samuel expressed the criterion of AI and machine learning as the behavior of machines that, when performed by humans, is assumed to involve the use of intelligence [179]. Koza found that this criterion is met with results produced by GP for various problems that are comparable to humans and exhibit a large amount of "A" capability [178]. In other words, the AI rate is high. What are the AI rates of the AP methods (Table 1) that we interpret with other developed high-level programming languages? To the best of our knowledge, there is no study in the literature on the AI rates of these methods. As we analyzed in the current metrics in Section 4, most SR problems in the literature have been solved using AP methods. What is the method with the highest AI rate for SR problems? To generalize the question further, which method has the highest AI rate for which problem? Therefore, it can be said that the evaluation of AI rates for different application areas of methods other than GP is one of the most important questions yet to be developed.
- *Syntax dependency*: While GP-based methods have been proposed, grammar/syntax-based schemas have gained prominence in modeling behaviors. In these schemas, individuals that exhibit syntactically similar but semantically different behaviors are matched. Therefore, new individuals with semantically different fitness values are generated during evolution. For this reason, Zojaji et al. proposed a semantic-based scheme and found that syntactic schemes cannot reliably predict behavior GP [180]. Proposing semantic schemes to find semantically similar individuals is essential for AP methods to enable better search with high-level structures with complex semantics such as loops and recursions. While there are studies in this area related to GP, there is almost no study on the syntactic/semantic behavior of other methods. Proposing these schemas of AP's methods in the literature will contribute significantly to the development of AP.
- *Parallelization of AP methods*: In our study, there is only one study on the computational time required to run the methods, the quality of the solutions, and the improvement due to parallel computational methods [103]. Therefore, parallelization of methods and evaluation of computational complexity and speed could also be one of the future aspects of AP.
- *Parameter tuning*: All methods of AP have parameters that depend on the algorithm method from which they are inspired. Setting the parameters correctly has a significant impact on the performance of the method. Many newly introduced methods use parameter values from the work on which they are based to allow fair comparison. However, the adjustment of the parameters of the method is done by the researcher. The more expert knowledge the researcher has about the method, the more accurately the researcher can adjust the parameters. We believe that more work should be done in this area. Suggestions and algorithms can be developed to help beginners use the methods without prior knowledge.
- *Solve real world problems*: AP has many useful features compared to traditional algorithms, such as optimizing the relationship between input and output variables, mathematical modeling. When we examine our review literature tables on AP, we find that the most studied problem topics are SR, prediction, feature selection, artificial ant problem, and optimal control. Thus, although we have seen applications of AP methods in many domains, we believe that they can also solve larger real-world problems given the success of conventional algorithms.
- *Development of AP-based toolboxes*: if you look at all the methods from AP, you will find that they have more complex architectures than many algorithms. Lack of programming skills by researchers is a major obstacle for AP studies. There are open source applications related to GP and its versions [181,182]. However, since most AP algorithms are not open source, it becomes difficult to conduct comparative studies using AP methods. For this reason, the development of user-friendly web-based and non-web-based toolboxes that facilitate the use of AP methods will be of great benefit to researchers. The ABCPWeb application can serve as an example of such studies [118].

5.2. Answers to research questions

The answers to the research questions based on our observations are listed below.

- Which AP methods are used for which problem solutions?
  Answer: In general, we can say that all AP methods solve SR problems. However, BBP is used for prediction and AntP for matrix riccati differential equation (MRDE). It is worth mentioning that GP is the most frequently used method for all problem topics in our study. As shown in Fig. 12, we classified all problem topics into 7 categories and plotted the number of publications of the methods according to the topics on the heat map.
- With which conventional algorithms can the performance of AP methods be compared?
  Answer: The methods are compared with very different conventional algorithms specific to the application. When studies have developed a version, it has been compared to the standard of that version. For example, CGE is compared to GE in [138]. It should be noted that in studies, methods are usually compared to GP. For conventional algorithms, we find a very wide range from CNN to metaheuristics. Also, we can see that comparisons are made with different tools such as Eureqa [50].
- In what direction will the field of AP develop?
  Answer: In our studies, we have discovered more than one developable issue related to AP. We draw researchers' attention to important issues that require in-depth analysis, such as developing methods, AI ratio, syntax dependency,

improving their speed, parameter tuning, testing solutions to large problems, coding open source AP software. For this reason, we believe that the growing AP will continue to develop once the above problems are solved.

6. Conclusion

AP is the process of producing a model of the system using the necessary information. AP methods can solve complex problems in various fields such as engineering, medicine, bioinformatics, chemistry, transportation, finance, and so on. With the development of evolutionary algorithms and increasing computation power and high-performance processors, new and powerful AP methods will be proposed and applied. More complex real-world problems will be solved more accurately and quickly due to this progress. We believe that this review will be a starting point for researchers to solve various optimization and machine learning problems using AP methods and to develop new AP methods.

Glossary

ABCEP Artificial Bee Colony Expression Programming.
ABC-EP Artificial Bee Colony-Evolutionary Programming.
ABCP Artificial Bee Colony Programming.
AC-CGP Ant Colony-Cartesian Genetic Programming.
ACO Ant Colony Optimization.
ACOP Ant Colony Optimization for Linear Imperative Programming.
ACP Ant Colony Programming.
ADGSGP Angle Drive Selection based Geometric Semantic GP.
AFSP Artificial Fish Swarm Programming.
AI Artificial Intelligence.
AIS Artificial Immune System.
AISP Artificial Immune System Programming.
ANFIS Adaptive Neuro-Fuzzy Inference System.
ANN Artificial Neural Network.
AnP Analytic Programming.
AntP Ant Programming.
AntTAG Ant Tree Adjunct Grammar.
AP Automatic Programming.
APIC Ant Programming for Imbalanced Classification.
BBP Biogeography-Based Programming.
BlBP Block Building Programming.
BNF Backus Naur Form.
CAP Cartesian Ant Programming.
CGE Christiansen Grammatical Evolution.
CGP Cartesian Genetic Programming.
CNN Convolutional Neural Network.
ConvGP Convolutional Genetic Programming.
CSP Clone Selection Programming.
DAP Dynamic Ant Programming.
DE Differential Evolution.
DNN Deep Neural Network.
DoME Deterministic Technique for Equation Development.
DSH Discrete Set Handling.
EBR Elite Bases Regression.
EC Evolutionary Computation.
EGAP Enhanced Generalized Ant Programming.
EP Evolutionary Programming.
EP-GPBoost Earthquake Predictor AdaBoost.
ES Evolutionary Strategies.
FA Firefly Algorithm.
FI-EP Fast Immunized-Evolutionary Programming.
FISTGP Fuzzy Inference System Strongly Typed Genetic Programming.
FP Firefly Programming.
FSGPMO Feature Selection Genetic Programming Multi Objective.
GA Genetic Algorithm.
GACP Grid Ant Colony Programming.
GAP Generalized Ant Programming.
GBAP Grammar-Based Ant Programming.
GBAP-RARM Grammar-Based Ant Programming for Rare Association Rule Mining.
GE Grammatical Evolution.
GEP Gen Expression Programming.
GFA Geometric Firefly Algorithm.
GFS General Functional Set.
GGP Grammar-based Genetic Programming.
GH Grammatical Herding.
GP Genetic Programming.
GP-FER GP-Facial Expression Recognition.
GPMO Genetic Programming Multi Objective.
GP-PD Standard GP Using Local Pixel Information.
GP-PN Neat-GP Using Local Pixel Information.
GPRL Genetic Programming for Reinforcement Learning.
GPRM Genetic Programming with Representation and Mutation.
GP-SD Standard GP Using Local Image Statistics.
GP-SN Neat-GP Using Local Image Statistics.
GP-SR Genetic Programming Symbolic Regression.
GPU-MOGBAP Graphic Processing Unit-Multi-Objective Grammar Based Ant Programming.
GS Grammatical Swarm.
GTS Genetic Texture Signature.
HP Herd Programming.
HSI Habitat Suitability Index.
IGP Inductive Genetic Programming.
ILNEP Immune Log-Normal Evolutionary Programming.
IP Immune Programming.
ISP Immune System Programming.
JRIP Decision Trees and Repetitive Incremental Pruning Algorithms for Error Reduction.
LBP Local Binary Pattern.
LDSE Low Dimensional Simplex Evolution.
MBB Multilevel Block Building.
MCEP Momentum Coefficient Evolutionary Programming.
MFGP Multifactorial Genetic Programming.
MGGP Multi Gene Genetic Programming.
MGP Memetic Genetic Programming.
MHABCP Multi-Hive Artificial Bee Colony Programming.
MIIGA Multi Objective Interval-Valued Immune Genetic Approach.
MLR Multi-Linear Regression.
MOCMIEP Multi-Objective Chaotic Mutation Immune Evolutionary Programming.
MOEP Multi-Objective Evolutionary Programming.
MOGBAP Multi-Objective Grammar-Based Ant Programming.
MOGBAP-RARM Multi-Objective Grammar-Based Ant Programming for Rare Association Rule Mining.
MOGP Multi Objective Genetic Programming.
MRDE Matrix Riccati Differential Equation.
MSC-GP Model Selection Criteria approximated Genetic Programming.
MSGP Multi-Stage Genetic Programming.
NARX Nonlinear AutoRegressive with eXogenous inputs.
NB Naive Bayes.
NSGA-II Non-Dominated Sorting Genetic Algorithm-II.
PME Parse Matrix Evolution.
PSMF2 Program Synthesis via Model Finder 2.
PSO Particle Swarm Optimization.
PSOP Particle Swarm Optimization Programming.
qABCP quick Artificial Bee Colony Programming.
qsABCP quick semantic Artificial Bee Colony Programming.
SA Simulated Annealing.
sABCP semantic Artificial Bee Colony Programming.

SBGP Semantic Schema-Based Genetic Programming.
SFT Santa Fe Trail.
SGE Shared Grammatical Evolution.
shABCP shrinking Artificial Bee Colony Programming.
SLP-CHC Straight Line Program with CHC Algorithm.
SOMA Self-Organizing Migration Algorithms.
SPs Security Procedures.
SR Symbolic Regression.
SRM-Driven GP Structural risk minimization-driven Genetic Programming.
SVM Support Vector Machine.
TS Tabu Search.
VAP Variational Analytic Programming.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

No data was used for the research described in the article.

Acknowledgments

This project was supported by Scientific Research Project Foundation of Erciyes University (Project ID: FBG-2022-11885).

References

[1] M. O'Neill, Automatic Programming in an Arbitrary Language: Evolving Programs with Grammatical Evolution (Ph.D. thesis), University Of Limerick, 2001.
[2] A.L. Samuel, Some studies in machine learning using the game of checkers. II—Recent progress, IBM J. Res. Dev. 11 (6) (1967) 601–617.
[3] F. Boudardara, B. Gorkemli, Solving artificial ant problem using two artificial bee colony programming versions, Appl. Intell. 50 (11) (2020) 3695–3717.
[4] M. Nekoei, S. Moghaddas, E. Golafshani, A. Gandomi, Introduction of ABCEP as an automatic programming method, Inform. Sci. 545 (2021) 575–594, http://dx.doi.org/10.1016/j.ins.2020.09.020.
[5] J.L. Olmo, J.R. Romero, S. Ventura, Swarm-based metaheuristics in automatic programming: a survey, Wiley Interdiscip. Rev.: Data Min. Knowl. Discov. 4 (6) (2014) 445–469.
[6] J.R. Koza, Genetic Programming: On the Programming of Computers by Means of Natural Selection, Vol. 1, MIT Press, 1992.
[7] O. Roux, C. Fonlupt, Ant programming: or how to use ants for automatic programming, in: Proceedings of ANTS, Vol. 2000, Springer, Berlin, 2000, pp. 121–129.
[8] C.G. Johnson, Artificial immune system programming for symbolic regression, in: European Conference on Genetic Programming, Springer, 2003, pp. 345–353.
[9] D. Karaboga, C. Ozturk, N. Karaboga, B. Gorkemli, Artificial bee colony programming for symbolic regression, Inform. Sci. 209 (2012) 1–15.
[10] E.M. Golafshani, Introduction of Biogeography-Based Programming as a new algorithm for solving problems, Appl. Math. Comput. 270 (2015) 1–12.
[11] D. Rivero, E. Fernandez-Blanco, A. Pazos, DoME: A deterministic technique for equation development and Symbolic Regression, Expert Syst. Appl. 198 (2022) 116712.
[12] C. Ryan, J.J. Collins, M.O. Neill, Grammatical evolution: Evolving programs for an arbitrary language, in: European Conference on Genetic Programming, Springer, 1998, pp. 83–96.
[13] I. Zelinka, Analytic programming by means of soma algorithm, in: Proceedings of the 8th International Conference on Soft Computing, Mendel, Vol. 2, 2002, pp. 93–101.
[14] M. O'Neill, A. Brabazon, Grammatical swarm, in: K. Deb (Ed.), Genetic and Evolutionary Computation – GECCO 2004, Springer Berlin Heidelberg, Berlin, Heidelberg, 2004, pp. 163–174.
[15] Z. Gan, T.W. Chow, W.N. Chau, Clone selection programming and its application to symbolic regression, Expert Syst. Appl. 36 (2) (2009) 3996–4005.
[16] C. Headleand, W. Teahan, Grammatical herding, J. Comput. Sci. Syst. Biol. 6 (2013) 043–047.
[17] A. Husselmann, K. Hawick, Geometric firefly algorithms on graphical processing units, in: Cuckoo Search and Firefly Algorithm, Springer, 2014, pp. 245–269.
[18] Q. Liu, T. Odaka, J. Kuroiwa, H. Ogura, Application of an artificial fish swarm algorithm in symbolic regression, IEICE Trans. Inf. Syst. 96 (4) (2013) 872–885.
[19] D.B. Fogel, Evolutionary Computation: Toward a New Philosophy of Machine Intelligence, EEE Press, Piscataway, NJ., 1995.
[20] P. Musilek, A. Lau, M. Reformat, L. Wyard-Scott, Immune programming, Inform. Sci. 176 (8) (2006) 972–1002.
[21] C. Luo, S.-L. Zhang, Parse-matrix evolution for symbolic regression, Eng. Appl. Artif. Intell. 25 (6) (2012) 1182–1193.
[22] C. Chen, C. Luo, Z. Jiang, Block building programming for symbolic regression, Neurocomputing 275 (2018) 1973–1980.
[23] N.L. Cramer, A representation for the adaptive generation of simple sequential programs, in: Proceedings of the 1st International Conference on Genetic Algorithms, L. Erlbaum Associates Inc., USA, 1985, pp. 183–187.
[24] R. Poli, W.B. Langdon, N.F. McPhee, J.R. Koza, A Field Guide to Genetic Programming, Lulu. com, 2008.
[25] I. BoussaïD, J. Lepagnot, P. Siarry, A survey on optimization metaheuristics, Inform. Sci. 237 (2013) 82–117.
[26] P.J. Hancock, An empirical comparison of selection methods in evolutionary algorithms, in: AISB Workshop on Evolutionary Computing, Springer, 1994, pp. 80–94.
[27] D. Karaboğa, Yapay Zeka Optimizasyon Algoritmaları, Nobel Akademi Yayıncılık, 2014.
[28] A. Hara, J.-i. Kushida, S. Tanabe, T. Takahama, Parallel ant programming using genetic operators, in: 2013 IEEE 6th International Workshop on Computational Intelligence and Applications, IWCIA, IEEE, 2013, pp. 75–80.
[29] M. Boryczka, Ant colony programming for approximation problems, in: Intelligent Information Systems 2002, Springer, 2002, pp. 147–156.
[30] M. Boryczka, Z.J. Czech, Solving approximation problems by ant colony programming, in: Proceedings of the 4th Annual Conference on Genetic and Evolutionary Computation, 2002, p. 133.
[31] S.A. Rojas, P.J. Bentley, A grid-based ant colony system for automatic program synthesis, in: Late Breaking Papers at the 2004 Genetic and Evolutionary Computation Conference, Seattle, Washington, USA, Vol. 26, Citeseer, 2004, pp. 1–12.
[32] H.A. Abbass, X. Hoai, R.I. McKay, AntTAG: A new method to compose computer programs using colonies of ants, in: Proceedings of the 2002 Congress on Evolutionary Computation. CEC'02 (Cat. No. 02TH8600), Vol. 2, IEEE, 2002, pp. 1654–1659.
[33] C. Keber, M.G. Schuster, Option valuation with generalized ant programming, in: Proceedings of the 4th Annual Conference on Genetic and Evolutionary Computation, 2002, pp. 74–81.
[34] J.L. Olmo, J.R. Romero, S. Ventura, A grammar based ant programming algorithm for mining classification rules, in: IEEE Congress on Evolutionary Computation, IEEE, 2010, pp. 1–8.
[35] A. Hara, M. Watanabe, T. Takahama, Cartesian ant programming, in: 2011 IEEE International Conference on Systems, Man, and Cybernetics, IEEE, 2011, pp. 3161–3166.
[36] S. Shirakawa, S. Ogino, T. Nagao, Dynamic ant programming for automatic construction of programs, IEEJ Trans. Electr. Electron. Eng. 3 (5) (2008) 540–548.
[37] N.I. Nikolaev, H. Iba, V. Slavov, Inductive genetic programming with immune network dynamics, in: Advances in Genetic Programming, Vol. 3, MIT Press, 1994, p. 355.
[38] I. Zelinka, Z. Oplatkova, L. Nolle, Analytic programming–Symbolic regression by means of arbitrary evolutionary algorithms, Int. J. Simul.: Syst. Sci. Technol. 6 (9) (2005) 44–56.
[39] Z. Oplatková, I. Zelinka, Investigation on artificial ant using analytic programming, in: Proceedings of the 8th Annual Conference on Genetic and Evolutionary Computation, 2006, pp. 949–950.
[40] Z. Oplatkova, I. Zelinka, Santa fe trail for artificial ant with analytic programming and three evolutionary algorithms, in: First Asia International Conference on Modelling & Simulation (AMS'07), IEEE, 2007, pp. 334–339.
[41] I. Zelinka, L. Skanderova, P. Šaloun, R. Senkerik, T.T. Dao, D.V. Hoang, Analytic programming powered by chaotic dynamics, in: Nostradamus 2014: Prediction, Modeling and Analysis of Complex Systems, Springer, 2014, pp. 123–129.
[42] L. Kojecky, I. Zelinka, CUDA-based analytic programming by means of SOMA algorithm, in: International Conference on Soft Computing-MENDEL, Springer, 2016, pp. 171–180.
[43] A.I. Diveev, N. Konyrbaev, E. Sofronova, Method of binary analytic programming to look for optimal mathematical expression, Procedia Comput. Sci. 103 (2017) 597–604.

[44] I. Zelinka, Z. Oplatkova, L. Nolle, Boolean symmetry function synthesis by means of arbitrary evolutionary algorithms-comparative study, Int. J. Simul. Syst. Sci. Technol. 6 (9) (2005) 44–56.
[45] X. Wu, M. Zhao, Y. Qu, Particle swarm optimization programming, in: 2010 International Conference on Computational Aspects of Social Networks, IEEE, 2010, pp. 397–400.
[46] D. Karaboga, B. Gorkemli, C. Ozturk, N. Karaboga, A comprehensive survey: artificial bee colony (ABC) algorithm and applications, Artif. Intell. Rev. 42 (1) (2014) 21–57.
[47] M. Aliwi, S. Aslan, S. Demirci, Firefly programming for symbolic regression problems, in: 2020 28th Signal Processing and Communications Applications Conference, SIU, IEEE, 2020, pp. 1–4.
[48] D.B. Fogel, An overview of evolutionary programming, in: Evolutionary Algorithms, Springer, 1999, pp. 89–109.
[49] X. Yao, Y. Liu, G. Lin, Evolutionary programming made faster, IEEE Trans. Evol. Comput. 3 (2) (1999) 82–102.
[50] C. Chen, C. Luo, Z. Jiang, A multilevel block building algorithm for fast modeling generalized separable systems, Expert Syst. Appl. 109 (2018) 25–34.
[51] F.A. Motta, J.M. De Freitas, F.R. De Souza, H.S. Bernardino, I.L. De Oliveira, H.J. Barbosa, A hybrid grammar-based genetic programming for symbolic regression problems, in: 2018 IEEE Congress on Evolutionary Computation, CEC, IEEE, 2018, pp. 1–8.
[52] J. Zhong, L. Feng, W. Cai, Y.-S. Ong, Multifactorial genetic programming for symbolic regression problems, IEEE Trans. Syst. Man Cybern.: Syst. 50 (11) (2018) 4492–4505.
[53] Q. Chen, B. Xue, M. Zhang, Improving generalization of genetic programming for symbolic regression with angle-driven geometric semantic operators, IEEE Trans. Evol. Comput. 23 (3) (2018) 488–502.
[54] Q. Chen, M. Zhang, B. Xue, Structural risk minimization-driven genetic programming for enhancing generalization in symbolic regression, IEEE Trans. Evol. Comput. 23 (4) (2018) 703–717.
[55] W.T. Hale, E. Safikou, G.M. Bollas, Inference of faults through symbolic regression of system data, Comput. Chem. Eng. 157 (2022) 107619.
[56] G.H. Yamashita, F.S. Fogliatto, M.J. Anzanello, G.L. Tortorella, Customized prediction of attendance to soccer matches based on symbolic regression and genetic programming, Expert Syst. Appl. 187 (2022) 115912.
[57] R. Rueda, M. Cuéllar, L. Ruiz, M. Pegalajar, A similarity measure for Straight Line Programs and its application to control diversity in Genetic Programming, Expert Syst. Appl. (2022) 116415.
[58] Z. Zojaji, M.M. Ebadzadeh, H. Nasiri, Semantic schema based genetic programming for symbolic regression, Appl. Soft Comput. 122 (2022) 108825.
[59] A. Behnood, E.M. Golafshani, Predicting the dynamic modulus of asphalt mixture using machine learning techniques: An application of multi biogeography-based programming, Constr. Build. Mater. 266 (2021) 120983.
[60] K.M. Asim, A. Idris, T. Iqbal, F. Martínez-Álvarez, Seismic indicators based earthquake predictor system using Genetic Programming and AdaBoost classification, Soil Dyn. Earthq. Eng. 111 (2018) 1–7.
[61] H. Yao, X. Jia, Q. Zhao, Z.-J. Cheng, B. Guo, Novel lithium-ion battery state-of-health estimation method using a genetic programming model, IEEE Access 8 (2020) 95333–95344.
[62] S. Hosseini, M. Moradkhani, M.M. Shah, M. Edalati, General equation for flow condensation heat transfer coefficient in different orientations of helical coils of smooth tubes using genetic programming, Int. Commun. Heat Mass Transfer 119 (2020) 104916.
[63] A.D. Mehr, A.H. Gandomi, MSGP-LASSO: an improved multi-stage genetic programming model for streamflow prediction, Inform. Sci. 561 (2021) 181–195.
[64] M. Sattar, A. Majid, N. Kausar, M. Bilal, M. Kashif, Lung cancer prediction using multi-gene genetic programming by selecting automatic features from amino acid sequences, Comput. Biol. Chem. (2022) 107638.
[65] M. Sharma, H. Agrawal, B. Choudhary, Multivariate regression and genetic programming for prediction of backbreak in open-pit blasting, Neural Comput. Appl. 34 (3) (2022) 2103–2114.
[66] M.G. De Giorgi, M. Quarta, Hybrid multigene genetic programming-artificial neural networks approach for dynamic performance prediction of an aeroengine, Aerosp. Sci. Technol. 103 (2020) 105902.
[67] S. Pawanr, G.K. Garg, S. Routroy, Prediction of energy consumption of machine tools using multi-gene genetic programming, Mater. Today: Proc. (2022).
[68] Z. Gu, Y. Liu, D.J. Hughes, J. Ye, X. Hou, A parametric study of adhesive bonded joints with composite material using black-box and grey-box machine learning methods: Deep neuron networks and genetic programming, Composites B 217 (2021) 108894.
[69] S. Arslan, C. Ozturk, Artificial bee colony programming descriptor for multi-class texture classification, Appl. Sci. 9 (9) (2019) 1930.
[70] H. Ghazouani, W. Barhoumi, Genetic programming-based learning of texture classification descriptors from local edge signature, Expert Syst. Appl. 161 (2020) 113667.
[71] I. Bakurov, M. Castelli, O. Gau, F. Fontanella, L. Vanneschi, Genetic programming for stacked generalization, Swarm Evol. Comput. 65 (2021) 100913.
[72] W. Fu, B. Xue, X. Gao, M. Zhang, Output-based transfer learning in genetic programming for document classification, Knowl.-Based Syst. 212 (2021) 106597.
[73] H. Majeed, A. Wali, M. Beg, Optimizing genetic programming by exploiting semantic impact of sub trees, Swarm Evol. Comput. 65 (2021) 100923.
[74] S. Arslan, C. Ozturk, Multi hive artificial bee colony programming for high dimensional symbolic regression with feature selection, Appl. Soft Comput. 78 (2019) 515–527.
[75] F. Viegas, L. Rocha, M. Gonçalves, F. Mourão, G. Sá, T. Salles, G. Andrade, I. Sandin, A genetic programming approach for feature selection in highly dimensional skewed data, Neurocomputing 273 (2018) 554–569.
[76] K. Nag, N.R. Pal, Feature extraction and selection for parsimonious classifiers with multiobjective genetic programming, IEEE Trans. Evol. Comput. 24 (3) (2019) 454–466.
[77] J. Ma, X. Gao, Designing genetic programming classifiers with feature selection and feature construction, Appl. Soft Comput. 97 (2020) 106826.
[78] H. Jia, N. Verma, Exploiting approximate feature extraction via genetic programming for hardware acceleration in a heterogeneous microprocessor, IEEE J. Solid-State Circuits 53 (4) (2018) 1016–1027.
[79] M. Nasrolahzadeh, S. Rahnamayan, J. Haddadnia, Alzheimer's disease diagnosis using genetic programming based on higher order spectra features, Mach. Learn. Appl. 7 (2022) 100225.
[80] S. Ghanbari, M. Othman, A priority based job scheduling algorithm in cloud computing, Procedia Eng. 50 (2012) 778–785.
[81] F.J. Gil-Gala, M.R. Sierra, C. Mencía, R. Varela, Genetic programming with local search to evolve priority rules for scheduling jobs on a machine with time-varying capacity, Swarm Evol. Comput. 66 (2021) 100944.
[82] M. Djurasevic, D. Jakobovic, Selection of dispatching rules evolved by genetic programming in dynamic unrelated machines scheduling based on problem characteristics, J. Comput. Sci. 61 (2022) 101649.
[83] B. Evans, H. Al-Sahaf, B. Xue, M. Zhang, Evolutionary deep learning: A genetic programming approach to image classification, in: 2018 IEEE Congress on Evolutionary Computation, CEC, IEEE, 2018, pp. 1–6.
[84] Q. Fan, Y. Bi, B. Xue, M. Zhang, Genetic programming for feature extraction and construction in image classification, Appl. Soft Comput. (2022) 108509.
[85] Y. Bi, B. Xue, M. Zhang, Using a small number of training instances in genetic programming for face image classification, Inform. Sci. (2022).
[86] J.E. Hernandez-Beltran, V.H. Diaz-Ramirez, L. Trujillo, P. Legrand, Design of estimators for restoration of images degraded by haze using genetic programming, Swarm Evol. Comput. 44 (2019) 49–63.
[87] H. Ghazouani, A genetic programming-based feature selection and fusion for facial expression recognition, Appl. Soft Comput. 103 (2021) 107173.
[88] A.I. Cabral, S. Silva, P.C. Silva, L. Vanneschi, M.J. Vasconcelos, Burned area estimations derived from Landsat ETM+ and OLI data: comparing genetic programming with maximum likelihood and classification and regression trees, ISPRS J. Photogramm. Remote Sens. 142 (2018) 94–105.
[89] P. Barmpalexis, A. Karagianni, G. Karasavvaides, K. Kachrimanis, Comparison of multi-linear regression, particle swarm optimization artificial neural networks and genetic programming in the development of mini-tablets, Int. J. Pharm. 551 (1–2) (2018) 166–176.
[90] Y. Hu, G. Cheng, Y. Tang, F. Wang, A practical design of hash functions for IPv6 using multi-objective genetic programming, Comput. Commun. 162 (2020) 160–168.
[91] D. Hein, S. Udluft, T.A. Runkler, Interpretable policies for reinforcement learning by genetic programming, Eng. Appl. Artif. Intell. 76 (2018) 158–169.
[92] K. Michell, W. Kristjanpoller, Strongly-typed genetic programming and fuzzy inference system: An embedded approach to model and generate trading rules, Appl. Soft Comput. 90 (2020) 106169.
[93] A. Garg, S. Su, F. Li, L. Gao, Framework of model selection criteria approximated genetic programming for optimization function for renewable energy systems, Swarm Evol. Comput. 59 (2020) 100750.
[94] A. Correia, J. Iyoda, A. Mota, Combining model finder and genetic programming into a general purpose automatic program synthesizer, Inform. Process. Lett. 154 (2020) 105866.
[95] L. Vanneschi, M. Castelli, Soft target and functional complexity reduction: A hybrid regularization method for genetic programming, Expert Syst. Appl. 177 (2021) 114929.
[96] W.B. Langdon, Genetic programming convergence, Genet. Program. Evol. Mach. 23 (1) (2022) 71–104.
[97] D. Schweim, D. Wittenberg, F. Rothlauf, On sampling error in genetic programming, Nat. Comput. (2021) 1–14.
[98] J. Green, J.L. Whalley, C.G. Johnson, Automatic programming with ant colony optimization, in: Proceedings of the 2004 UK Workshop on Computational Intelligence, Loughborough University, 2004, pp. 70–77.

[99] S. Shirakawa, S. Ogino, T. Nagao, Automatic construction of programs using dynamic ant programming, in: Ant Colony Optimization Methods and Applications, InTech, 2011, p. 75.
[100] A. Salehi-Abari, T. White, Enhanced generalized ant programming (EGAP), in: Proceedings of the 10th Annual Conference on Genetic and Evolutionary Computation, 2008, pp. 111–118.
[101] A. Salehi-Abari, T. White, The uphill battle of ant programming vs. genetic programming, in: IJCCI, 2009, pp. 171–176.
[102] J.L. Olmo, J.M. Luna, J.R. Romero, S. Ventura, An automatic programming aco-based algorithm for classification rule mining, in: Trends in Practical Applications of Agents and Multiagent Systems, Springer, 2010, pp. 649–656.
[103] A. Cano, J.L. Olmo, S. Ventura, Parallel multi-objective ant programming for classification using GPUs, J. Parallel Distrib. Comput. 73 (6) (2013) 713–728.
[104] N. Kumaresan, Optimal control for stochastic singular integro-differential Takagi-Sugeno fuzzy system using ant colony programming, Filomat 26 (3) (2012) 415–426.
[105] J.L. Olmo, J.R. Romero, S. Ventura, Using ant programming guided by grammar for building rule-based classifiers, IEEE Trans. Syst. Man Cybern. B 41 (6) (2011) 1585–1599.
[106] N. Kumaresan, Optimal control for stochastic linear quadratic singular Takagi-Sugeno fuzzy system using ant colony programming, Neural Parallel Sci. Comput. 18 (1) (2010) 89–108.
[107] N. Kumaresan, P. Balasubramaniam, Singular optimal control for stochastic linear quadratic singular system using ant colony programming, Int. J. Comput. Math. 87 (14) (2010) 3311–3327.
[108] N. Kumaresan, Optimal control for stochastic linear quadratic singular periodic neuro Takagi–Sugeno (TS) fuzzy system with singular cost using ant colony programming, Appl. Math. Model. 35 (8) (2011) 3797–3808.
[109] M. Kamali, N. Kumaresan, K. Ratnavelu, Solving differential equations with ant colony programming, Appl. Math. Model. 39 (10–11) (2015) 3150–3163.
[110] M. Kamali, N. Kumaresan, K. Ratnavelu, Takagi–Sugeno fuzzy modelling of some nonlinear problems using ant colony programming, Appl. Math. Model. 48 (2017) 635–654.
[111] A. Hara, J.-i. Kushida, K. Fukuhara, T. Takahama, Cartesian Ant Programming with adaptive node replacements, in: 2014 IEEE 7th International Workshop on Computational Intelligence and Applications, IWCIA, IEEE, 2014, pp. 119–124.
[112] J.-i. Kushida, A. Hara, T. Takahama, Cartesian Ant Programming with node release mechanism, in: 2015 IEEE 8th International Workshop on Computational Intelligence and Applications, IWCIA, IEEE, 2015, pp. 83–88.
[113] S. Arslan, C. Öztürk, Artificial bee colony programming for feature selected cancer data classification, Int. J. Sci. Technol. Res. 4 (2018).
[114] S. Arslan, C. Ozturk, Feature selection for classification with artificial bee colony programming, in: Swarm Intelligence-Recent Advances, New Perspectives and Applications, IntechOpen, 2019, pp. 1–19.
[115] S. Arslan, C. Ozturk, A comparative study of automatic programming techniques, Informatica 43 (2019) http://dx.doi.org/10.31449/inf.v43i2.2133.
[116] F. Boudardara, B. Gorkemli, Application of artificial bee colony programming to two trails of the artificial ant problem, in: 2018 2nd International Symposium on Multidisciplinary Studies and Innovative Technologies, ISMSIT, 2018, pp. 1–6, http://dx.doi.org/10.1109/ISMSIT.2018.8567048.
[117] B. Görkemli, H. Citakoglu, T. Haktanir, D. Karaboga, A new method based on artificial bee colony programming for the regional standardized intensity–duration–frequency relationship, Arab. J. Geosci. 15 (3) (2022) 1–19.
[118] C. Bozoğullarından, C. Öztürk, Cloud-based artificial bee colony programming web application, in: 2019 Innovations in Intelligent Systems and Applications Conference, ASYU, IEEE, 2019, pp. 1–5.
[119] B. Gorkemli, D. Karaboga, A quick semantic artificial bee colony programming (qsABCP) for symbolic regression, Inform. Sci. 502 (2019) 346–362.
[120] B. Yassine, H. Hacene, Scaled artificial bee colony programming, in: 2018 International Conference on Applied Smart Systems, ICASS, IEEE, 2018, pp. 1–5.
[121] E.M. Golafshani, A. Ashour, A feasibility study of BBP for predicting shear capacity of FRP reinforced concrete beams without stirrups, Adv. Eng. Softw. 97 (2016) 29–39.
[122] E.M. Golafshani, A. Ashour, Prediction of self-compacting concrete elastic modulus using two symbolic regression techniques, Autom. Constr. 64 (2016) 7–19.
[123] E.M. Golafshani, S. Talatahari, Predicting the climbing rate of slip formwork systems using linear biogeography-based programming, Appl. Soft Comput. 70 (2018) 263–278.
[124] E.M. Golafshani, A. Behnood, Estimating the optimal mix design of silica fume concrete using biogeography-based programming, Cem. Concr. Compos. 96 (2019) 95–105.
[125] V. Stanovov, S. Akhmedova, E. Semenkin, The automatic design of parameter adaptation techniques for differential evolution with genetic programming, Knowl.-Based Syst. (2022) 108070.
[126] W.B. Chaabene, M.L. Nehdi, Genetic programming based symbolic regression for shear capacity prediction of SFRC beams, Constr. Build. Mater. 280 (2021) 122523.
[127] A.H. El-Bosraty, A.M. Ebid, A.L. Fayed, Estimation of the undrained shear strength of east Port-Said clay using the genetic programming, Ain Shams Eng. J. 11 (4) (2020) 961–969.
[128] F. Zhang, S. Nguyen, Y. Mei, M. Zhang, Genetic Programming for Production Scheduling, Springer, 2021.
[129] F. Zhang, Y. Mei, S. Nguyen, M. Zhang, Evolving scheduling heuristics via genetic programming with feature selection in dynamic flexible job-shop scheduling, IEEE Trans. Cybern. 51 (4) (2021) 1797–1811.
[130] S.B. Hamida, H. Hmida, A. Borgi, M. Rukoz, Adaptive sampling for active learning with genetic programming, Cogn. Syst. Res. 65 (2021) 23–39.
[131] J.-i. Kushida, A. Hara, T. Takahama, S. Nagura, Cartesian ant programming with transition rule considering internode distance, in: 2016 IEEE 9th International Workshop on Computational Intelligence and Applications, IWCIA, IEEE, 2016, pp. 101–105.
[132] J.L. Olmo, J. Raúl Romero, S. Ventura, Single and multi-objective ant programming for mining interesting rare association rules, Int. J. Hybrid Intell. Syst. 11 (3) (2014) 197–209.
[133] S. Luis, M.V. dos Santos, On the evolvability of a hybrid ant colony-cartesian genetic programming methodology, in: European Conference on Genetic Programming, Springer, 2013, pp. 109–120.
[134] J.-i. Kushida, A. Hara, T. Takahama, N. Mimura, Cartesian ant programming introducing symbiotic relationship between ants and aphids, in: 2017 IEEE 10th International Workshop on Computational Intelligence and Applications, IWCIA, IEEE, 2017, pp. 115–120.
[135] D. Li, Y. Chen, An efficient ant colony programming approach, in: 2018 IEEE SmartWorld, Ubiquitous Intelligence Computing, Advanced Trusted Computing, Scalable Computing Communications, Cloud Big Data Computing, Internet of People and Smart City Innovation (SmartWorld/SCALCOM/UIC/ATC/CBDCom/IOP/SCI), 2018, pp. 1438–1443, http://dx.doi.org/10.1109/SmartWorld.2018.00249.
[136] J.L. Olmo, A. Cano, J.R. Romero, S. Ventura, Binary and multiclass imbalanced classification using multi-objective ant programming, in: 2012 12th International Conference on Intelligent Systems Design and Applications, ISDA, IEEE, 2012, pp. 70–76.
[137] M.d.l. Cruz Echeandía, A.O.d.l. Puente, M. Alfonseca, Attribute grammar evolution, in: International Work-Conference on the Interplay Between Natural and Artificial Computation, Springer, 2005, pp. 182–191.
[138] A. Ortega, M. De La Cruz, M. Alfonseca, Christiansen grammar evolution: grammatical evolution with semantics, IEEE Trans. Evol. Comput. 11 (1) (2007) 77–90.
[139] M.H. Luerssen, D.M. Powers, Evolvability and redundancy in shared grammar evolution, in: 2007 IEEE Congress on Evolutionary Computation, IEEE, 2007, pp. 370–377.
[140] F.C.M. Rodrigues, J.B.C. Neto, C.A. Vidal, Split grammar evolution for procedural modeling of virtual buildings, in: 2015 XVII Symposium on Virtual and Augmented Reality, IEEE, 2015, pp. 75–83.
[141] A.O. de la Puente, R.S. Alfonso, M.A. Moreno, Automatic composition of music by means of grammatical evolution, in: Proceedings of the 2002 Conference on APL: Array Processing Languages: Lore, Problems, and Applications, 2002, pp. 148–155.
[142] M. Cebrian, M. Alfonseca, A. Ortega, Towards the validation of plagiarism detection tools by means of grammar evolution, IEEE Trans. Evol. Comput. 13 (3) (2009) 477–485.
[143] A.I. Diveev, S. Ibadulla, N. Konyrbaev, E.Y. Shmalko, Variational analytic programming for synthesis of optimal control for flying robot, IFAC-PapersOnLine 48 (19) (2015) 75–80.
[144] Z. Oplatková, I. Zelinka, Creating evolutionary algorithms by means of analytic programming-design of new cost function, in: ECMS 2007, European Council for Modelling and Simulation, 2007, pp. 271–276.
[145] Z. Oplatkova, I. Zelinka, R. Senkerik, Santa fe trail for artificial ant by means of analytic programming and evolutionary computation, Int. J. Simul. Syst. Sci. Technol. 9 (3) (2008) 20–33.
[146] I. Zelinka, D.D. Davendra, R. Šenkeřík, R. Jašek, Z. Oplatkova, Analytical programming-a novel approach for evolutionary synthesis of symbolic structures, in: Evolutionary Algorithms, InTech, 2011, pp. 149–176.
[147] R. Senkerik, Z. Oplatkova, I. Zelinka, D. Davendra, Synthesis of feedback controller for three selected chaotic systems by means of evolutionary techniques: Analytic programming, Math. Comput. Modelling 57 (1–2) (2013) 57–67.
[148] R. Senkerik, Z. Kominkova Oplatkova, I. Zelinka, B. Chramcov, D.D. Davendra, M. Pluhacek, Utilization of analytic programming for the evolutionary synthesis of the robust multi-chaotic controller for selected sets of discrete chaotic systems, Soft Comput. 18 (4) (2014) 651–668.

[149] L. Tomaszek, M. Chadli, On the particle swarm optimization control using analytic programming and self organizing migrating algorithm, in: 2019 IEEE Congress on Evolutionary Computation, CEC, IEEE, 2019, pp. 2886–2893.
[150] Z.K. Oplatkova, A. Viktorin, R. Senkerik, T. Urbanek, Different approaches for constant estimation in analytic programming, in: ECMS, 2017, pp. 326–332.
[151] R. Senkerik, A. Viktorin, M. Pluhacek, T. Kadavy, I. Zelinka, Differential evolution driven analytic programming for prediction, in: International Conference on Artificial Intelligence and Soft Computing, Springer, 2017, pp. 676–687.
[152] R. Senkerik, A. Viktorin, M. Pluhacek, T. Kadavy, I. Zelinka, Hybridization of analytic programming and differential evolution for time series prediction, in: International Conference on Hybrid Artificial Intelligence Systems, Springer, 2017, pp. 686–698.
[153] A. Diveev, E. Shmalko, Complete binary variational analytic programming for synthesis of control at dynamic constraints, in: ITM Web of Conferences, Vol. 10, EDP Sciences, 2017, p. 02004.
[154] N. Konyrbaev, A. Dauitbayeva, A. Ostayeva, A. Yessirkepova, A. Bexeitova, Variational analytic programming for synthesis of optimal control, Procedia Comput. Sci. 186 (2021) 743–750.
[155] A. Fonseca, P. Santos, G. Espada, S. Silva, Grammatical evolution mapping for semantically-constrained genetic programming, in: Genetic Programming Theory and Practice XVIII, Springer, 2022, pp. 45–62.
[156] M. O'Neill, C. Ryan, Grammatical evolution, IEEE Trans. Evol. Comput. 5 (4) (2001) 349–358.
[157] A. Fornells Herrera, E. Golobardes, E. Bernadó-Mansilla, J. Marti, Decision support system for breast cancer diagnosis by a meta-learning approach based on grammar evolution, in: ICEIS 2006 - 8th International Conference on Enterprise Information Systems, Proceedings, 2006, pp. 222–229.
[158] M.H. Luerssen, D.M. Powers, Graph design by graph grammar evolution, in: 2007 IEEE Congress on Evolutionary Computation, IEEE, 2007, pp. 386–393.
[159] D. Yang, Y. Ma, An air combat decision-making method based on knowledge and grammar evolution, in: Theory, Methodology, Tools and Applications for Modeling and Simulation of Complex Systems, Springer, 2016, pp. 508–518.
[160] K. Juste, H. Kita, E. Tanaka, J. Hasegawa, An evolutionary programming solution to the unit commitment problem, IEEE Trans. Power Syst. 14 (4) (1999) 1452–1459.
[161] S. Kumar, A. Sikander, Optimum mobile robot path planning using improved artificial bee colony algorithm and evolutionary programming, Arab. J. Sci. Eng. 47 (3) (2022) 3519–3539.
[162] M.A. Abido, A. Elazouni, Modified multi-objective evolutionary programming algorithm for solving project scheduling problems, Expert Syst. Appl. 183 (2021) 115338.
[163] W. Gao, Fast immunized evolutionary programming, in: Proceedings of 2004 International Conference on Machine Learning and Cybernetics (IEEE Cat. No. 04EX826), Vol. 1, IEEE, 2004, pp. 198–203.
[164] Y. Alipouri, J. Poshtan, Y. Alipouri, M.R. Alipour, Momentum coefficient for promoting accuracy and convergence speed of evolutionary programming, Appl. Soft Comput. 12 (6) (2012) 1765–1786.
[165] S. Wang, J. Ma, Q. He, An immune programming-based ranking function discovery approach for effective information retrieval, Expert Syst. Appl. 37 (8) (2010) 5863–5871.
[166] S. Xu, X. Xuesong, Improved immune programming algorithm and its application in the model clustering, in: 2016 Chinese Control and Decision Conference, CCDC, IEEE, 2016, pp. 3124–3129.
[167] M.H. Mansor, I. Musirin, M. Othman, Immune Log-Normal Evolutionary Programming (ILNEP) for solving economic dispatch problem with prohibited operating zones, in: 2017 4th International Conference on Industrial Engineering and Applications, ICIEA, IEEE, 2017, pp. 163–167.
[168] E. Mabrouk, A. Ayman, Y. Raslan, A.-R. Hedar, Immune system programming for medical image segmentation, J. Comput. Sci. 31 (2019) 111–125.
[169] Z. Zhang, X. Wang, J. Lu, Multi-objective immune genetic algorithm solving nonlinear interval-valued programming, Eng. Appl. Artif. Intell. 67 (2018) 235–245.
[170] S.S. Mustaffa, I. Musirin, M.M. Zamani, M. Othman, Pareto optimal approach in multi-objective chaotic mutation immune evolutionary programming (MOCMIEP) for optimal distributed generation photovoltaic (DGPV) integration in power system, Ain Shams Eng. J. 10 (4) (2019) 745–754.
[171] C. Chen, C. Luo, Z. Jiang, Elite bases regression: A real-time algorithm for symbolic regression, in: 2017 13th International Conference on Natural Computation, Fuzzy Systems and Knowledge Discovery (ICNC-FSKD), IEEE, 2017, pp. 529–535.
[172] Z. Jiang, C. Luo, A multi-space interrelation theory for correlating aerodynamic data from hypersonic ground testing, in: International Symposium on Shock Waves, Springer, 2017, pp. 503–510.
[173] M. Sarkar, B. Yegnanarayana, A clustering algorithm using evolutionary programming, in: Proceedings of International Conference on Neural Networks (ICNN'96), Vol. 2, IEEE, 1996, pp. 1162–1167.
[174] L. Hong, J.H. Drake, J.R. Woodward, E. Özcan, A hyper-heuristic approach to automated generation of mutation operators for evolutionary programming, Appl. Soft Comput. 62 (2018) 162–175.
[175] Z.-H. Sun, D. Liang, Z. Zhuang, L. Chen, X. Ming, Multi-task processing oriented production layout based on evolutionary programming mechanism, Appl. Soft Comput. 98 (2021) 106896.
[176] A. Lau, P. Musilek, Immune programming models of Cryptosporidium parvum inactivation by ozone and chlorine dioxide, Inform. Sci. 179 (10) (2009) 1469–1482.
[177] M. O'Neill, L. Spector, Automatic programming: The open issue? Genet. Program. Evol. Mach. 21 (1) (2020) 251–262.
[178] J.R. Koza, M.A. Keane, M.J. Streeter, W. Mydlowec, J. Yu, G. Lanza, Genetic Programming IV: Routine Human-Competitive Machine Intelligence, Vol. 5, Springer Science & Business Media, 2006.
[179] A.L. Samuel, AI, where it has been and where it is going, in: IJCAI, 1983, pp. 1152–1157.
[180] Z. Zojaji, M.M. Ebadzadeh, Semantic schema modeling for genetic programming using clustering of building blocks, Appl. Intell. 48 (6) (2018) 1442–1460.
[181] D.P. Searson, D.E. Leahy, M.J. Willis, GPTIPS: an open source genetic programming toolbox for multigene symbolic regression, in: Proceedings of the International Multiconference of Engineers and Computer Scientists, Vol. 1, Citeseer, 2010, pp. 77–80.
[182] M. Graff, E.S. Tellez, S. Miranda-Jiménez, H.J. Escalante, Evodag: A semantic genetic programming python library, in: 2016 IEEE International Autumn Meeting on Power, Electronics and Computing, ROPEC, IEEE, 2016, pp. 1–6.