BRNO UNIVERSITY OF TECHNOLOGY

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

FACULTY OF INFORMATION TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ

DEPARTMENT OF COMPUTER SYSTEMS

ÚSTAV POČÍTAČOVÝCH SYSTÉMŮ

ADVANCED GENETIC PROGRAMMING TECHNIQUES

POKROČILÉ TECHNIKY V GENETICKÉM PROGRAMOVÁNÍ

DOCTORAL THESIS TOPIC

POJEDNÁNÍ

AUTHOR — **Ing. Martin Hurta**

AUTOR PRÁCE

SUPERVISOR — **prof. Ing. Lukáš Sekanina, Ph.D.**

ŠKOLITEL

BRNO 2024

Chapter 1

Introduction

Over previous decades, natural computing (NC) has become a significant part of computer science. Thanks to its broad scope, NC can be split into three main sections: computing inspired by nature, synthesizing natural phenomena in computers, and computing with new natural materials [20]. While the latter two have also undergone essential advances, especially in quantum computing, this work is focused on the arguably most prominent section – computing inspired by nature.

The prominence of computing inspired by nature is given, in large part, by artificial neural networks (ANN) and the broader use of advanced variants and techniques such as deep neural networks (DNN) and convolutional neural networks (CNN). Lately, generative artificial intelligence and natural language processing have allowed impressive results in many domains [55, 72]. However, many application areas can still benefit from using less complex and more explainable algorithms inspired by nature [18].

One subgroup of such algorithms is evolutionary computing (EC), drawing inspiration from evolutionary biology. In EC, candidate solutions undergo a repeated evaluation, selection of better-performing solutions, and the creation of offspring through recombination and random mutation. After a sufficient number of generations, this process, inspired by Darwinism, leads to obtaining solutions adapted specially to the given task. A very specific category of EC is genetic programming (GP), popularised by Koza [52].

GP applies biology-inspired operators to transform candidate programs in the shape of trees into new ones with a better ability to perform a given task. GP was successfully applied to many problems, including symbolic regression, image and signal processing, financial trading, industrial process control, medicine, and bioinformatics, including many human competitive results [53]. Nevertheless, many alternative variants of GP have been presented in search of higher accuracy, faster evolution, better hardware implementations, or support for different problem domains. This includes a variant called cartesian genetic programming (CGP), which is the main subject of this thesis.

This thesis aims to summarize GP, with a focus on CGP and advanced techniques proposed since its origin, together with existing problems and exploration of applications in biomedical informatics and bioinformatics. Particularly, the objective is to identify open problems in this area and formulate a research hypothesis and goals for the Ph.D. thesis.

This document is organized as follows. Chapter 2 summarizes related work, notable versions, applications, and open problems of CGP. Chapter 3 includes the specification of the goals and central hypothesis of the work. Chapter 4 presents accomplished work on this topic. Chapter 5 outlines a time plan for the rest of the Ph.D. study and the work needed to finalize the Ph.D. thesis. Finally, Chapter 6 concludes the presented work.

Chapter 2

Related Work Summary

This chapter contains an overview of work related to the doctoral thesis topic. The chapter starts with an explanation of machine learning (ML) basics, including explainable artificial intelligence (AI) and multi-objective optimization. The chapter continues with a summary of genetic programming (GP), followed by a further focus on its variant called cartesian genetic programming (CGP). An overview of CGP is given, together with a listing of the most notable variants and modifications. The chapter further focuses on subtopics targeted by the thesis in the shape of scalability of CGP, use of CGP in classification tasks, and, lastly, application in biomedical informatics and bioinformatics.

2.1 Introduction to Selected Machine Learning Principles

Machine learning, a field of study of programs that can learn without being explicitly programmed, includes many different ML algorithms and target tasks. ML algorithms can be divided by various principles of their work. Algorithms performing *batch learning* learn only during the initial training phase; however, algorithms performing *online learning* learn incrementally and can adapt to new data. *Instance-based* algorithms compare the similarity of new cases to the one in training data. *Model-based* algorithms build and use a model to make predictions. The most prominent categorization is, however, based on the amount and type of supervision and splits algorithms into the following three groups [25]:

1. **Supervised Learning**
   In supervised learning, data used by the ML model for training include information about the desired solution, called labels. Typical tasks targeted by supervised learning are classification and regression.

2. **Unsupervised Learning**
   In unsupervised learning, training data are unlabeled, and the ML algorithm tries to find patterns in the data. Typical tasks are clustering, anomaly detection, or visualization.

3. **Reinforcement Learning**
   Reinforcement learning places a model called an agent into the environment. The agent then performs actions affecting the environment and receives rewards based on the environment state. The agent learns a strategy leading to the highest rewards, called a policy.

Applications targeted in this thesis (classification and regression) fall under the supervised learning category. During classification, the algorithm aims to assign each input vector to one of a finite number of discrete categories. The task becomes a regression if the desired output is not a discrete category but one or more continuous variables [7]. More precisely, in the regression task, we try to optimize the coefficient vector Θ of a linear function $f : \mathbb{R}^n \to \mathbb{R}$ to fit training data, with p independent variables, using a predefined model:

$$f = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + ... + \theta_p x_p. \tag{2.1}$$

Symbolic regression (performed by GP) generalizes regression by formulating the problem as follows:

$$y = f(X, \Theta), \tag{2.2}$$

where not only Θ but also f is unknown. Symbolic regression does not impose prior assumptions about the model structure but rather discovers the model and parameters to best fit the training data. This allows the creation of simple and accurate expressions that might not be possible with regular linear regression [61].

2.1.1 Explainable Artificial Intelligence

Explainable AI is a relatively new research field that aims to produce explainable models while maintaining high performance (i.e., accuracy). Produced models should enable users to understand and effectively manage AI, which is increasingly used in many fields. This is very important as AI's decisions can significantly impact individual beings, businesses, and society as a whole. [65]

Explainability can be achieved using less complex models such as logistic regression, linear regression, or decision trees. Some problems, however, require models with higher complexity, such as ANN, which generally cannot be understood and are referred to as „black-box“ solutions. [13]

Different post-modeling tools were proposed to obtain insight into decisions made by more complex models. One of the examples is Layer-wise relevance propagation (LRP) [4] for ANN. LRP utilizes backtracking to propagate the contribution of each neuron to the input layer and show the influence of different inputs on the final decision.

Nevertheless, explainable AI is still a new and open topic that is not yet fully defined, allows for different approaches and refers to overall „movement, initiatives, and efforts made in response to AI transparency and trust concerns, more than to a formal technical concept“ [1].

2.1.2 Multi-objective Optimization

Solving real-world problems often means operating with conflicting objectives (e.g., higher precision resulting in worse hardware parameters). Unlike standard optimization techniques with one optimal solution, multi-objective optimization solves a vector of decision variables and requires the introduction of the Pareto optimum concept. A vector $\bar{x}_n^* \in F$ representing candidate solution:

$$\bar{x}_n^* = [x_1, x_2, ..., x_n], \tag{2.3}$$

that optimizes k objective functions:

$$\bar{f}_k(\bar{x}_n) = [o_1(\bar{x}_n), o_2(\bar{x}_n), ..., o_k(\bar{x}_n)] \tag{2.4}$$

becomes Pareto optimal (in the task of minimization) if for every other $\bar{x}_n \in F$, it is true that

$$\bar{f}_k(\bar{x}_n^*) \leq \bar{f}_k(\bar{x}_n). \tag{2.5}$$

In other words, Pareto optimality gives a set of nondominated solutions: solutions where no other solution can improve some objectives without worsening others. Pareto optimal solutions form a Pareto front [16].

Multi-objective design can be solved by many approaches, including scalarization-based methods, metaheuristics, hybrid metaheuristics, and trust-based algorithms [16]. Popular methods include, for example, Weighted sum, which linearly combines multiple objective functions into one. ε-constraints solve the problem of multiple objective functions by selecting one primary function and transforming others into constraints. One of the most used methods is then NSGA-II [21], a second version of a nondominated sorting genetic algorithm aiming to preserve the diversity of the population of candidate solutions.

2.2 Genetic Programming

Genetic programming is a stochastic population-based algorithm that uses biology-inspired mutation, crossover, and selection operators to transform a population of candidate programs into new ones with, possibly, higher fitness, i.e., better ability to perform a given task. The most common variant of GP represents candidate programs in the shape of tree-based structures. Internal nodes contain functions taken from a function set. Leaf nodes contain either constants from a given set of constants or primary inputs. An example of a simple GP program can be seen in Fig. 2.1. [52, 53]

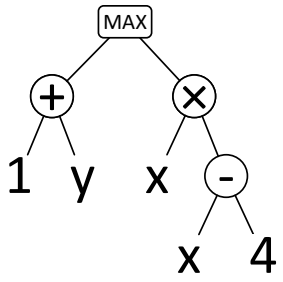

Figure 2.1: An example of GP tree representing mathematical function $result = max((1 + y), (x * (x - 4)))$.

Candidate programs are evolved throughout multiple generations. Each generation involves fitness evaluation of all candidate programs, selection of parent programs, with higher selection probability for programs with higher fitness, and creation of a new population by crossover of parent programs followed by random mutation. [52, 53]

Genetic programming and its variants were successfully applied to an impressive number of problem domains, including symbolic regression [12, 28]; image and signal processing [19, 34]; financial modeling [15, 63]; industrial process control [14, 51]; medicine, biology and bioinformatics [49, 71]; hyper-heuristics [5, 11]; art and computer games [3, 75, 94] [62, 70].

Despite many applications, standard tree-based representation lacks important abilities such as supporting multiple outputs and suffers from unexpected growth of programs called bloat [53, 84]. Therefore, many different improvements and variants of GP have been proposed since its origin to overcome these problems, achieve better results, and allow

application in new problem domains. These improvements covered different aspects of GP, including representation, crossover operators, and mutation operators. Among the most significant are stack-based GP [82], linear GP [6, 9], parallel and distributed [74], grammar-based representations [64, 96], encapsulated GP [58], or extended compact GP [10]. One of the variants that attracted many researchers is CGP [53, 84].

2.3 Cartesian Genetic Programming

Cartesian genetic programming was first proposed by Miller [68] based on a method for digital circuit evolution [67]. CGP is considered one of the most popular graph-based GP methods [45] and one of the most efficient methods for the evolutionary design and optimization of digital circuits [90].

CGP is a specific variant of GP that replaces traditional tree-based representation of GP programs by directed acyclic graphs with nodes ordered in the shape of the 2D grid, thus the term „cartesian“. Given CGP inspiration in biological evolution, candidate programs are encoded in fixed-size arrays of pointers called genotypes or chromosomes. Genotype is made out of individual computation nodes. Each node contains one pointer to a set of functions Γ and pointers to the data sources, either output of nodes in previous columns or program input. The fixed size of candidate programs gives CGP some advantages over GP: (1) the maximal size of a solution is fixed; (2) the bloat is not usually present [45]. The length of the genotype is given by:

$$Length = Row * Col * (Car + 1) + Out, \tag{2.6}$$

where Row is the number of rows, Col is the number of columns, Car is the cardinality (i.e., number of inputs) of individual nodes, and Out is the number of primary outputs.

A simple transformation from the genotype is made to obtain a cartesian grid representing the program called phenotype. An example of program encoding (i.e., genotype), computational graph (i.e., phenotype), and appropriate mathematical function is shown in Fig. 2.2. Similarly to GP, two programs with different phenotypes can result in semantically identical programs, i.e., $y = x_1 + x_2$ and $y = x_2 + x_1 + 1 - 1$.

Unlike GP, the representation of CGP allows programs to have more than one output and, thus, design more complex programs directly. The evolution process of CGP is also specific from general GP in several ways. A significant difference is that CGP does not utilize a crossover operator. Several studies were published on the utilization of crossover in CGP, but with mixed results and missing decisive proof of CGP benefiting from crossover across different tasks [70]. Offsprings are thus created only through mutation, conventionally one-point mutation. Another big difference is the search algorithm inspired by the $1 + \lambda$ evolutionary strategy. In every generation, one best-performing program is selected as a parent and continues to the next generation together with λ offspring created from the selected program using random mutation.

CGP is successfully used in Evolvable Hardware (EHW), where it allows the design of specialized electronics without the need for manual engineering. A key aspect of EHW is that it can create programs that cannot be fully specified, but their targeted behavior is known. Moreover, EHW can fully utilize properties of the target environment and data used for evolution and thus directly evolve circuits with better parameters, such as size, delay, or power consumption, without the need for further optimization [90].

CGP also shows its interesting properties in the growing field of approximate computing, where programs are designed with a certain amount of imprecision [90]. This is ideal for

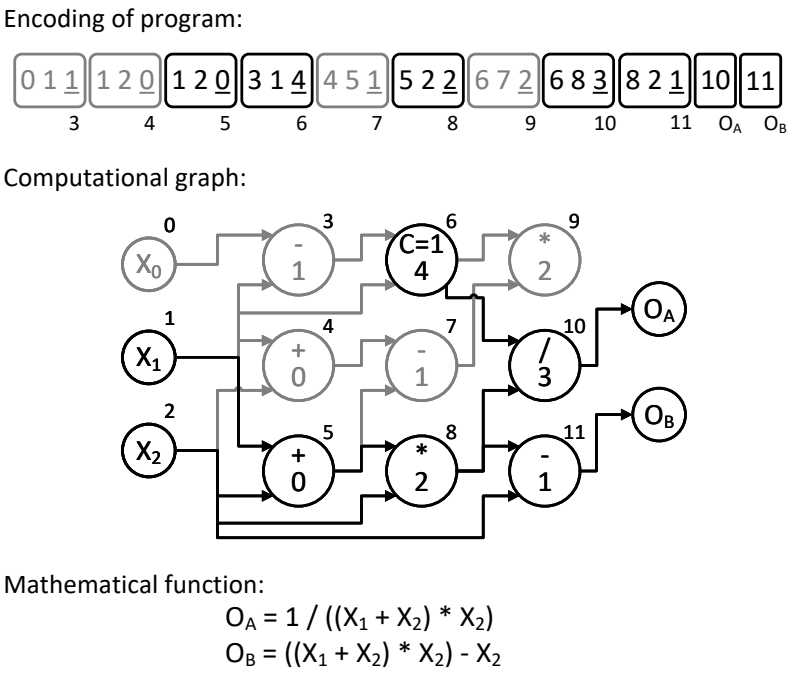

Figure 2.2: An example of CGP program encoding, corresponding computational graph, and mathematical functions represented by the program. Grayed-out parts represent inactive genes and nodes, i.e., parts not utilized by the CGP program. $\Gamma = \{+^{(0)}, -^{(1)}, *^{(2)}, /^{(3)}, 1^{(4)}\}$.

CGP, which creates increasingly more precise programs by the principle of its function. Partially working solutions are also building stones of multi-objective design, useful in many application areas in which suitable trade-offs between accuracy and properties such as power consumption, memory requirements, or speed have to be delivered [70].

Apart from the design and optimization of circuits [23, 89], CGP was successfully used in many different areas, including the design of image filters [31, 77], design of artificial neural networks [60, 83, 88], classification [54] or cryptography [73]. [62, 70]

2.3.1 Variants of CGP

Since its introduction, different variants and modifications of CGP have been introduced to improve different aspects of the algorithm and obtain better-performing solutions. Nevertheless, Kalkreuth [45] has concluded that the majority of topics of CGP (i.e., computational efficiency, genetic operators, parametrization, selection schemes, and diversity) are still in the point of only initial work and need future research. Furthermore, most modifications of CGP were evaluated only on problems of symbolic regression, Boolean functions, and circuit design, with classification being only a marginal area and evaluation on real-world data rare.

The following subchapter contains an overview of some of the most important variants and modifications of CGP done since its introduction, as shown by [62] and [70].

Modular/Embeded CGP

Walker and Miller first introduced Modular CGP [95] in 2004. In Modular CGP, parts of the genotype of candidate programs are extracted as modules and later reused in different programs or destroyed. Results have shown improvement in evolution time on several different problems. However, it should be noted that the size of genotypes of modular and basic variants in experiments differed, resulting in unfair comparison [70].

Kaufmann and Platzner [47] later proposed two new alternative methods for creating and destroying the modules: age-based and cone-based methods. The age-based method creates modules from nodes that did not change for a set number of generations and keeps modules with a higher average age of included nodes. The age-based method has shown to be superior over basic Modular CGP in evaluated tasks of circuit synthesis and classification. The second proposed method, the cone-based method, creates modules from nodes within specific cone structures and has shown to be superior only on circuit problems.

The effective form of modularity in CGP still remains an open question [70]. Miller [70] mentions that an interesting potential form of Modular CGP could utilize multiple chromosomes and use outputs of secondary chromosomes as input of primary chromosomes. Thus splitting the problem into multiple sub-problems.

Implicit-context CGP

Smith et al. [80] proposed an Implicit-context CGP (ICCGP). ICCGP tries to remove the positional dependence of genes presented in standard CGP by introducing a new, more complex alternative to nodes called enzymes. Each enzyme contains type, binding vectors, output vectors, and function genes. CGP-like phenotype is obtained by a self-assembly process, where individual enzymes bind to each other based on their parameters. ICCGP was successfully applied to the diagnosis of various health problems, including Alzheimer's disease, Parkinson's disease, and cancer [57, 79, 80, 81].

Real-valued CGP

Real-valued CGP (RVCGP) proposed by Clegg et al. [17] replaces integer-based genotypes with floating point numbers in the interval [0.0, 1.0]. This representation allows for smaller, more natural changes in genotype, as not every gene change leads to a new rounded integer value. Thus, changes can happen gradually, and changes in connection or function can happen over a period of several incremental mutations. Results have shown improvement in symbolic regression tasks in comparison with basic CGP. Later results [85] have shown, at best, comparable performance to basic CGP in tasks of digital circuit synthesis, function optimization, and wall avoidance (in simple robotic tasks).

Positional CGP

An alternative approach to the use of real values in CGP was later proposed by Wilson et al. [97] in the form of Positional CGP (PCGP). In PCGP, nodes no longer have a set position in the genotype. Instead, each node contains five real-valued genes: two for connection, one for a position, one for function, and one extra value that can either be the weight of inputs or an additional function parameter. Connections are created by multiplying connection genes with the node's position and snapping to nodes in the closest proximity. Experiments on nine benchmarks (classification, symbolic regression, and reinforcement learning) have

shown better results than RVCGP in classification and symbolic regression and worse in reinforcement learning.

Self-modifying CGP

The concept of Self-modifying CGP (SMCGP), introduced by Harding et al. [32], extends CGP by introducing a new type of function called self-modifying functions. These functions transform the genotype into a new phenotype that can again be used as a genotype. One genotype can thus represent several programs obtained by repeated runs of self-modifying functions. These runs can end after a set number of transformations or when no self-modifying function is left in phenotype. SMCGP was successfully applied to problems of parity circuits evolution, regression, and calculation of Fibonacci, square value, pi, and e.

Mixed-type CGP

Harding et al. [30] proposed a Mixed-type CGP (MTCGP), which allows data processing with multiple data types. Different data types are internally cast to either a double precision number or a vector of real numbers. Each node in MTCGP then inspects the data type of its inputs and chooses a corresponding function operator and output data type. In case of no possible function, the default output value is returned. MTCGP was applied to four problems from the UCI Machine Learning Repository dataset: Wisconsin breast cancer, phoneme, diabetes, and heart datasets. MTCGP achieved comparable and sometimes better results than standard ML techniques while resulting in small explainable solutions [30].

Recurrent CGP

Recurrent CGP (RCGP) proposed by Turner and Miller [87] extends standard CGP by allowing connections between nodes to be both feed-forward and feedback. The run of recurrent programs requires an initial setting of all node outputs to zero. Programs are run repeatedly with follow-up inputs until all inputs are applied or a set number of additional iterations is performed. An additional parameter of recurrent connection probability was proposed to control the share between feed-forward and feedback connections. RCGP achieved significantly better results than CGP on Artificial Ant and Sunspot benchmarks and on the problem of integer sequence prediction [86].

Iterative CGP

Iterative CGP (ICGP) proposed by Ryser-Welch et al. [76] introduces conditional loop genes. ICGP assumes the program to be linear (i.e., one row). Nodes then have four genes: function gene, first input from immediate previous node, second standard connection, and Boolean condition. If the second connection refers to one of the previous nodes except the immediate previous node, it creates a loop that is performed based on a Boolean condition. This allows ICGP to encode traditional algorithms. ICGP was applied to problems of traveling salesman, mimicry, and nurse scheduling. It has shown that it can successfully design human-readable problem-solving algorithms rather than mathematical functions as in standard CGP.

Differentiable CGP

Izzo et al. [43] proposed an innovative variant called Differentiable CGP (dCGP). In dCGP, all connections have weights, and node functions are represented as truncated Taylor expansions of a given order. These modifications allow the computation of the derivative of the CGP program up to a given order and thus utilize concepts such as backpropagation, known from ANN, to obtain highly tuned graphs. The dCGP was successfully evaluated on a suite of symbolic regression problems, solving ordinary differential equations and searching for prime integrals of sets of differential equations.

2.3.2 Other Modifications of CGP

Besides more extensive variants mentioned in Chapter 2.3.1, different modifications focused on individual parts of the CGP algorithm were also proposed and successfully evaluated since its inception. Thus, this chapter covers some of the important improvements in mutation, crossover, and search algorithms used in CGP.

Mutation Operator

Standard CGP uses either point mutation (i.e., a certain amount of genes is mutated) or probabilistic mutation (i.e., each gene is mutated with a set probability). These simple, non-informed kinds of mutations worsen the problem of the inefficiency of CGP. On a large collection of circuits, Vasicek [89] has shown that, in the problem of circuit optimization, approximately 180 candidate solutions must be generated to obtain a valid solution showing at least the same fitness as the parent. This is given by a big probability of mutating inactive genes (i.e., genes not utilized in the computation of any program output) and producing offspring with identical phenotypes. Even worse, mutation of an active gene can also produce an effectively identical phenotype by, for example, only changing the connection to the node with the same output value. At the same time, mutating inactive genes is important as it creates neutral drift and allows the algorithm to explore new solutions while stuck in local optima. Mutating only the active genes leads to a worse performance [70].

Goldman and Punch [26] proposed several new mutation operators in an attempt to improve the efficiency of mutation. The operator *Single* has led to the best results by mutating candidate solution until one active gene is changed. This preserves the genotype drift by potentially changing multiple inactive genes but removes the need to evaluate their fitness. Additionally, this operator removes the parameter of mutation probability from CGP.

Goldman and Punch [27] later proposed two methods (Reorder and DAG) targeting positional bias in CGP. Positional bias in CGP indicates a much larger probability of active nodes being close to the inputs than outputs. The best results were obtained with the Reorder method that, in each generation, shuffles nodes in the parent's genome. Shuffle is done in a way that keeps the semantics of the original phenotype and, at the same time, makes active nodes more likely to be in the middle between inputs and outputs.

Lastly, Kalkreuth [44] achieved improvement in three Boolean benchmarks and symbolic regression problems with a pair of new mutation operators called Insertion and Deletion. The Insertion selects an inactive node and changes different connections to make it active. Deletion selects the active node and changes its input connections in a way that makes it inactive.

Crossover Operator

The crossover operator is not used in standard CGP, and research on its application remains underdeveloped [70] and an open question [45].

Previously mentioned RVCGP [17] included simple crossover by combining gene values of two parents with random share from each of them.

The work mentioned earlier on modular CGP [47] investigated a crossover operator, where the cone of the chromosome was implanted into different chromosomes. Nevertheless, this solution performed better than regular Modular CGP only in two out of six benchmark problems.

Kalkreuth et al. [46] proposed a sub-graph crossover that behaves similarly to a single-point crossover. However, unlike single-point crossover, sub-graph crossover tries to minimize the distortion of chromosomes. It selects the crossover point as a point between two active nodes that are preserved as being active after the crossover is done. The proposed method achieved better results than the method without crossover on several benchmark problems, including circuit design, symbolic regression, and image filters. However, the method was not compared with basic CGP.

Husa and Kalkreuth [42] proposed a Block crossover for application in linear CGP. Blocks contain at least a given number of genes that are directly linked and are all active genes. Block crossover then selects one random block in two chromosomes and swaps them. However, their results have shown that if parameters of both basic CGP and CGP with crossover using methods are sufficiently fine-tuned, results tend to be similar.

Search Algorithm

Standard CGP uses evolutionary strategy $1+\lambda$, which is thus used in most existing research on CGP. Very little research was done on the use of other possible selection schemes and search algorithms [45].

Most works focus only on tuning the λ value in $1 + \lambda$. Some research was done on wider evolutionary strategy $\mu + \lambda$ and different hybrid algorithms. Milano et al. [66] compared $\mu + 1$ with $1 + \lambda$ and showed that $1 + \lambda$ achieves significantly better results. However, they also proposed a Parallel stochastic hill climber (PSHC) combing $\mu + 1$ with further optimization of parents by $1 + 1$. PSHC achieved significantly better results than the $1 + \lambda$ algorithm. Kaufmann and Platzner [48] compared hybrids of $1 + \lambda$ with multi-objective algorithms NSGAII and SPEA on multi-objective benchmarks and circuit design and showed that CGP-based hybrids achieved better results.

Yazdani and Shanbehzadeh [98] proposed a Balanced CGP (BCGP) incorporating features from biogeography-based optimization and mutation operator inspired by the concept of opposition-based learning. Experiments on symbolic regression have shown that BCGP outperforms CGP regarding accuracy and convergence speed.

Drahosova et al. [22] proposed a coevolutionary surrogate-model-like method that combines the traditional $1 + \lambda$ strategy with a second population of fitness predictors. Fitness predictors contain subsets of training data and are co-evolved by genetic algorithm to contain ideally broad and difficult subsets of data. Candidate programs are then, for the majority of the time, evaluated only on the currently best-performing subset instead of the whole training dataset. This decreased CPU time required to converge on various symbolic regression and image filter design problems.

2.4 Problem of CGP Scalability

Even though CGP is one of the most efficient methods for the evolutionary design of circuits, one of the major problems of CGP is scalability. The scalability problem in CGP can be understood in two domains: scalability of representation and scalability of fitness evaluation [90].

More complex problems can require large representation (chromosome size) with multiple inputs and outputs to capture a solution. Large search space then makes it challenging to find sufficient solutions as the mutation operator lacks any information about the problem domain [70], and many evaluations might be needed to obtain a solution of at least the same fitness as the current best-performing solution [90].

One of the ways of solving this problem is an initialization of CGP with an already working solution obtained by traditional methods and the use of CGP for further optimization of this solution. This is not possible in problems where the exact solution is not known. If design from scratch is required, it is recommended to use higher-level functions instead of gate-level design, thus simplifying the task. [90]

The problem of scaling of fitness evaluation emerges with the use of large datasets or design of circuits with a large number of inputs (tens to hundreds), where the requirement to run every candidate solution in each generation makes it infeasible to run CGP algorithm for a larger number of generations and thus creates suboptimal solutions.

Several papers were published on the acceleration of CGP evaluation. These include options like translating CGP solutions phenotypes to machine code for quick execution [93], use of virtual reconfigurable circuits (VRC) [91] or field programmable gate arrays (FPGA) [92] and utilization of graphics processing units (GPUs) [29]. However, even though there is a need for distributed and parallel architectures of CGP [62], research targeting acceleration through means of improving the CGP algorithm itself is limited.

CGP algorithm could thus greatly benefit from introducing a method that would allow easier design of more complex solutions and new research aimed at improving evaluation scalability through overall method improvement and not only acceleration of calculations.

2.5 Use of CGP for Hardware-aware Classification

CGP, thanks to its origin in the design of electronic circuits, allows for the direct design of small and energy-efficient solutions targeting specific environments and can thus achieve great hardware properties such as size, delay, or power consumption. This makes CGP ideal for designing programs for long-term application in wearable devices (useful for long-term assessment of health conditions), battery-powered devices, or overall edge computing and the Internet of Things (IoT). Moreover, multi-objective design utilizing CGP can create a Pareto set of different solutions with different trade-offs between accuracy and hardware parameters. These allow the selection of solutions ideal for specific use cases or switching between different models based on available energy or currently detected phase (i.e., simpler model for classification of movement to different categories, more complex model for specific important category). Big data is another field where efficient solutions are important, even on a personal computer or in server settings.

Nevertheless, existing research on the application of CGP on tasks of classification and prediction [8, 50, 54, 69] (and other mentioned in Chapter 2.6) focuses on the design of classifiers and predictors with the highest accuracy without further consideration of the hardware properties of designed programs. Good hardware properties of designed classifiers

are usually only a byproduct of using CGP. The lack of work on this topic is unfortunate as it is arguably much more suited for the CGP than basic classification, targeting the highest accuracy. The use of CGP for the design of hardware-aware classifiers is thus offering space for further research.

2.6 Use of CGP in Biomedical Informatics and Bioinformatics

The use of ML algorithms, especially artificial neural networks, has, in recent years, become an almost standard approach to classifying and predicting many health problems and diseases. However, the real-world use of these algorithms is still limited as the use of black box solutions that can not be fully understood is in certain fields (e.g., healthcare, defense) either not ideal or at all possible [78].

CGP was also successfully applied to detect and classify different health conditions, including Parkinson's disease [79], Alzheimer's disease [33], Levodopa-Induced Dyskinesia (LID), breast cancer [2, 79], thyroid cancer [57] or task of post docking filtering [24], with many of these applications utilizing Implicit context representation CGP (IRCGP). However, despite these successful applications, the overall use of CGP in biomedical informatics and bioinformatics is still limited.

Insufficient usage of CGP in these fields is especially saddening, given that CGP tends to design small and explainable solutions that may not only guarantee their functionality but often also provide new insights into solved tasks and thus broaden current understanding in fields of application [70]. Moreover, multi-objective design with CGP could be expanded to deliver good trade-offs between accuracy and explainability. This could be used to search for the most precise solutions that can still be analyzed and evaluated by a professional expert. A larger number of successful applications and software for easy use by biomedical informatics and bioinformatics experts could thus help with the popularization and spread of CGP use in these fields.

Chapter 3

Research Hypothesis and Objectives

Research on the GP, CGP, their different variants, and advanced techniques has highlighted many strong properties of CGP and successful applications, including the design of classifiers, predictors, and digital circuits. Yet, there are still existing problems and unexplored areas hindering the extended use of CGP. Based on the survey presented in Chapter 2 and existing knowledge gaps, the following research hypothesis can be formulated.

3.1 Hypothesis

> *Incorporating selected advanced techniques (such as co-evolution, modular design, surrogate models, approximation techniques, and multi-objective design approaches) into the standard CGP can improve its important properties (such as the number of fitness evaluations or time requirements of evolution). When applied to a particular problem, extended CGP can produce solutions with excellent properties (such as high classification accuracy, good trade-offs between accuracy and hardware requirements, and naturally explainable behavior) that can, in desired aspects, overcome state-of-the-art solutions, especially in selected applications of biomedical informatics and bioinformatics.*

3.2 Research Objectives

For the evaluation of the proposed hypothesis, the following research objectives must be accomplished.

3.2.1 Improving the Effectiveness of Fitness Evaluation

The scalability of fitness evaluation is still a problematic part of CGP, made more critical by the fact that most computational resources in the CGP algorithm are used for the fitness evaluation of candidate programs. Improving fitness evaluation effectiveness is thus a significant objective to allow a sufficient number of evaluations in a reasonable time and make use of extensive biomedical and bioinformatic datasets viable.

3.2.2 Improving CGP Performance on Complex Problems

A straightforward use of CGP in the design of complex circuits tends to struggle with finding minimal solutions and often any solution meeting the required criteria. One of the objectives is to adapt the CGP algorithm for easier search of complex solutions, for example, in use cases requiring multiple steps such as data filtering, preprocessing, feature extraction, and classification.

3.2.3 Adaptation of the CGP for the Design of Hardware-aware Classifiers and Predictors

Common works describing the use of CGP for the design of classifiers and predictors are focused only on achieving the highest accuracy or other corresponding functionality metrics. Yet, applying evolved solutions to big data or real-time classification in wearable and edge devices requires solutions with good trade-offs between accuracy and hardware parameters (such as delay and power consumption). Adaptation of the CGP algorithm for the multi-objective design of classifiers and predictors has to be explored further and is one of the thesis objectives.

3.2.4 Application of the CGP to New Problems in the Biomedical Informatics and Bioinformatics on Real-world Data

Potentially explainable solutions generated by CGP make it a promising candidate for use in situations where black-box solutions are not feasible. Yet, its use in biomedical informatics and bioinformatics is still limited and often does not leverage CGP's ability to design efficient computer programs or explainable solutions. The objective is to apply CGP to different problems in biomedical informatics and bioinformatics and explore its capabilities in these fields. Details about selected problems are provided in Chapters 4 and 5.

3.2.5 Comparison of Proposed Methods with Basic CGP and Common Machine Learning Algorithms

Existing variants and modifications of CGP often lack fair comparison with other commonly used ML algorithms and sometimes even basic CGP. The lack of fair comparison then discourages other researchers from using CGP and hinders further development of CGP itself. A fair comparison of proposed methods with basic CGP and different ML algorithms on multiple datasets is thus an essential objective to prove the usefulness of the proposed methods.

3.3 Methodology

The methodology described below will be followed during work on the Ph.D. thesis in order to achieve the presented research objectives and consequently evaluate the research hypothesis.

3.3.1 Innovative Incorporation of Selected Extensions

Standard CGP will be used as a base for proposed methods incorporating various extensions and advanced techniques. To target research objectives, employing the following modifications is proposed:

- **Faster Fitness Evaluation**
  To improve the efficiency of fitness evaluation, co-evolution with Adaptive size fitness predictors (ASFP) [22] will be employed. The use of ASFP in the classification task is yet to be fully explored, but could potentially significantly reduce the CPU time required for fitness evaluation of candidate solutions and help with designing better quality solutions in a shorter time.

- **Modular Design**
  The use of modular design is proposed to help CGP with solving complex problems. Unlike conventional modular CGP described in Chapter 2.3.1, the planned approach aims to split solved problems into multiple simpler sub-problems. Solutions to these sub-problems will then be simultaneously co-designed by multiple populations of CGP (one population for each sub-problem).

- **Multi-objective Design**
  The design of hardware-aware classifiers and predictors is aimed to be solved by incorporating some of the existing approaches for multi-objective design (e.g., NSGA-II, ε-constraints). The subsequent problem to be solved is the calculation of hardware parameters of candidate solutions. Simple parameters such as the number of active genes can be imprecise, and synthesizing each candidate solution is too computationally expensive. A suitable method will be selected based on experimental results.

- **Supporting Explainability**
  Achieving explainable solutions requires the CGP program to be reasonably complex and analyzable by experts. To ensure the explainability of designed programs, limits on the maximal size of grids and multi-objective design techniques (e.g., trade-offs between accuracy and the number of nodes) will be employed.

3.3.2 Efficient Implementation

Efficient implementation of CGP and incorporated extensions is essential. During implementation, emphasis will be given to parallelizing fitness evaluation and running of populations. Vectorization will be utilized in the calculation of candidate program fitness. Implementation will be done using C++ and Python programming languages. C++ stands as an ideal candidate for the efficient implementation of proposed methods. Python was selected for its wider usage outside of computer science, as it allows simpler use by other researchers. Techniques such as vectorization and full utilization of the Numpy[1] library will be used to compensate for worse performance in comparison with C++ language.

3.3.3 Detailed Statistical Evaluation on Target Applications and Comparison with State of the Art

As CGP is a stochastic algorithm, each run can achieve a different final program. Fair statistical evaluation of proposed methods will thus be based on multiple independent runs of each proposed method and its settings. Proposed methods will be compared with existing solutions from literature, standard CGP, and state-of-the-art ML algorithms to ensure their quality. Comparisons will be made on multiple datasets, primarily based on real-world data. Where it is possible, publicly available datasets will be used to allow for easier comparison and replication of the results.

[1] https://numpy.org/

Chapter 4

Accomplished Work

Some of the approaches described in Chapter 3 have already been adopted, implemented, and evaluated. This chapter contains a list of accepted and published works related to the dissertation topic. Publications are described using text, figures, and tables originating from given publications.

Accepted and Published Work

[36] Hurta, M., Drahosova, M., Sekanina, L., Smith, L. S. and Alty, E. J. Evolutionary Design of Reduced Precision Levodopa-Induced Dyskinesia Classifiers. In: Medvet, E., Pappa, G. and Xue, B., ed. *Genetic Programming, 25th European Conference, EuroGP 2022*. Springer Nature Switzerland AG, 2022, vol. 13223, p. 85–101. Lecture Notes in Computer Science. DOI: 10.1007/978-3-031-02056-8_6. ISBN 978-3-031-02055-1. Available at: https://link.springer.com/chapter/10.1007/978-3-031-02056-8_6.
CONTRIBUTION: 50%

[35] Hurta, M., Drahosova, M. and Mrazek, V. Evolutionary Design of Reduced Precision Preprocessor for Levodopa-Induced Dyskinesia Classifier. In: Rudolph, G., Kononova, A. V., Aguirre, H., Kerschke, P., Ochoa, G. et al., ed. *Parallel Problem Solving from Nature - PPSN XVII*. Springer Nature Switzerland AG, 2022, vol. 13398, p. 491–504. Lecture Notes in Computer Science. DOI: 10.1007/978-3-031-14714-2_34. ISBN 978-3-031-14713-5. Available at: https://link.springer.com/chapter/10.1007/978-3-031-14714-2_34.
CONTRIBUTION: 50%

[39] Hurta, M., Mrazek, V., Drahosova, M. and Sekanina, L. Multi-objective Design of Hardware Accelerators for Levodopa-Induced Dyskinesia Classifiers. In: *Evo* 2023 – Late-Breaking Abstracts Volume*. Available at: https://www.fit.vut.cz/research/publication/12973/.
CONTRIBUTION: 50%

[37] Hurta, M., Mrazek, V., Drahosova, M. and Sekanina, L. ADEE-LID: Automated Design of Energy-Efficient Hardware Accelerators for Levodopa-Induced Dyskinesia Classifiers. In: *2023 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. Institute of Electrical and Electronics Engineers, 2023, p. 1–2. DOI: 10.23919/DATE56975-.2023.10137079. ISBN 978-3-9819263-7-8. Available at: https://ieeexplore.ieee.org/

document/10137079.
CONTRIBUTION: 50%

[38] Hurta, M., Mrazek, V., Drahosova, M. and Sekanina, L. MODEE-LID: Multiobjective Design of Energy-Efficient Hardware Accelerators for Levodopa-Induced Dyskinesia Classifiers. In: Jenihhin, M., Kubátová, H., Metens, N., Raik, J., Ahmed, F. et al., ed. *2023 26th International Symposium on Design and Diagnostics of Electronic Circuits and Systems (DDECS)*. Institute of Electrical and Electronics Engineers, 2023, p. 155–160. DOI: 10.1109/DDECS57882.2023.10139399. ISBN 979-8-3503-3277-3. Available at: https://ieeexplore.ieee.org/document/10139399.
CONTRIBUTION: 50%

[41] Hurta, M., Schwarzerova, J., Provaznik, V., Weckwerth, W., Walther, D. et al. Utilizing Cartesian Genetic Programming for Efficient Polygenic Risk Score Calculation in Plants. In: *Program and Abstract Book: Swedish Bioinformatics Workshop 2023*. 2023. 49 p. Available at: https://drive.google.com/file/d/1YQ2dbigYqRdsG_c_1Xz6rKw9Qz5TC200/.
CONTRIBUTION: 50%

[40] Hurta, M., Schwarzerova, J., Nagele, T., Weckwerth, W., Provaznik, V. et al. Utilizing Genetic Programming to Enhance Polygenic Risk Score Calculation. In: *2023 IEEE International Conference on Bioinformatics and Biomedicine (BIBM 2023)*. Institute of Electrical and Electronics Engineers, 2023, p. 3782–3787. DOI: 10.1109/BIBM58861.2023.10385615. ISBN 979-8-3503-3748-8. Available at: https://ieeexplore.ieee.org/document/10385615.
CONTRIBUTION: 34%

4.1 Evolutionary Design of Reduced Precision Levodopa-induced Dyskinesia Classifiers [36]

Parkinson's disease is one of the most common neurological conditions whose symptoms are usually treated with a drug called levodopa. To minimize levodopa side effects, i.e., levodopa-induced dyskinesia (LID), it is necessary to manage levodopa dosage correctly. As LID can fluctuate in severity throughout the day, it is difficult to classify it during a short period of a physician's visit. A low-power wearable classifier enabling long-term and continuous LID classification would thus significantly help with LID detection and dosage adjustment.

This paper covers the application of CGP for the automatic design of a low-power LID classifier operating on a time series collected using accelerators attached to the patient's body. An overview of the classifier model is shown in Fig. 4.1. The paper describes three main experiments:

- E1 - Comparison of automated LID classifier design using basic CGP and proposed method combining CGP with the coevolution of adaptive size fitness predictors (CGPcoASFP) [22] to reduce the computation cost of the evolution process.
- E2 - Investigation of utilizing reduced precision to decrease the power consumption of evolved LID classifier while maintaining precision comparable to a baseline software implementation. Five data representations are compared, including baseline 32-bit

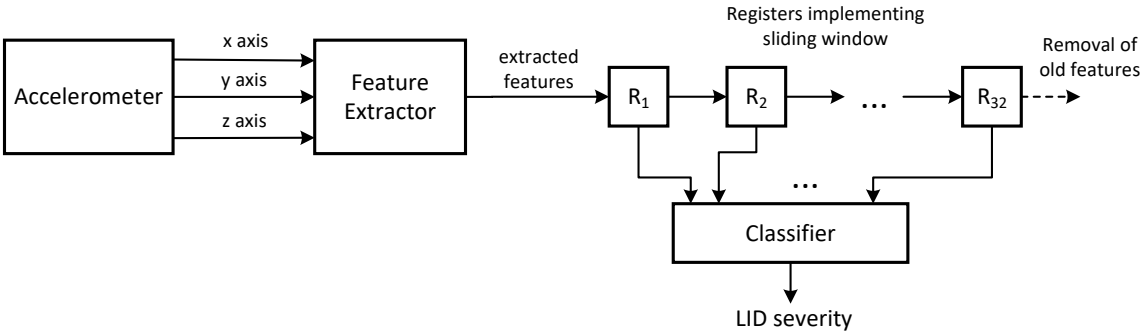

Figure 4.1: Overview of proposed LID-classifier model. Acceleration data in three axes are obtained every 0.01 seconds and undergo feature extraction by magnitude calculation. Data saved in 32 registers (implementing sliding window) are shifted to the right, and the newly extracted feature is stored in register R_1. The classifier calculates the response for every new state of the sliding window. LID severity of recording is determined by averaging the response of the classifier and thresholding.

floating point (FP), fixed-point 8-bit representation in both signed (FX-8s) and unsigned (FX-8u) variants, and fixed-point 16-bit representation in both signed (FX-16s) and unsigned (FX-16u) variants.

- E3 - Synthesis of three selected solutions using a standard circuit design flow and comparing their hardware characteristics, i.e., area on the chip, delay, power consumption, and power delay product (PDP).

Experiments show that CGPcoASFP can design classifiers that can determine the presence and the severity of LID approximately nine times faster than basic CGP. CGPcoASFP outperforms CGP in terms of computational cost in all investigated experiments and effectively prevents overfitting when evolving LID-classifiers with FX-8u data representation.

The design of a LID classifier working with reduced precision was investigated to find a hardware-efficient classifier with a classification accuracy as close as possible to a baseline software implementation. It was observed that classifiers working with FX-8u, together with input data preprocessing using the logical right shift by five bits (SR5), achieved better classifier accuracy (AUC) than classifiers using both 16 bits data representation and even baseline software implementation using FP data representation, as shown by results obtained on individual test groups[1] in Table 4.1.

Moreover, selected FX-8u LID classifier C1 significantly outperformed the hardware characteristics of other investigated solutions C2 and C3 obtained using FX-16u data representation, as shown in Table 4.2.

[1]Collected test data were split into test groups, according to the severity of LID graded by the standard UDysRS (Unified Dyskinesia Rating Scale) scoring system from 0 (no dyskinesia) to 4 (severe dyskinesia). Additional test groups *sitting-at-rest* and *walking* were created from data recorded while performing given activities.

		LID 1 [AUC]	LID 2 [AUC]	LID 3 [AUC]	LID 4 [AUC]	*Sitting-at-rest* [AUC]	*Walking* [AUC]
CGP	FP	0.55	0.69	0.85	0.93	0.92	0.75
CGPcoASFP	FP	**0.56**	0.69	0.85	0.93	0.92	0.76
CGP	FX-8u(SR-5)	**0.56**	**0.71**	**0.89**	0.95	**0.95**	0.84
CGPcoASFP	FX-8u(SR-5)	**0.56**	**0.71**	**0.89**	**0.96**	**0.95**	**0.85**
Lones et al. [56]		**0.56**	0.69	0.85	0.93	0.92	0.73

Table 4.1: Mean AUC presented in [56] and mean AUC out of 100 runs for standard CGP with FP, CGPcoASFP with FP, standard CGP with FX-8u(SR-5) and CGPcoASFP with FX-8u(SR-5). For each test group, the best result is marked in bold font.

LID-Classifier	Area on chip [μm^2]	Delay [ns]	Power [mW]	PDP [pJ]
C1: FX-8u	258.58	0.94	0.17	0.16
C2: FX-16u	901.53	3.36	0.36	1.20
C2: FX-32u	1936.80	6.63	0.80	5.28
C3: FX-16u	6417.68	40.54	2.78	112.77
C3: FX-32u	27156.98	179.69	10.32	1854.11

Table 4.2: Hardware characteristics of synthesized LID-classifiers for a 45nm technology.

4.2 Evolutionary Design of Reduced Precision Preprocessor for Levodopa-induced Dyskinesia Classifier [35]

This work aims to design a hardware-efficient implementation of data preprocessing in the task of LID classification. Previously proposed HW-efficient LID-classifier still employed data preprocessing in the form of magnitude calculation, which is considerably complex to implement directly in HW as the square root calculation is complex and power-inefficient, and the use of a lookup table would be for the 12-bit input enormous. Therefore, this paper aims to replace this preprocessor with HW-effective implementation while keeping a reasonable accuracy of the classifier and thus to produce a solution that could be implemented directly into a home wearable device.

Three scenarios, shown in Fig. 4.2, for the automated design of an energy-efficient variant of data preprocessing for the LID classifier, were proposed and compared:

- S1) Evolution of magnitude approximation using basic CGP.
- S2) Design of a preprocessing unit using two-population coevolution (2P-CoEA) considering CGP with the coevolution of adaptive size fitness predictors (CGPcoASFP). 2P-CoEA is performed with a fixed classifier presented in [36].
- S3) A design using three-population coevolution (3P-CoEA) combining the compositional coevolution of the preprocessor and classifier with the coevolution of fitness predictors.

It was shown that all three investigated scenarios, including compositional coevolution of preprocessor and classifier, can produce energy-saving solutions suitable for implementation in hardware units, with AUC comparable to the baseline software implementation.

Evolutionary design of magnitude approximation using CGP (scenario S1) led to larger and more energy-demanding solutions. The evolution of the preprocessor for the baseline classifier implementation (scenario S2) has resulted in a solution with significantly better

AUC for the „walking“ test group compared with other investigated approaches. Compositional coevolution of preprocessor and classifier supplemented with coevolution of adaptive size fitness predictors (scenario S3) has resulted in the smallest solutions on average. All of the proposed solutions significantly improved investigated hardware characteristics compared to the baseline solution, with the best results achieved by newly proposed scenarios S2 and S3, with energy consumption up to 29 times lower than in the existing solution. HW parameters of selected solutions from all scenarios are presented in Table 4.3.

LID-Classifier	Area on chip [μm^2]	Number of cycles	Power [mW]	Energy [pJ]
P0+C0 (baseline solution)	7115	13	1.0459	135.967
P1+C0 (S1: magnitude approx.)	5014	1	0.8593	8.593
P2+C0 (S2: 2P-CoEA)	2435	1	0.4645	4.645
P3+C3 (S3: 3P-CoEA)	2629	1	0.5131	5.131

Table 4.3: HW characteristics of synthesized solutions for a 45 nm technology on 100 MHz frequency.

4.3 Multi-objective Design of Hardware Accelerators for Levodopa-induced Dyskinesia Classifiers [39]

This late-breaking abstract presents ideas for improvements of the LID classifier model and adaptation of the method for automated classifier design to support the multi-objective design (with a trade-off between accuracy and energy consumption). Presented ideas were fully described and evaluated as part of subsequent publications [37] and [38].

4.4 ADEE-LID: Automated Design of Energy-efficient Hardware Accelerators for Levodopa-induced Dyskinesia Classifiers [37]

This paper focuses on an improved method for the automated design of energy-efficient hardware accelerators for LID classifiers, consisting of a feature extractor (FE) and a classifier co-designed using genetic programming, as shown in Fig. 4.3.

Several improvements over the state-of-the-art method (presented in [35]) were proposed and experimentally evaluated:

- Existing state-of-the-art solutions do not consider the sub-byte operations successfully used in machine learning accelerators. The paper thus proposes to include variable bit widths of both FE and classifier in their genotype.

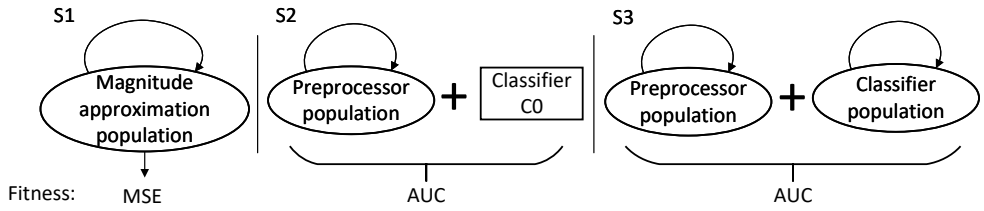

Figure 4.2: Three scenarios for automated design of hardware-efficient implementation of data preprocessing in the task of LID classification.

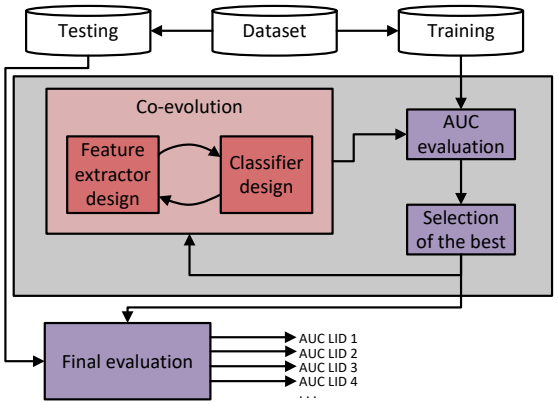

Figure 4.3: Overview of the methodology for the design of classifiers. FE and classifier are designed simultaneously, using CGP, by switching the currently-evolved population in each epoch. Populations interact through the evaluation phase, where candidate solutions of one population (e.g., the classifiers) are evaluated in connection with the currently best candidate solution from the other population (e.g., the FE). The fitness of candidate solutions is given as the classifier accuracy (AUC) of the composition of the FE and classifier on training data. The final combination of FE and classifier is evaluated on separate test data.

- Extracted features are stored in 32 registers representing the sliding window used by the classifier. However, evolved classifiers do not often utilize all sliding window samples. Thus, eliminating unnecessary circuits could reduce energy consumption.
- The hardware complexity of evolved circuits in the existing method was estimated using the number of arithmetic operations. Adopting a standard synthesis procedure would provide more precise results that could lead to better classifiers.

Experimental results have shown that a combination of proposed method improvements leads to comparable or better AUC across all test groups except for test group LID1, as shown in Table 4.4. A comparison of HW parameters with the existing work shows a significant decrease in energy consumption, justifying the need to employ proper hardware characteristics during the evolution.

Methodology	FE/Clas. bit widths	Window size	Energy [pJ]	AUC						
				LID1	LID2	LID3	LID34	LID4	Sitting	Walking
This [37]	9/11	28	0.228	0.553	0.737	**0.921**	**0.962**	**0.983**	**0.968**	0.906
This	9/8	24	0.226	0.557	0.740	0.916	0.958	0.979	0.959	0.901
This	8/11	26	0.218	0.550	0.739	0.913	0.956	0.978	0.957	0.905
This	9/12	26	0.192	0.552	0.741	0.916	0.956	0.975	0.959	0.912
This	9/12	22	0.167	0.555	**0.746**	0.912	0.955	0.976	0.957	**0.914**
This	10/9	31	0.146	0.552	**0.747**	0.916	0.953	0.971	0.957	0.868
This	6/7	32	0.093	0.527	0.708	0.889	0.946	0.974	0.951	**0.914**
Lones [56]	floats	—	—	**0.56**	0.69	0.85	—	0.93	0.92	0.73
Hurta [35]	8/8	32	0.859	0.55	0.73	0.89	—	0.96	0.95	0.82
Hurta [35]	8/8	32	0.465	**0.56**	0.73	0.89	—	0.97	0.95	0.87
Hurta [35]	8/8	32	0.513	0.55	0.73	0.90	—	0.97	0.95	0.83

Table 4.4: Energy and AUC for various test groups of selected Pareto-optimal solutions.

4.5 MODEE-LID: Multiobjective Design of Energy-efficient Hardware Accelerators for Levodopa-induced Dyskinesia Classifiers [38]

This paper deals with a method for the multi-objective design of energy-efficient hardware accelerators for LID classifiers, including a new approach for efficient calculation of power consumption of candidate solutions.

Designing an energy-efficient LID classifier is a multi-objective task where we wish to maximize AUC and minimize energy consumption. Nevertheless, existing works use only a single-objective optimization of AUC. This can be solved by employing a multi-objective optimization, in this case, by transforming the multi-objective design into a single-objective one by introducing constraints. This problem was solved by modifying the calculation of the fitness f of the candidate solution (the composition of feature extractor FE and classifier C) into the equation:

$$f(FE,C) = \begin{cases} AUC(FE,C), & \text{if } E(FE,C) < \varepsilon \\ -E(FE,C), & \text{otherwise} \end{cases}, \tag{4.1}$$

where $E()$ is the energy consumption of a solution, $AUC()$ is the classification accuracy in terms of AUC, and ε is the set energy constraint. The fitness is equal to classification accuracy if energy consumption is sufficiently low. Candidate solutions with energy consumption higher than ε have their fitness equal to the negative value of the energy consumption. Solutions obtained from several runs with different constraints are combined, thus covering a wide range of different energy consumption values.

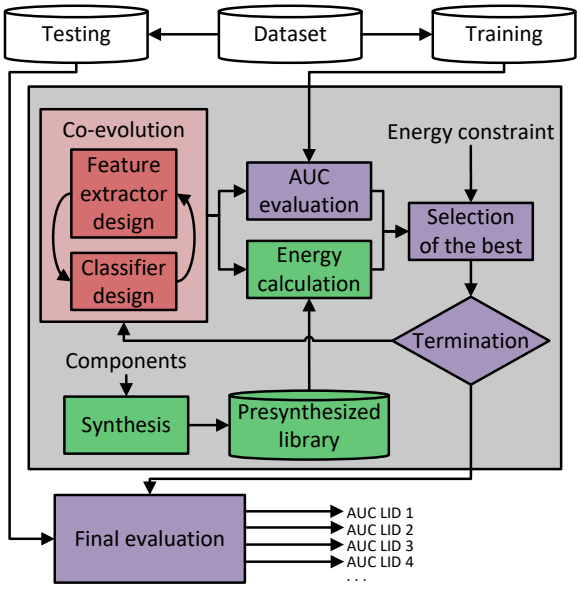

Figure 4.4: Overview of the proposed methodology for multi-objective co-evolutionary design of energy-efficient LID classifier. In addition to the method shown in Fig. 4.3, MODEE-LID uses the energy consumption of candidate solutions as part of the fitness evaluation. To overcome the computationally expensive synthesis of candidate solutions, energy consumption is calculated from the library of pre-synthesized components created before the run of the automated design method.

It was proposed to use pre-synthesized components to obtain the energy consumption value needed for the multi-objective design without the need for a computationally expensive synthesis of each candidate solution. Hence, each combination of 18 allowed functions and 10 possible values of bit widths was designed and synthesized, thus creating a look-up table of 180 different possible values of energy consumption. As the LID classifier comprises up to 32 registers, a similar table is also created for ten different bit widths of registers. With all variants of nodes and registers pre-synthesized, the energy consumption of the final solution is then calculated as the sum of the energy consumption of the feature extractor, the aggregation part (the approximate mean), registers, and the classifier, with respect to their active parts and bit widths.

The proposed approach for the automated design of an energy-efficient LID classifier is shown in Fig. 4.4. The feature extractor is co-designed simultaneously with the classifier using a CGPcoASFP.

Proposed improvements allowed the design of a wide range of high-quality solutions and significantly reduced their energy consumption compared to the state-of-the-art method. A decrease in AUC on test group LID34 of only 0.002 allowed the method to reduce energy consumption by 49 %. Further, an AUC decrease of 0.004 allowed for energy reduction by 57 %. Fig. 4.5 shows properties of the solutions obtained by merging seven different energy constraint results and the same number of runs without a set energy constraint. The Pareto front achieved by combined energy constraints dominates most space, while the variant without energy constraints is better only in finding a solution with the highest AUC.

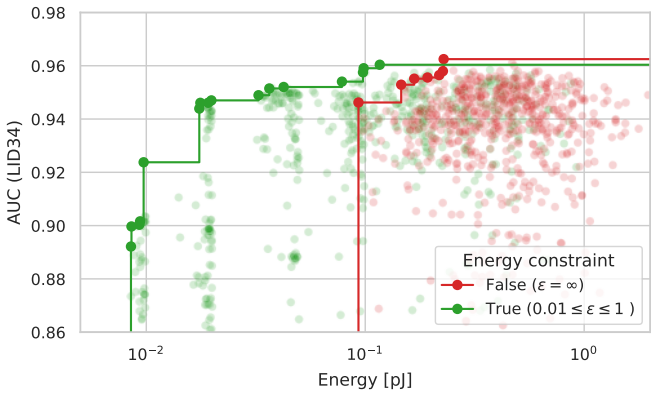

Figure 4.5: Trade-off between energy consumption and AUC on test group LID34. Green points represent solutions obtained by a union of 100 runs for each of the seven selected energy constraints. Red represents 700 runs of the method without energy constraint.

4.6 Utilizing Cartesian Genetic Programming for Efficient Polygenic Risk Score Calculation in Plants [41]

This abstract presents the idea of utilizing polygenic risk scores (PRS) calculation for plant genetics research to help quantify the probability of certain outcomes, such as crop disease susceptibility or yield potential. Further, the idea of enhancing PRS calculation by applying CGP to improve the method's performance is presented. The complete method and results were later published as part of [40].

4.7 Utilizing Genetic Programming to Enhance Polygenic Risk Score Calculation [40]

The polygenic risk score (PRS) has proven to be a valuable tool for assessing an individual's genetic predisposition to phenotype (disease) within biomedicine in recent years [59]. However, traditional regression-based methods for PRS calculation have limitations that can impede their accuracy and predictive power. This study introduces an innovative approach to overcome the limitations of traditional regression techniques by utilizing CGP and improving the accuracy of PRS predictions.

The paper proposes a new CGP-based PRS calculation (cgpPRS), modifying conventional PRS yet preserving the basic principles of this tested method. The proposed method is compared with conventional PRS (cPRS) using the Ordinary Least Squares (OLS) model as a classical multivariable linear regression. Both methods were compared using 5-fold cross-validation on three real-world plant datasets: A. thaliana, Maize, and Barley. Groups of healthy and unhealthy individuals were divided for each dataset based on yield or height higher or lower than the mean value. The methodology overview can be seen in Fig. 4.7.

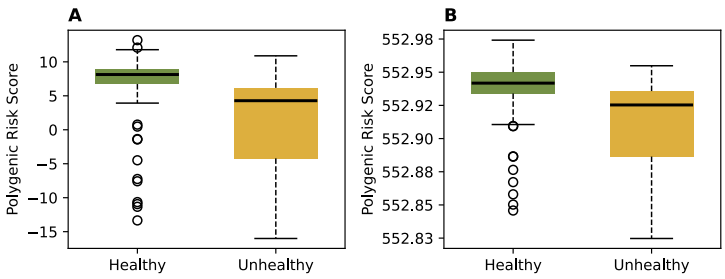

Figure 4.6: Comparison of cPRS (A) and cgpPRS (B) for Arabidopsis thaliana. The green color presents healthy individuals (growth rate >= mean growth rate), and the yellow color shows unhealthy individuals (growth rate < mean growth rate).

Proposed cgpPRS achieved comparable or better results in all three experiments. Fig. 4.6 shows results on the plant Arabidopsis thaliana, where both methods achieved the most significant separation. Moreover, the proposed method cgpPRS could separate healthy and unhealthy samples even for dataset Barley, where the conventional method failed, as confirmed by the Mann-Whitney U test shown in Table 4.5.

Experiment	Mann-Whitney U Test		Student's t-test	
	cPRS p-value	cgpPRS p-value	cPRS p-value	cgpPRS p-value
I (Ath)	*8.17e-20	*4.87e-15	*3.07e-11	*2.52e-11
II (Barley)	0.784	*0.29e-07	*0.001	*3.62e-07
III (Maize)	*1.15e-07	*1.15e-07	*8.62e-07	*4.74e-06

Table 4.5: Summary of the statistical evaluation for significant difference between calculated PRS for healthy and unhealthy individuals (* significance level: alpha < 0.05).

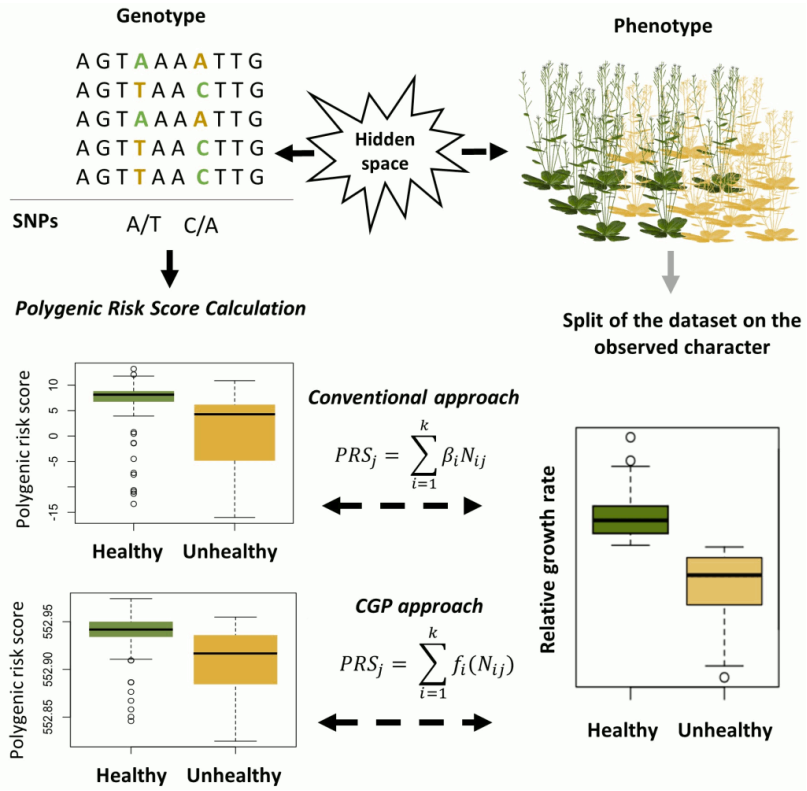

Figure 4.7: Representation of the methodology and results of the presented work. Samples in datasets are separated into two groups of healthy and unhealthy individuals based on yield (height for A. thaliana) being higher or lower than the mean of the dataset. PRS calculation methods (conventional method cPRS and newly proposed method cgpPRS) are then employed to uncover parts of the relationship between genotype and phenotype and successfully separate samples based on their genotype.

Chapter 5

Dissertation Work Plan

Academic Year 2023/2024 - Summer Semester

- Conduct experiments on the application of modified CGP on the tasks of alcoholism and major depressive disorder classification from EEG data. This is a new problem, not discussed so far in this thesis, that will alow to experiment with proposed CGP methods on another type of time-series data and provide additional opportunity for comparison with ML algorithms already used in these tasks.
- Write and publish a paper describing achieved results and comparisons with other machine learning algorithms. Planned to be published at the Parallel Problem Solving from Nature (PPSN 2024) conference. CORE rank A, Google Scholar H5-index = 26.
- Propose a CGP-based method for simultaneous and energy-efficient classification of Parkinson's disease symptoms and the side effects of a drug used to suppress these symptoms.

Academic Year 2024/2025 - Winter Semester

- Undertake a mandatory one-month internship under the supervision of Professor Stephen Smith at the University of York, United Kingdom.
- Conduct experiments and write and publish a journal paper concluding research done in the field of Parkinson's disease. Planned to be published in one of the Q1/Q2 journals on the topic of evolutionary computation or soft computing (e.g., Applied Soft Computing, Evolutionary Computation, Soft Computing)

Academic Year 2024/2025 - Summer Semester

- Finalize and publish an open-source Python library containing proposed CGP-based methods and conduct a comparison with other machine learning methods.
- Write the dissertation.

Chapter 6

Conclusions

This document covered the basics of genetic programming, cartesian genetic programming, and current state-of-the-art variants and modifications of CGP. The open problems of CGP scalability, the use of CGP for designing hardware-aware classifiers, and CGP application in biomedical informatics and bioinformatics were presented. A research hypothesis was proposed based on current state-of-the-art methods and identified knowledge gaps. The presented hypothesis stands that incorporating multiple advanced techniques into the CGP could improve some of the method's important aspects and lead to new exciting solutions applicable in biomedical informatics and bioinformatics. The hypothesis was further translated into research goals. Already accomplished work on the dissertation topic was presented, followed by a time plan for the rest of the dissertation project.

Bibliography

[1] Adadi, A. and Berrada, M. Peeking Inside the Black-Box: A Survey on Explainable Artificial Intelligence (XAI). *IEEE access*. PISCATAWAY: IEEE. 2018, vol. 6, p. 52138–52160. ISSN 2169-3536.

[2] Ahmad, A. M., Khan, G. M., Mahmud, S. A. and Miller, J. F. Breast cancer detection using cartesian genetic programming evolved artificial neural networks. In: Soule, T., Auger, A., Moore, J., Pelta, D., Solnon, C. et al., ed. *Proceedings of the 14th Annual Conference on Genetic and Evolutionary Computation*. New York, NY, USA: Association for Computing Machinery, 2012, p. 1031–1038. GECCO '12. DOI: 10.1145/2330163.2330307. ISBN 9781450311779.

[3] Azaria, Y. and Sipper, M. GP-Gammon: Genetically Programming Backgammon Players. *Genetic Programming and Evolvable Machines*. Springer Science and Business Media LLC. september 2005, vol. 6, no. 3, p. 283–300. DOI: doi:10.1007/s10710-005-2990-0. ISSN 1389-2576.

[4] Bach, S., Binder, A., Montavon, G., Klauschen, F., Müller, K. R. et al. On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation. *PloS one*. SAN FRANCISCO: Public Library Science. 2015, vol. 10, no. 7, p. e0130140–e0130140. ISSN 1932-6203.

[5] Bader El Den, M. and Poli, R. Generating SAT Local-Search Heuristics Using a GP Hyper-Heuristic Framework. In: Monmarché, N., Talbi, E.-G., Collet, P., Schoenauer, M. and Lutton, E., ed. *Artificial Evolution*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, vol. 4926, p. 37–49. ISBN 9783540793045.

[6] Banzhaf, W. Genetic Programming for Pedestrians. In: Forrest, S., ed. *Proceedings of the 5th International Conference on Genetic Algorithms*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1993, p. 628. ISBN 1558602992.

[7] Bishop, C. M. *Pattern recognition and machine learning*. New York: Springer Science + Business Media, 2006. Information science and statistics. ISBN 978-1-4939-3843-8.

[8] Bouhaddou, M., Distefano, M. S., Riesel, E. A., Carrasco, E., Holzapfel, H. Y. et al. Drug response consistency in CCLE and CGP. *Nature (London)*. LONDON: Springer Nature. 2016, vol. 540, no. 7631, p. E9–E10. ISSN 0028-0836.

[9] Brameier, M. and Banzhaf, W. *Linear genetic programming*. New York: Springer, 2007. Genetic and evolutionary computation series. ISBN 978-0387-31029-9.

[10] Branco, H. M. G. C., Silva, R. P. M. da, Oleskovicz, M., Coury, D. V. and Delbem, A. C. B. Extended compact genetic algorithm applied for optimum allocation of power quality monitors in transmission systems. In: *2011 IEEE Power and Energy Society General Meeting*. IEEE, 2011, p. 1–6. DOI: 10.1109/PES.2011.6039680. ISBN 9781457710001.

[11] Burke, E., Kendall, G., Newall, J., Hart, E., Ross, P. et al. Hyper-Heuristics: An Emerging Direction in Modern Search Technology. In: Glover, F. and Kochenberger, G. A., ed. *Handbook of Metaheuristics*. Boston, MA: Springer US, 2003, p. 457–474. DOI: 10.1007/0-306-48056-5_16. ISBN 978-0-306-48056-0.

[12] Cai, W., Pacheco Vega, A., Sen, M. and Yang, K. Heat transfer correlations by symbolic regression. *International Journal of Heat and Mass Transfer*. november 2006, vol. 49, p. 4352–4359. DOI: 10.1016/j.ijheatmasstransfer.2006.04.029.

[13] Calzarossa, M. C., Giudici, P. and Zieni, R. Explainable Machine Learning for Bag of Words-Based Phishing Detection. In: Longo, L., ed. *Communications in Computer and Information Science*. 2023, vol. 1901, p. 531–543. ISBN 3031440633.

[14] Castillo, F., Kordon, A. and Smits, G. Robust Pareto Front Genetic Programming Parameter Selection Based on Design of Experiments and Industrial Data. In: Riolo, R., Soule, T. and Worzel, B., ed. *Genetic Programming Theory and Practice IV*. Boston, MA: Springer US, 2007, p. 149–166. DOI: 10.1007/978-0-387-49650-4_10. ISBN 978-0-387-49650-4.

[15] Chen, S.-H. and Liao, C.-C. Agent-based computational modeling of the stock price-volume relation. *Information Sciences*. 2005, vol. 170, no. 1, p. 75–100. DOI: doi:10.1016/j.ins.2003.03.026.

[16] Cho, J.-H., Wang, Y., Chen, I.-R., Chan, K. S. and Swami, A. A Survey on Modeling and Optimizing Multi-Objective Systems. *IEEE Communications surveys and tutorials*. New York: IEEE. 2017, vol. 19, no. 3, p. 1867–1901. ISSN 1553-877X.

[17] Clegg, J., Walker, J. A. and Miller, J. F. A new crossover technique for Cartesian genetic programming. In: Thierens, D., Beyer, H.-G., Bongard, J., Branke, J., Clark, J. A. et al., ed. *Proceedings of the 9th Annual Conference on Genetic and Evolutionary Computation*. New York, NY, USA: Association for Computing Machinery, 2007, p. 1580–1587. GECCO '07. DOI: 10.1145/1276958.1277276. ISBN 9781595936974.

[18] Cortacero, K., McKenzie, B., Müller, S., Khazen, R., Lafouresse, F. et al. Evolutionary design of explainable algorithms for biomedical image segmentation. *Nature communications*. London: Nature Publishing Group. 2023, vol. 14, no. 1, p. 7112–7112. ISSN 2041-1723.

[19] Dai, F., Kushida, N., Shang, L. and Sugisaka, M. A survey of genetic algorithm-based face recognition. *Artificial life and robotics*. Japan: Springer Japan. 2011, vol. 16, no. 2, p. 271–274. ISSN 1433-5298.

[20] de Castro, L. N. Fundamentals of natural computing: an overview. *Physics of Life Reviews*. 2007, vol. 4, no. 1, p. 1–36. DOI: https://doi.org/10.1016/j.plrev.2006.10.002. ISSN 1571-0645.

[21] Deb, K., Pratap, A., Agarwal, S. and Meyarivan, T. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE transactions on evolutionary computation*. PISCATAWAY: IEEE. 2002, vol. 6, no. 2, p. 182–197. ISSN 1089-778X.

[22] Drahosova, M., Sekanina, L. and Wiglasz, M. Adaptive Fitness Predictors in Coevolutionary Cartesian Genetic Programming. *Evol. Comput.* Cambridge, MA, USA: MIT Press. september 2019, vol. 27, no. 3, p. 497–523. DOI: 10.1162/evco_a_00229. ISSN 1063-6560.

[23] Gajda, Z. and Sekanina, L. An Efficient Selection Strategy for Digital Circuit Evolution. In: Tempesti, G., Tyrrell, A. M. and Miller, J. F., ed. *Evolvable Systems: From Biology to Hardware*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, p. 13–24. ISBN 978-3-642-15323-5.

[24] Garmendia Doval, A. B., Miller, J. F. and Morley, S. D. Cartesian Genetic Programming and the Post Docking Filtering Problem. In: O'Reilly, U.-M., Yu, T., Riolo, R. and Worzel, B., ed. *Genetic Programming Theory and Practice II*. Boston, MA: Springer US, 2005, p. 225–244. DOI: 10.10070-387-23254-0_14. ISBN 978-0-387-23254-6.

[25] Géron, A. *Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow : concepts, tools, and techniques to build intelligent systems*. Second editionth ed. Beijing: O'Reilly, 2019. ISBN 978-1-4920-3264-9.

[26] Goldman, B. W. and Punch, W. F. Reducing Wasted Evaluations in Cartesian Genetic Programming. In: Krawiec, K., Moraglio, A., Hu, T., Etaner Uyar, A. Ş. and Hu, B., ed. *Genetic Programming*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, p. 61–72. ISBN 978-3-642-37207-0.

[27] Goldman, B. W. and Punch, W. F. Analysis of Cartesian Genetic Programming's Evolutionary Mechanisms. *IEEE Transactions on Evolutionary Computation*. 2015, vol. 19, no. 3, p. 359–373. DOI: 10.1109/TEVC.2014.2324539.

[28] Gustafson, S., Burke, E. and Krasnogor, N. On improving genetic programming for symbolic regression. In: Corne, D., Michalewicz, Z., Dorigo, M., Eiben, G., Fogel, D. et al., ed. *2005 IEEE Congress on Evolutionary Computation*. IEEE, 2005, vol. 1, p. 912–919 Vol.1. DOI: 10.1109/CEC.2005.1554780. ISBN 0780393635.

[29] Harding, S. and Banzhaf, W. Fast Genetic Programming on GPUs. In: Ebner, M., O'Neill, M., Ekárt, A., Vanneschi, L. and Esparcia Alcázar, A. I., ed. *Genetic Programming*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007, p. 90–101. ISBN 978-3-540-71605-1.

[30] Harding, S., Graziano, V., Leitner, J. and Schmidhuber, J. MT-CGP: mixed type cartesian genetic programming. In: Soule, T., Auger, A., Moore, J., Pelta, D., Solnon, C. et al., ed. *Proceedings of the 14th Annual Conference on Genetic and Evolutionary Computation*. New York, NY, USA: Association for Computing Machinery, 2012, p. 751–758. GECCO '12. DOI: 10.1145/2330163.2330268. ISBN 9781450311779.

[31] Harding, S., Leitner, J. and Schmidhuber, J. Cartesian Genetic Programming for Image Processing. In: Riolo, R., Vladislavleva, E., Ritchie, M. D.

and Moore, J. H., ed. *Genetic Programming Theory and Practice X*. New York, NY: Springer New York, 2013, p. 31–44. DOI: 10.1007/978-1-4614-6846-2_3. ISBN 978-1-4614-6846-2.

[32] Harding, S. L., Miller, J. F. and Banzhaf, W. Self-modifying cartesian genetic programming. In: Thierens, D., Beyer, H.-G., Bongard, J., Branke, J., Clark, J. A. et al., ed. *Proceedings of the 9th Annual Conference on Genetic and Evolutionary Computation*. New York, NY, USA: Association for Computing Machinery, 2007, p. 1021–1028. GECCO '07. DOI: 10.1145/1276958.1277161. ISBN 9781595936974.

[33] Hazell, A. and Smith, S. L. Towards an objective assessment of alzheimer's disease: the application of a novel evolutionary algorithm in the analysis of figure copying tasks. In: Ebner, M., Cattolico, M., van Hemert, J., Gustafson, S., Merkle, L. D. et al., ed. *Proceedings of the 10th Annual Conference Companion on Genetic and Evolutionary Computation*. New York, NY, USA: Association for Computing Machinery, 2008, p. 2073–2080. GECCO '08. DOI: 10.1145/1388969.1389024. ISBN 9781605581316.

[34] Howard, D. and Roberts, S. C. Incident Detection on Highways: Development of GP Alarms for Motorway Management. In: O'Reilly, U.-M., Yu, T., Riolo, R. L. and Worzel, B., ed. *Genetic Programming Theory and Practice II*. Boston, MA: Springer US, 2005, p. 263–282. Genetic Programming. DOI: 10.10070-387-23254-0_16. ISBN 978-0-387-23254-6.

[35] Hurta, M., Drahosova, M. and Mrazek, V. Evolutionary Design of Reduced Precision Preprocessor for Levodopa-Induced Dyskinesia Classifier. In: Rudolph, G., Kononova, A. V., Aguirre, H., Kerschke, P., Ochoa, G. et al., ed. *Parallel Problem Solving from Nature - PPSN XVII*. Springer Nature Switzerland AG, 2022, vol. 13398, p. 491–504. Lecture Notes in Computer Science. DOI: 10.1007/978-3-031-14714-2_34. ISBN 978-3-031-14713-5.

[36] Hurta, M., Drahosova, M., Sekanina, L., Smith, L. S. and Alty, E. J. Evolutionary Design of Reduced Precision Levodopa-Induced Dyskinesia Classifiers. In: Medvet, E., Pappa, G. and Xue, B., ed. *Genetic Programming, 25th European Conference, EuroGP 2022*. Springer Nature Switzerland AG, 2022, vol. 13223, p. 85–101. Lecture Notes in Computer Science. DOI: 10.1007/978-3-031-02056-8_6. ISBN 978-3-031-02055-1.

[37] Hurta, M., Mrazek, V., Drahosova, M. and Sekanina, L. ADEE-LID: Automated Design of Energy-Efficient Hardware Accelerators for Levodopa-Induced Dyskinesia Classifiers. In: *2023 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. Institute of Electrical and Electronics Engineers, 2023, p. 1–2. DOI: 10.23919/DATE56975.2023.10137079. ISBN 978-3-9819263-7-8.

[38] Hurta, M., Mrazek, V., Drahosova, M. and Sekanina, L. MODEE-LID: Multiobjective Design of Energy-Efficient Hardware Accelerators for Levodopa-Induced Dyskinesia Classifiers. In: Jenihhin, M., Kubátová, H., Metens, N., Raik, J., Ahmed, F. et al., ed. *2023 26th International Symposium on Design and Diagnostics of Electronic Circuits and Systems (DDECS)*. Institute of

Electrical and Electronics Engineers, 2023, p. 155–160. DOI: 10.1109/DDECS57882.2023.10139399. ISBN 979-8-3503-3277-3.

[39] Hurta, M., Mrazek, V., Drahosova, M. and Sekanina, L. Multi-objective Design of Hardware Accelerators for Levodopa-Induced Dyskinesia Classifiers. In: *Evo*2023 – Late-Breaking Abstracts Volume*. In press.

[40] Hurta, M., Schwarzerova, J., Nagele, T., Weckwerth, W., Provaznik, V. et al. Utilizing Genetic Programming to Enhance Polygenic Risk Score Calculation. In: *2023 IEEE International Conference on Bioinformatics and Biomedicine (BIBM 2023)*. Institute of Electrical and Electronics Engineers, 2023, p. 3782–3787. DOI: 10.1109/BIBM58861.2023.10385615. ISBN 979-8-3503-3748-8.

[41] Hurta, M., Schwarzerova, J., Provaznik, V., Weckwerth, W., Walther, D. et al. Utilizing Cartesian Genetic Programming for Efficient Polygenic Risk Score Calculation in Plants. In: *Program and Abstract Book: Swedish Bioinformatics Workshop 2023*. 2023, p. 49.

[42] Husa, J. and Kalkreuth, R. A Comparative Study on Crossover in Cartesian Genetic Programming. In: Castelli, M., Sekanina, L., Zhang, M., Cagnoni, S. and García Sánchez, P., ed. *Genetic Programming*. Cham: Springer International Publishing, 2018, p. 203–219. ISBN 978-3-319-77553-1.

[43] Izzo, D., Biscani, F. and Mereta, A. Differentiable Genetic Programming. In: McDermott, J., Castelli, M., Sekanina, L., Haasdijk, E. and García Sánchez, P., ed. *Genetic Programming*. Cham: Springer International Publishing, 2017, p. 35–51. ISBN 978-3-319-55696-3.

[44] Kalkreuth, R. *Towards Advanced Phenotypic Mutations in Cartesian Genetic Programming*. 2018.

[45] Kalkreuth, R. *Reconsideration and Extension of Cartesian Genetic Programming*. 2021. Dissertation. Technische Universität Dortmund, Germany.

[46] Kalkreuth, R., Rudolph, G. and Droschinsky, A. A New Subgraph Crossover for Cartesian Genetic Programming. In: McDermott, J., Castelli, M., Sekanina, L., Haasdijk, E. and García Sánchez, P., ed. *Genetic Programming*. Cham: Springer International Publishing, 2017, p. 294–310. ISBN 978-3-319-55696-3.

[47] Kaufmann, P. and Platzner, M. Advanced techniques for the creation and propagation of modules in cartesian genetic programming. In: Keijzer, M., Antoniol, G., Congdon, C. B., Deb, K., Doerr, B. et al., ed. *Proceedings of the 10th Annual Conference on Genetic and Evolutionary Computation*. New York, NY, USA: Association for Computing Machinery, 2008, p. 1219–1226. GECCO '08. DOI: 10.1145/1389095.1389334. ISBN 9781605581309.

[48] Kaufmann, P. and Platzner, M. Combining Local and Global Search: A Multi-objective Evolutionary Algorithm for Cartesian Genetic Programming. In: Stepney, S. and Adamatzky, A., ed. *Inspired by Nature: Essays Presented to Julian F. Miller on the Occasion of his 60th Birthday*. Cham: Springer International Publishing, 2018, p. 175–194. DOI: 10.1007/978-3-319-67997-6_8. ISBN 978-3-319-67997-6.

[49] Kell, D. B., Allen, J., Davey, H. M., Broadhurst, D., Heald, J. K. et al. High-throughput classification of yeast mutants for functional genomics using metabolic footprinting. *Nature biotechnology*. BERLIN: NATURE PORTFOLIO. 2003, vol. 21, no. 6, p. 692–696. ISSN 1087-0156.

[50] Knezevic, K., Picek, S. and Miller, J. F. Amplitude-oriented mixed-type CGP classification. In: *Proceedings of the Genetic and Evolutionary Computation Conference Companion*. New York, NY, USA: Association for Computing Machinery, 2017, p. 1415–1418. GECCO '17. DOI: 10.1145/3067695.3082500. ISBN 9781450349390.

[51] Kovačič, M. and Balic, J. Evolutionary programming of a CNC cutting machine. *International Journal of Advanced Manufacturing Technology*. january 2003, vol. 22, p. 118–124. DOI: 10.1007/s00170-002-1450-8.

[52] Koza, J. R. *Genetic programming : On the programming of computers by means of natural selection*. Cambridge : London,: Bradford Book, MIT Press, 1992. Complex adaptiev systems. ISBN 0-262-11170-5.

[53] Langdon, W. B., Poli, R., McPhee, N. F. and Koza, J. R. Genetic Programming: An Introduction and Tutorial, with a Survey of Techniques and Applications. In: Fulcher, J. and Jain, L. C., ed. *Computational Intelligence: A Compendium*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, p. 927–1028. DOI: 10.1007/978-3-540-78293-3_22. ISBN 978-3-540-78293-3.

[54] Leitner, J., Harding, S., Forster, A. and Schmidhuber, J. Mars Terrain Image Classification using Cartesian Genetic Programming. In: *11th International Symposium on Artificial Intelligence, Robotics and Automation in Space (i-SAIRAS)*. Turin, Italy: [b.n.], September 2012.

[55] Li, Z., Liu, F., Yang, W., Peng, S. and Zhou, J. A Survey of Convolutional Neural Networks: Analysis, Applications, and Prospects. *IEEE Transactions on Neural Networks and Learning Systems*. 2022, vol. 33, no. 12, p. 6999–7019. DOI: 10.1109/TNNLS.2021.3084827.

[56] Lones, M. A., Alty, J. E., Cosgrove, J., Duggan-Carter, P., Jamieson, S. et al. A New Evolutionary Algorithm-Based Home Monitoring Device for Parkinson's Dyskinesia. *Journal of medical systems*. 2017, vol. 41, no. 11, p. 176:1–176:8. DOI: 10.1007/s10916-017-0811-7.

[57] Lones, M. A., Smith, S. L., Harris, A. T., High, A. S., Fisher, S. E. et al. Discriminating normal and cancerous thyroid cell lines using implicit context representation Cartesian genetic programming. In: Sobrevilla, P., ed. *IEEE Congress on Evolutionary Computation*. IEEE, 2010, p. 1–6. DOI: 10.1109/CEC.2010.5586494. ISBN 1424469090.

[58] López, E. G., Poli, R. and Coello, C. A. C. Reusing Code in Genetic Programming. In: Keijzer, M., O'Reilly, U.-M., Lucas, S., Costa, E. and Soule, T., ed. *Genetic Programming*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2004, p. 359–368. ISBN 978-3-540-24650-3.

[59] Ma, Y. and Zhou, X. Genetic prediction of complex traits with polygenic scores: a statistical review. *Trends in genetics*. England: Elsevier Ltd. 2021, vol. 37, no. 11, p. 995–1011. ISSN 0168-9525.

[60] Mahsal Khan, M., Masood Ahmad, A., Muhammad Khan, G. and Miller, J. F. Fast learning neural networks using Cartesian genetic programming. *Neurocomputing*. 2013, vol. 121, p. 274–289. DOI: https://doi.org/10.1016/j.neucom.2013.04.005. ISSN 0925-2312. Advances in Artificial Neural Networks and Machine Learning.

[61] Makke, N., Sadeghi, M. A. and Chawla, S. Symbolic Regression for Interpretable Scientific Discovery. In: Sachdeva, S., Watanobe, Y. and Bhalla, S., ed. *Big-Data-Analytics in Astronomy, Science, and Engineering*. Cham: Springer International Publishing, 2022, p. 26–40. ISBN 978-3-030-96600-3.

[62] Manazir, A. and Raza, K. Recent Developments in Cartesian Genetic Programming and Its Variants. *ACM Comput. Surv.* New York, NY, USA: Association for Computing Machinery. january 2019, vol. 51, no. 6. DOI: 10.1145/3275518. ISSN 0360-0300.

[63] Marney, J. P., Miller, D., Fyfe, C. and Tarbert, H. F. E. *Risk Adjusted Returns to Technical Trading Rules: a Genetic Programming Approach*. Computing in Economics and Finance 2001 147. Yale: Society for Computational Economics, april 2001.

[64] McKay, R., Hoai, N., Whigham, P., Shan, Y. and O'Neill, M. Grammar-based Genetic Programming: a survey. *Genetic Programming and Evolvable Machines*. september 2010, vol. 11, p. 365–396. DOI: 10.1007/s10710-010-9109-y.

[65] Meske, C., Bunde, E., Schneider, J. and Gersch, M. Explainable Artificial Intelligence: Objectives, Stakeholders, and Future Research Opportunities. *Information systems management*. Boston: Taylor & Francis. 2022, vol. 39, no. 1, p. 53–63. ISSN 1058-0530.

[66] Milano, N., Pagliuca, P. and Nolfi, S. *Robustness, Evolvability and Phenotypic Complexity: Insights from Evolving Digital Circuits*. 2017.

[67] Miller, J., Thomson, P. and Fogarty, T. Designing Electronic Circuits Using Evolutionary Algorithms. Arithmetic Circuits: A Case Study. *Genetic Algorithms and Evolution Strategies in Engineering and Computer Science*. 1997.

[68] Miller, J. F. An Empirical Study of the Efficiency of Learning Boolean Functions Using a Cartesian Genetic Programming Approach. In: Banzhaf, W., Daida, J., Eiben, A. E., Garzon, M. H., Honavar, V. et al., ed. *Proceedings of the 1st Annual Conference on Genetic and Evolutionary Computation - Volume 2*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1999, p. 1135–1142. GECCO'99. ISBN 1558606114.

[69] Miller, J. F., Smith, S. L. and Zhang, Y. Detection of microcalcifications in mammograms using multi-chromosome Cartesian genetic programming. In: *Proceedings of the 12th Annual Conference Companion on Genetic and Evolutionary Computation*. New York, NY, USA: Association for Computing

Machinery, 2010, p. 1923–1930. GECCO '10. DOI: 10.1145/1830761.1830827. ISBN 9781450300735.

[70] Miller, J. F. Cartesian genetic programming: its status and future. *Genetic programming and evolvable machines*. New York: Springer US. 2020, vol. 21, 1-2, p. 129–168. ISSN 1389-2576.

[71] Moore, J. H., Parker, J. S., Olsen, N. J. and Aune, T. M. Symbolic discriminant analysis of microarray data in autoimmune disease. *Genetic epidemiology*. New York: Wiley Subscription Services, Inc., A Wiley Company. 2002, vol. 23, no. 1, p. 57–69. ISSN 0741-0395.

[72] Ooi, K.-B., Tan, G. W.-H., Al Emran, M., Al Sharafi, M. A., Capatina, A. et al. The Potential of Generative Artificial Intelligence Across Disciplines: Perspectives and Future Directions. *The Journal of computer information systems*. 2023, p. 1–32. ISSN 0887-4417.

[73] Picek, S., Jakobovic, D., Miller, J. F., Batina, L. and Cupic, M. Cryptographic Boolean functions: One output, many design criteria. *Applied Soft Computing*. 2016, vol. 40, p. 635–653. DOI: https://doi.org/10.1016/j.asoc.2015.10.066. ISSN 1568-4946.

[74] Poli, R. *Parallel Distributed Genetic Programming*. Technical Report CSRP-96-15. University of Birmingham, B15 2TT, UK: School of Computer Science, september 1996.

[75] Romero, J. and Machado, P. *The Art of Artificial Evolution: A Handbook on Evolutionary Art and Music*. 1. Aufl.th ed. Berlin, Heidelberg: Springer-Verlag, 2008. Natural Computing Series. ISBN 9783540728764.

[76] Ryser Welch, P., Miller, J. F., Swan, J. and Trefzer, M. A. Iterative Cartesian Genetic Programming: Creating General Algorithms for Solving Travelling Salesman Problems. In: Heywood, M. I., McDermott, J., Castelli, M., Costa, E. and Sim, K., ed. *Genetic Programming*. Cham: Springer International Publishing, 2016, p. 294–310. ISBN 978-3-319-30668-1.

[77] Sekanina, L., Harding, S. L., Banzhaf, W. and Kowaliw, T. Image Processing and CGP. In: Miller, J. F., ed. *Cartesian Genetic Programming*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, p. 181–215. DOI: 10.1007/978-3-642-17310-3_6. ISBN 978-3-642-17310-3.

[78] Shao, Y., Cheng, Y., Shah, R. U., Weir, C. R., Bray, B. E. et al. Shedding Light on the Black Box: Explaining Deep Neural Network Prediction of Clinical Outcomes. *Journal of medical systems*. New York: Springer US. 2021, vol. 45, no. 1, p. 5–5. ISSN 0148-5598.

[79] Smith, S. L. Cartesian Genetic Programming and its Application to Medical Diagnosis. *IEEE Computational Intelligence Magazine*. 2011, vol. 6, no. 4, p. 56–67. DOI: 10.1109/MCI.2011.942583.

[80] Smith, S. L., Leggett, S. and Tyrrell, A. M. An Implicit Context Representation for Evolving Image Processing Filters. In: Rothlauf, F., Branke,

J., Cagnoni, S., Corne, D. W., Drechsler, R. et al., ed. *Applications of Evolutionary Computation, Proceedings*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005, vol. 3449, p. 407–416. Lecture Notes in Computer Science. DOI: 10.1007/978-3-540-32003-6_41. ISBN 9783540253969.

[81] Smith, S. L., Walker, J. A. and Miller, J. F. Medical Applications of Cartesian Genetic Programming. In: Miller, J. F., ed. *Cartesian Genetic Programming*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, p. 309–336. DOI: 10.1007/978-3-642-17310-3_11. ISBN 978-3-642-17310-3.

[82] Stoffel, K. and Spector, L. High-Performance, Parallel, Stack-Based Genetic Programming. In: Koza, J. R., Goldberg, D. E., Fogel, D. B. and Riolo, R. L., ed. *Proceedings of the 1st Annual Conference on Genetic Programming*. Cambridge, MA, USA: MIT Press, 1996, p. 224–229. ISBN 0262611279.

[83] Suganuma, M., Shirakawa, S. and Nagao, T. A genetic programming approach to designing convolutional neural network architectures. In: *Proceedings of the Genetic and Evolutionary Computation Conference*. New York, NY, USA: Association for Computing Machinery, 2017, p. 497–504. GECCO '17. DOI: 10.1145/3071178.3071229. ISBN 9781450349208.

[84] Taleby Ahvanooey, M., Li, Q., Wu, M. and Wang, S. A Survey of Genetic Programming and Its Applications. *KSII Transactions on Internet and Information Systems*. april 2019, Vol.13, p. 1765–1793. DOI: 10.3837/tiis.2019.04.002.

[85] Turner, A. J. *Improving Crossover Techniques in a Genetic Program*. 2012. Master's thesis. University of York.

[86] Turner, A. and Miller, J. Recurrent Cartesian Genetic Programming Applied to Famous Mathematical Sequences. In: *Proceedings of the Seventh York Doctoral Symposium on Computer Science & ElectronicsAt: York, UK*. January 2014.

[87] Turner, A. J. and Miller, J. F. Recurrent Cartesian Genetic Programming. In: Bartz Beielstein, T., Branke, J., Filipič, B. and Smith, J., ed. *Parallel Problem Solving from Nature – PPSN XIII*. Cham: Springer International Publishing, 2014, p. 476–486. ISBN 978-3-319-10762-2.

[88] Turner, A. J. and Miller, J. F. Recurrent Cartesian Genetic Programming of Artificial Neural Networks. *Genetic programming and evolvable machines*. New York: Springer US. 2017, vol. 18, no. 2, p. 185–212. ISSN 1389-2576.

[89] Vasicek, Z. Cartesian GP in Optimization of Combinational Circuits with Hundreds of Inputs and Thousands of Gates. In: Machado, P., Heywood, M. I., McDermott, J., Castelli, M., García Sánchez, P. et al., ed. *Genetic Programming*. Cham: Springer International Publishing, 2015, p. 139–150. ISBN 978-3-319-16501-1.

[90] Vasicek, Z. Bridging the Gap Between Evolvable Hardware and Industry Using Cartesian Genetic Programming. In: Stepney, S. and Adamatzky, A., ed. *Inspired by Nature: Essays Presented to Julian F. Miller on the Occasion of his 60th Birthday*. Cham: Springer International Publishing, 2018, p. 39–55. DOI: 10.1007/978-3-319-67997-6_2. ISBN 978-3-319-67997-6.

[91] Vasicek, Z. and Sekanina, L. Hardware Accelerators for Cartesian Genetic Programming. In: O'Neill, M., Vanneschi, L., Gustafson, S., Esparcia Alcázar, A. I., De Falco, I. et al., ed. *Genetic Programming*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, p. 230–241. ISBN 978-3-540-78671-9.

[92] Vasicek, Z. and Sekanina, L. Hardware Accelerator of Cartesian Genetic Programming with Multiple Fitness Units. *Computing and informatics (Bratislava, Slovakia)*. Bratislava: Slovak Acad Sciences Inst Informatics. 2010, vol. 29, no. 6, p. 1359–1371. ISSN 1335-9150.

[93] Vasicek, Z. and Slany, K. Efficient Phenotype Evaluation in Cartesian Genetic Programming. In: Moraglio, A., Silva, S., Krawiec, K., Machado, P. and Cotta, C., ed. *Genetic Programming*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, p. 266–278. ISBN 978-3-642-29139-5.

[94] Vowk, B., Wait, A. S. and Schmidt, C. An Evolutionary Approach Generates Human Competitive Coreware Programs. In: Bedau, M., Husbands, P., Hutton, T., Kumar, S. and Sizuki, H., ed. *Workshop and Tutorial Proceedings Ninth International Conference on the Simulation and Synthesis of Living Systems(Alife XI)*. Boston, Massachusetts: [b.n.], December 2004, p. 33–36.

[95] Walker, J. A. and Miller, J. F. Evolution and Acquisition of Modules in Cartesian Genetic Programming. In: Keijzer, M., O'Reilly, U.-M., Lucas, S., Costa, E. and Soule, T., ed. *Genetic Programming*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2004, p. 187–197. ISBN 978-3-540-24650-3.

[96] Whigham, P. A. Search Bias, Language Bias and Genetic Programming. In: Koza, J. R., Goldberg, D. E., Fogel, D. B. and Riolo, R. L., ed. *Proceedings of the 1st Annual Conference on Genetic Programming*. Cambridge, MA, USA: MIT Press, 1996, p. 230–237. ISBN 0262611279.

[97] Wilson, D., Miller, J. F., Cussat Blanc, S. and Luga, H. *Positional Cartesian Genetic Programming*. 2018.

[98] Yazdani, S. and Shanbehzadeh, J. Balanced Cartesian Genetic Programming via migration and opposition-based learning: application to symbolic regression. *Genetic Programming and Evolvable Machines*. USA: Kluwer Academic Publishers. june 2015, vol. 16, no. 2, p. 133–150. DOI: 10.1007/s10710-014-9230-4. ISSN 1389-2576.