



INFOTEC CENTRO DE INVESTIGACIÓN E
INNOVACIÓN EN TECNOLOGÍAS DE LA
INFORMACIÓN Y COMUNICACIÓN



DIRECCIÓN ADJUNTA DE INNOVACIÓN Y
CONOCIMIENTO
GERENCIA DE CAPITAL HUMANO
POSGRADOS



Selection Heuristics on Semantic Genetic Programming for Classification Problems



Tesis
Que para obtener el grado de DOCTORA EN
CIENCIAS EN CIENCIA DE DATOS

Presenta:

Claudia Nallely Sánchez Gómez

Asesor:

Dr. Mario Graff Guerrero



Aguascalientes, Julio 2020

Autorización de impresión



AUTORIZACIÓN DE IMPRESIÓN Y NO ADEUDO EN BIBLIOTECA DOCTORADO EN CIENCIAS EN CIENCIA DE DATOS

Ciudad de México, 10 de julio de 2020
INFOTEC-DAIC-GCH-SE-0437/2020.

La Gerencia de Capital Humano / Gerencia de Investigación hacen constar que el trabajo de titulación intitulado

SELECTION HEURISTICS ON SEMANTIC GENETIC PROGRAMMING FOR CLASSIFICATION PROBLEMS

Desarrollado por la alumna **Claudia Nallely Sánchez Gómez** y bajo la asesoría del **Dr. Mario Graff Guerrero**; cumple con el formato de biblioteca. Por lo cual, se expide la presente autorización para impresión del proyecto terminal al que se ha hecho mención.

Asimismo se hace constar que no debe material de la biblioteca de INFOTEC.

Vo. Bo.

A handwritten signature in blue ink, appearing to read "Julieta Alcibar Hermosillo", is written over a light blue rectangular background.

Mtra. Julieta Alcibar Hermosillo
Coordinadora de Biblioteca

Anexar a la presente autorización al inicio de la versión impresa del trabajo referido que ampara la misma.

C.p.p Servicios Escolares

Resumen

En esta tesis, se proponen tres heurísticas para la selección de padres en Programación Genética (PG) basadas en propiedades de funciones e información semántica de los individuos. La semántica se ha utilizado recientemente para guiar el proceso de aprendizaje cuando PG es utilizada para resolver problemas de aprendizaje supervisado. Sin embargo, de lo mejor de nuestro conocimiento, esta es la primera vez que las propiedades de las funciones son utilizadas para guiar el proceso de aprendizaje de PG. Las heurísticas están diseñadas para la función suma Σ , y los clasificadores *Naive Bayes* y *Nearest Centroid*. La primera heurística basada en la similitud coseno promueve la selección de padres cuyas semánticas son lo más perpendicular posible entre ellas en el espacio semántico. La segunda, basada en el coeficiente de correlación de Pearson, busca padres cuyos vectores semánticos no están correlacionados. Finalmente, la última heurística, basada en *accuracy*, trata de seleccionar padres cuya semánticas son diferentes entre ellas. Además, analizamos el uso de elección complementamente aleatoria para selección de padres y selección negativa. Estas técnicas de selección fueron implementadas en EvoDAG, un sistema de Programación Genética Semántica. Para comparar los diferentes esquemas de selección, usamos 30 problemas de clasificación con un variado número de muestras, variables y clases. Los resultados indican que la combinación de nuestra heurística basada en *accuracy* para la selección de padres y selección negativa aleatoria genera la mejor combinación, y la diferencia en eficiencia entre esta combinación y la selección clásica basada en aptitud es estadísticamente significativa. Además también comparamos nuestras heurísticas con los esquemas del estado del arte, *Angle-Driven Selection*, y *Novelty Search*. EvoDAG junto con las heurísticas propuestas fue comparado con 18 clasificadores que incluyen enfoques tradicionales, así como técnicas de auto aprendizaje computacional. Concluimos que el uso de nuestras heurísticas mejora significativamente el proceso de aprendizaje de EvoDAG.

Abstract

In this dissertation, three heuristics for parent selection in Genetic Programming (GP) based on functions' properties and individuals' semantics have been proposed. Semantics have recently used for guiding the learning process when GP is used to solve supervised learning problems. However, to the best of our knowledge, this is the first time that functions' properties are used for guiding the learning process in GP. The heuristics are tailored to the function Σ , and the classifiers Naive Bayes and Nearest Centroid. The first heuristic based on cosine similarity promotes the selection of parents whose semantics are as perpendicular as possible between them in the semantics space. The second one, based on Pearson's correlation coefficient, searches parents whose semantics' vectors are uncorrelated. Finally, the last one, based on accuracy, tries to select parents whose semantics are different between them. In addition, we analyze the use of completely random selection for parents and negative selection. These selection techniques were implemented on EvoDAG, a Semantic Genetic Programming system. For comparing the different selection schemes, we use 30 classification problems with a variable number of samples, variables, and classes. The results indicate that the combination of our heuristic based on accuracy for parent selection and negative random selection produces the best combination, and the difference in performances between this combination and the classical selection based on fitness is statistically significant. Furthermore, we compare our heuristics with state-of-the-art schemes, Angle-Driven Selection, and Novelty Search. Besides, EvoDAG with the proposed selection heuristics was compared against 18 classifiers that included traditional approaches as well as auto-machine-learning techniques. We conclude that the use of our proposed heuristics significantly improves the learning process of EvoDAG.

Acknowledgements

This thesis could not be possible without the invaluable help of many people.

Firstly, my sincere thanks to my advisor Dr. Mario Graff for his continuous support, motivation, patience, and encouragement. Your guidance helped me a lot all the time on my Ph.D. journey. Thanks for share with us your immense knowledge and experience. Besides being a well-prepared researcher, he was always open to help us. I consider that it is complicated to find a good advisor, and I have the fortune to have an excellent one.

Besides my Ph.D. advisor, I would like to thank my other advisors: Dr. Mariano Rivera and Dra. Julieta Domínguez. Dr. Mariano Rivera has been mi advisor since my master's degree. He believed in me and still supporting my research work. Thanks very much for encouraging me and for showed me this amazing area of research. I want to thank Julieta for her guidance and encouragement. Thanks for pressuring me because thanks to that, I am here.

I would like to thank the rest of my thesis committee: Dr. Sabino Miranda, Dra. Daniela Moctezuma and Dr. Eric Téllez. Together with Dr. Mario Graff, they have created a pleasant working environment where all of us felt comfortable. They have shared with us their knowledge, and they give us always support and comments for improving our work. The most valuable of INFOTEC and INGEOTEC is the quality of people that they have, speaking of professionalism and humanism. Besides, thanks to Dr. Efrén Mezura for his valuable comments and suggestions that help me to improve the quality of my thesis.

I thank my classmates, Jose Ortíz, Abel Coronado, Sergio Nava, Vladimir Salgado, and Aldo Márquez. Together we have lived this journey filled with a lot of emotions, like happiness and stress. Thanks for helping me and for giving me your friendship.

Thanks to Universidad Panamericana for the support during my Ph.D. Espe-

cially, I want to thank my bosses for all their support, facilities, and comprehension: Juan Carlos García, Ramiro Velázquez, Ricardo Macías, Pedro Rodrigo, Begoña Noriega, and Erika Hernández. Also, I want to thank my fellow researchers for their support and advises, mainly thanks to Héctor Buendía Escalona and Josué Enríquez. Thanks to Tere Orvañanos, we were together in this Ph.D. journey. Thanks for the days of working and for being there to listen to me. Also, thanks to Karla Salazar, for being my friend and the best English teacher. Since the beginning, this Ph.D. could not be possible without your help. Additionally, I want to thank my brilliant students from UP and Infotec for motivating me.

Last but not least, I would like to thank God and my family. Thanks to my parents Raquel Gómez and Javier Sánchez, for the love, guidance, and help that you have given me all my life. Especially, thanks to my mom for her support and help during these years. Thanks to my husband, Ernesto Santillán, for your patience and comprehension. Mainly, thanks for your love and for being my support. My daughter, Valeria Santillán, thanks for bringing to my life happiness, and motivation. Thanks to my siblings Cindy Sánchez and Luis Javier Sánchez, for being always there for me. Thank all my family for your support and encouragement. I love you.

Por último pero no menos importante, quiero agradecer a Dios y a mi familia. Gracias a mis padres Raquel Gómez y Javier Sánchez, por el amor, la guía y el apoyo que me han dado durante toda mi vida. Especialmente, gracias por el infito apoyo durante estos años. Gracias a mi esposo, Ernesto Santillán, por tu paciencia y comprensión. Principalmente, gracias por tu amor y por ser siempre mi soporte. A mi hija, Valeria, gracias por llenar mi vida de felicidad y motivación. Gracias a mis hermanos Cindy y Luis Ja, por estar siempre ahí para mi. Gracias a toda mi familia y amigos por su apoyo y aliento. Los amo.

Contents

Introduction	1
1 Genetic Programming	12
1.1 A Brief Introduction to Evolutionary Computing	12
1.2 General Concepts of Genetic Programming	15
1.3 Semantic Genetic Programming for Supervised Learning Problems	21
1.4 EvoDAG	22
1.5 Summary	33
2 Related Work	37
2.1 Indirect Semantic Genetic Programming	37
2.2 Direct Semantic Genetic Programming	41
2.3 Fitness and Selection in Genetic Programming	51
2.4 Classification in Genetic Programming	54
2.5 Summary	58
3 Selection Heuristics	65
3.1 Motivation	65
3.2 Parent Selection	67
3.2.1 Parent Selection based on Fitness (fit)	68
3.2.2 Random Selection of Parents (rnd)	68
3.2.3 Parent Selection based on Cosine Similarity (sim) and Pearson's Correlation Coefficient (prs)	71
3.2.4 Parent Selection based on the Accuracy (acc)	80
3.3 Negative Selection	82
3.3.1 Negative Selection based on Fitness (fit)	84
3.3.2 Random Negative Selection (rnd)	85
3.4 Summary	85

4 Experiments and Results	88
4.1 Datasets	89
4.2 Computer Equipment	91
4.3 Performance Metrics	91
4.4 Comparison of the Proposed Selection Heuristics and Classic Selection Techniques	93
4.5 Comparison of the Proposed Selection Heuristics and State-of-the-Art Selection Schemes	107
4.6 Comparison of EvoDAG and State-of-the-Art Classifiers	118
4.7 Visualization Map for Classifiers and Datasets	124
4.8 Summary	129
Conclusions	132
Bibliography	150

List of Figures

1.1	Diagram of evolutionary algorithms' evolution process	14
1.2	Example of a GP individual representation structure	15
1.3	Genetic Programming syntax tree	16
1.4	Illustration of Genetic Programming crossover	19
1.5	Illustration of Genetic Programming mutation	19
1.6	Example of genotype and phenotype spaces	22
1.7	Example of a model evolved by EvoDAG on the Iris dataset	26
1.8	Example of EvoDAG's individuals and their semantics in the initial population \mathcal{P}_0	30
1.9	Example of EvoDAG's individuals and their semantics when a new offspring is created	31
2.1	Illustration of Geometric Semantic Genetic Programming crossover operator	43
2.2	Illustration of Random Desired Operator (RDO)	45
2.3	Illustration of Forward Propagation Mutation (FPM)	46
2.4	Comparison of Moraglio's and Graff's crossover operators	47
3.1	Diagram of parent selection in EvoDAG	67
3.2	Diagram of parent selection based on fitness (in EvoDAG)	69
3.3	Diagram of random selection of parents (in EvoDAG)	71
3.4	Example of Nearest Centroid classifier with uncorrelated and correlated variables	73
3.5	Example of cosine Similarity	74
3.6	Diagram of the relationship between linearly independent, uncorrelated, and orthogonal vectors	75
3.7	Diagram of parent selection based on cosine similarity (in EvoDAG)	78
3.8	Diagram of parent selection based on Pearson's coefficient (in EvoDAG)	80
3.9	Diagram of parent selection based on accuracy (in EvoDAG)	82

4.1	Selections schemes results based on macro-F1 ranks and time	97
4.2	Analysis of selection schemes combinations based on macro-F1 average rank and the time	99
4.3	Statistical comparison of selection schemes combinations	100
4.4	Analysis of EvoDAG acc-rnd according to the number of models in the ensemble	104
4.5	Analysis of selection schemes combinations changing the number of arguments in the function Σ	106
4.6	Example of parent selection based on ADS (in EvoDAG)	110
4.7	Example of individual's novelty calculation	112
4.8	Parent selection based on novelty search (in EvoDAG)	112
4.9	Proposed heuristics against state-of-the-art selection schemes based on macro-F1	115
4.10	Proposed heuristics against state-of-the-art selection schemes based on macro-F1 average rank and time	117
4.11	Comparison of EvoDAG against state-of-the-art classifiers based on macro-F1 rank	119
4.12	Statistical comparison of the different classifiers based on macro-F1	120
4.13	Comparison of EvoDAG against state-of-the-art classifiers based on the time required by the classifiers' training phase	121
4.14	Comparison of EvoDAG with state-of-the-art classifiers based on macro-F1 and the time required by the classifiers' training phase	123
4.15	Comparison of different selection schemes for parent and negative selection based on macro-F1 rank using LPL as a visualization map	126
4.16	Comparison of EvoDAG vs state-of-the-art classifiers based on macro-F1 rank using LPL as a visualization map	128

List of Tables

1.1	Function set of EvoDAG	24
1.2	The possible values of EvoDAG's parameters	33
1.3	The possible number of arguments of EvoDAG's functions	34
1.4	EvoDAG's parameters	34
4.1	Datasets used to compare the performance of the algorithms	90
4.2	Characteristics of the computer where the experiments were executed	91
4.3	Results of EvoDAG with different selection schemes (test sets)	95
4.4	Results of EvoDAG with different selection schemes (training sets)	96
4.5	Results of EvoDAG with different selection schemes ordered by classes' entropy (test sets)	101
4.6	Results of EvoDAG with different selection schemes ordered by number of classes (test sets)	102
4.7	Results of EvoDAG with different selection schemes ordered by number of variables (test sets)	103
4.8	Selection techniques that recently use the angle between individuals' semantics	135
4.9	Comparison of Novelty Search and Random Selection	136
4.10	Comparison of recent GP classifiers	137

List of Algorithms

1	Evolution process of Genetic Programming	20
2	Evolution process of EvoDAG	29
3	Selection of arguments based on fitness	70
4	Random selection of arguments	70
5	Selection of arguments based on the absolute of cosine similarity	77
6	Selection of arguments based on Pearson's correlation coefficient	79
7	Selection of arguments based on the accuracy	83
8	Negative selection of individuals based on fitness	84
9	Random negative selection	85
10	Selection of arguments based on Angle-Driven Selection	109
11	Selection of arguments based on Novelty Search	113

Acronyms and Abbreviations

acc	Tournament selection based on accuracy
ADS	Angle-Driven Selection
EAs	Evolutionary algorithms
EC	Evolutionary computing
fit	Tournament selection based on fitness
GP	Genetic Programming
SGP	Semantic Genetic Programming
GSGP	Geometric Semantic Genetic Programming
OLS	Ordinary Least Squares
prs	Tournament selection based on Pearson's correlation coefficient
rnd	Random selection
sim	Tournament selection based on cosine similarity

Glossary

Evolutionary Computation is a research area within computer science, as the name suggests, it is a particular flavor of computing, which draws inspiration from the process of natural evolution. It is a family of algorithms for solving combinatorial and optimization problems.

Genetic Programming is an evolutionary algorithm that evolves a population of computer programs for solving a specific problem.

selection is one of the main characteristics of evolutionary algorithms. It consists of choose individuals from the population for being parents from the next generation.

Semantic Genetic Programming is an approach of Genetic Programming that uses semantics for guiding the learning process.

semantics is a representation of Genetic Programming individuals where they are seen as vectors in a multidimensional space, known as semantic space. The vector's entries correspond to the evaluation of the training samples in the function that the individual represents.

tournament is a selection technique. It consists of comparing several individuals from the population, randomly chosen, and select the best one.

Introduction

Nowadays, with advances in technology, we can store and process large amounts of data, as well as access it from physically distant locations over a computer network [1]. Bank transactions, satellite images, video cameras for surveillance, and industrial sensors are examples of large amounts of data. Nevertheless, this stored data becomes useful only when it is analyzed and turned into information [1].

Applications in which the data comprises several samples with their corresponding targets are known as *supervised learning* problems. The main idea is that a model can learn of labeled samples to make predictions for new data. Specifically, the supervised learning problems, where the samples are grouped in different categories, are called *classification* problems. Some examples of classification are the followings [25]:

- Predict whether a patient, hospitalized due to a heart attack, will have a second heart attack. The prediction could be based on demographics, diet, and clinical measurements for that patient.
- Identify the numbers in a handwritten zip code, from a digitized image.
- Identify the emotion in a short text.

Formally, *classification* consists of finding a function that learns a relation between input variables x_1, \dots, x_m and one output t . In this case, the value t is taken from a set of labels. The starting point would be a set of samples, this is, input-output pairs. In each sample, the input vector $\vec{x}^{(i)} \in \mathbb{R}^m$ is related to its output $t^{(i)}$, in this case, $\vec{x}^{(i)} = [x_1^{(i)}, \dots, x_m^{(i)}]$. For example, the input of a sample could be the results of a blood test, and the output the label that indicates whether the patient has a specific illness or not. Formally, the set of samples, input-output pairs, is called training set, i.e, $\mathcal{X} = \{(\vec{x}^{(1)}, t^{(1)}), \dots, (\vec{x}^{(n)}, t^{(n)})\}$. Returning to the the example, using many samples of blood tests with the label that indicates if the patients had the illness or not, we can train an algorithm to predict whether new patients have or not the illness based on

their blood tests. Formally, the training set is used to find a function f that minimize an error function, \mathcal{E} , that is, f is the function that minimize $\sum_{(\vec{x}, t) \in \mathcal{X}} \mathcal{E}(f(\vec{x}^{(i)}), t^{(i)})$ where the ideal scenario would be $\forall_{(\vec{x}, t) \in \mathcal{X}} f(\vec{x}) = t$. Once the function f is learned, it can be use for predicting new inputs.

There are well-known techniques that can satisfactorily solve classification problems, for example, Artificial Neural Networks (ANN), Support Vector Machines (SVM), and Decision Trees (DT). However, those techniques follow a hard defined structure. ANN needs to have several artificial neurons organized in layers, and the learning process consists of finding the neurons' weights. SVM is a linear classifier that separates the data of different classes using a hyper-plane. Also, it can use kernels to perform a non-linear classifier. The kernels and the model are defined, and the learning process consists of calculating the parameters of functions. DT divide the data space into hyper-rectangles with the idea of grouping together samples with the same label, and then, in the prediction phase, they assign to the new sample the label that corresponds to the majority of samples in the hyper-rectangle. Its model is composed of a series of binary questions that give a final prediction. All these examples of techniques can find successful models that fit the data. However, they always follow the same structure for all problems. On the other hand, Genetic Programming (GP) is an evolutionary algorithm that can find a function that is adapted to the classification problem. Its proposal does not follow a hard structure. GP finds a function in a search space composed of all the possible combinations of elementary functions (e.g., +, -, *, /) and terminals (e.g., input variables, constants). In this sense, GP can return as a classifier, the same model of an SVM, or an ANN, but it also can find a more straightforward model or a more complex one.

Genetic programming (GP) [48], proposed by John Koza around the 1990s, is one of the youngest members of the evolutionary algorithm family. It has been used to solve a variety of problems. For example, autonomous controllers for unmanned aerial vehicles [2, 75], prediction of energy performance [11, 61], antennas designing

[19, 63], sentimental analysis [31, 34], predicting financial data [45, 88], circuits designing [49, 71, 86]. However, the flexibility of GP for solving supervised learning problems makes it very difficult to converge. It means that the process of searching in a vast space of functions and terminals is very complicated. Most of research documents presents GP for solving symbolic regression problems (see Section 2.5). To the best of our knowledge, the classifiers constructed with GP started appearing around the 2000s (see Section 2.5). Despite its shortage, robust models have been developed, as M4GP [60] or TPOT [76], those techniques are explained in detail in Chapter 2. We consider that GP, specifically for solving supervised learning problems, can be improved and deserves the attention of researchers because it presents a new way of constructing models, GP evolves models instead of only adjusting parameters.

Motivation

GP individuals are commonly represented as syntax trees, and when the objective is solving supervised learning problems, those syntax trees represent functions that model the relationship between inputs and outputs. Moreover, in supervised learning problems, *Semantic Genetic Programming (SGP)* has received much attention because it has allowed improving the performance of GP in recent years. According to Vanneschi [97], in SGP, the individuals can be represented in two spaces. On the first hand, we have the *genotype space*, where individuals are represented by their tree structures. On the other hand, in the *semantics space*, each individual is represented by a vector that contains the outputs of the individual's function when all the training samples are evaluated (see Figure 1.6). The *target* vector, which contains the real outputs, also is a point in the semantic space. From this point of view, the learning process consists of finding the individual whose semantics is as close as possible to the target vector. The use of individuals' semantics is very useful for designing operators and GP systems. Krawiec [52] affirmed that aware semantic methods make search algorithms better informed.

Based on our review of the state-of-the-art (see Chapter 2), most of the work

in SGP is focused on the development of crossover and mutation operators. However, only a few documents have focused on selection, that in GP is traditionally performed using *tournament selection*, where a set of individuals are randomly chosen from the population, and the most adapted one is selected for being a parent. Some of the documents that are focus on selection are the followings. They explained in detail in Section 2.3. In Novelty Search [62], 2011, the fitness function is replaced by the novelty of individuals that is computed as the average of the distances between itself and its nearest neighbors. The fitted individuals are the ones that are distant to the rest of individuals. The result is that the population diversity is promoted. In 2016, Hara *et al.* proposed Deterministic Geometric Semantic Genetic Programming with Optimal Mate Selection [38], where a selection technique was designed with the objective of improving the performance of their crossover operator. In 2018, Chu *et al.* [17, 18] used the Wilcoxon signed-rank test to compare individuals' semantics and decide whether to select the individual with the best fitness or the smallest one. Angle-Driven Selection (ADS) [15], 2019, aims to choose parents whose angles between their relative semantics are big with the goal of selecting different parents. Nested alignment genetic programming (NAGP) [98], 2019, introduces a selection scheme based on five selection criteria, which had been organized into a nested tournament. Their main objective is to find individuals whose angle between their error vectors is equal to zero.

Nevertheless, since a model can be evolved with GP using a set of functions and terminals, we consider that selection can be guided, besides individuals' semantics, by functions' properties. For example, in a multidimensional space, the function Σ can be seen as a linear combination, and linearly independent vectors are appropriate when that function is going to be used. To the best of our knowledge, no one has tried to use functions' properties for designing selection techniques, and we consider that it can improve the performance of SGP.

On the other hand, only some GP classifiers have been proposed, and they started to appearing around 2000s (see Section 2.5). Ingalalli *et al.* affirmed in 2014

that GP was never regarded as a good method to perform multi-class classification [46]. However, some robust GP classifiers have been introduced recently. The auto-machine learning technique called Tree-based Pipeline Optimization Tool (TPOT) [76], proposed in 2016, uses GP for developing a robust algorithm that automatically constructs and optimizes machine learning pipelines with the aim of solving regression and classification problems. In 2019, M4GP [60] was proposed. It is an extended version of M2GP [46] and M3GP [70]. The main idea is to transform the original space into another one using functions evolved with GP, then, they calculate a centroid for each class, and the vectors are assigned to the class that corresponds to the nearest centroid using the Mahalanobis distance. TPOT and M4GP are explained in Section 2.4. Nevertheless, we consider that it is essential to improve the process of evolving full GP models for solving supervised learning problems, as it is done in EvoDAG [36], proposed by Graff *et al.*, and explained in Section 1.4.

Summarizing, this dissertation has been motivated by the following:

- Genetic Programming, proposed in the 1990s, is one of the youngest members of the evolutionary algorithm family. We believed that much of its potential for solving supervised learning problems had not been discovered yet.
- According to Vanneschi [99], in 2014, methods that work directly in the semantic space are recent, and much of its potentiality is to be seen.
- Most of SGP research focuses on developing crossover and mutation operators, but only a few of them proposes new selection schemes.
- To the best of our knowledge, no one has tried to use functions' properties for designing selection techniques, and we consider that it can improve the learning phase of SGP.
- Classification has a lot of practical applications, and GP tools for solving supervised learning problems have recently been proposed, as EvoDAG [36], M4GP [60], and TPOT [76]. We consider that we can research new selection schemes for improving those kinds of GP classifiers.

Objectives

The main objective of this thesis is to develop new parent selection schemes based on functions' properties and individuals' semantics. In addition to performing a comparison of classical and state-of-the-art selection techniques. The idea is to answer questions such as: Is it worth selecting parents based on fitness? What will happen if the individuals are randomly selected? Which functions and which properties can be helpful for designing selection techniques?

GP algorithms traditionally use the *steady-state* population model. It starts with a population of individuals, and in each iteration, one individual is created and it replaces another one from the population. There are two stages where selection takes place, on the one hand, the selection is used to choose the parents of the new individual, and on the other hand, the selection is applied to decide which individual, in the current population, is replaced with the offspring. We called the first stage *parent selection* and the second one *negative selection*. Chapter 1 explains this process in detail. The goal is comparing different techniques for both stages: parent and negative selection.

Besides, we want to concentrate into classification problems. The idea is designing selection techniques for improving SGP in those specific problems.

In summary the objectives of this thesis are:

- Developing new parent selection schemes based on functions' properties and individuals' semantics.
- Performing a comparison of classical and state-of-the-art selection techniques for parent and negative selection.
- Using the new selection techniques to improve the performance of SGP for solving classification problems.

Contribution

Based on the results (see Chapter 4), this dissertation has probed that fitness is not always the best option for selecting individuals. In this thesis, we proposed three heuristics for parent selection in GP that were designed based on the properties of the function sum (Σ), and the classifiers Naive Bayes (NB and MN) and Nearest Centroid (NC) (see Chapter 3). To the best of our knowledge, this is the first time that selection schemes based on functions' properties have been proposed. The heuristics are designed for solving classification problems. They are the followings:

- *Tournament selection based on cosine similarity* (sim). It consists of selecting individuals whose semantics' vectors ideally have rectangle angles. The absolute cosine similarity is used to measure the angle between individuals' semantics. It is explained in Section 3.2.3.
- *Tournament selection based on Pearson's correlation coefficient* (prs). The main idea is promoting the selection of parents whose semantics' vectors are uncorrelated. We use the absolute Pearson's correlation coefficient for measuring the correlation among inputs. It is explained in Section 3.2.3.
- *Tournament selection based on accuracy* (acc). The main idea is promoting the selection of parents whose predictions are different, and this is measured with the accuracy score. It is explained in Section 3.2.4.

Besides, we analyze two techniques for negative selection, one of them based on fitness and the other totally random selection (see Section 3.3).

We implemented classical and state-of-the-art selection techniques, in addition to the proposed heuristics, in the GP system EvoDAG (see 1.4). Based on the results (see Chapter 4), the combination of tournament selection based on accuracy for parent selection plus negative random selection statistically outperformed the classical selection tournaments based on fitness. Besides, that combination also improved the performance of EvoDAG and positioned it as one of the best classifiers. It is based on

the comparison of EvoDAG against sixteen scikit-learn classifiers [80] and two auto-machine-learning tools.

List of Publications

Throughout the time in the Ph.D. program, some manuscripts have been published.

Journal Article

- Claudia N. Sánchez, Julieta Domínguez-Soberanes*, Héctor B. Escalona-Buendía, Mario Graff*, Sebastián Gutiérrez and Gabriela Sánchez. *Liking Product Landscape: Going Deeper into Understanding Consumers' Hedonic Evaluations*. *Foods* 2019, 8, 461.

* Corresponding authors

Proceedings

- Sanchez, C.N., Graff, M. *Semantic genetic operators based on a selection mechanism tailored for the root function*. (2018) 2017 IEEE International Autumn Meeting on Power, Electronics and Computing, ROPEC 2017, 2018-January, pp. 1-6.
- Ortiz-Bejar, J., Tellez, E.S., Graff, M., Miranda-Jiménez, S., Ortiz-Bejar, J., Moctezuma, D., Sanchez, C.N. *I3GO+ at RICATIM 2017: A semi-supervised approach to determine the relevance between images and text-annotations*. (2018) 2017 IEEE International Autumn Meeting on Power, Electronics and Computing, ROPEC 2017, 2018-January, pp. 1-6.
- Graff, M., Miranda-Jimenez, S., Tellez, E.S., Moctezuma, D., Salgado, V., Ortiz-Bejar, J., Sanchez, C.N. *INGEOTEC at MEX-A3T: Author profiling and aggressiveness analysis in Twitter using microTC and EvoMSA*. (2018) CEUR Workshop Proceedings, 2150, pp. 128-133.

Thesis Outline

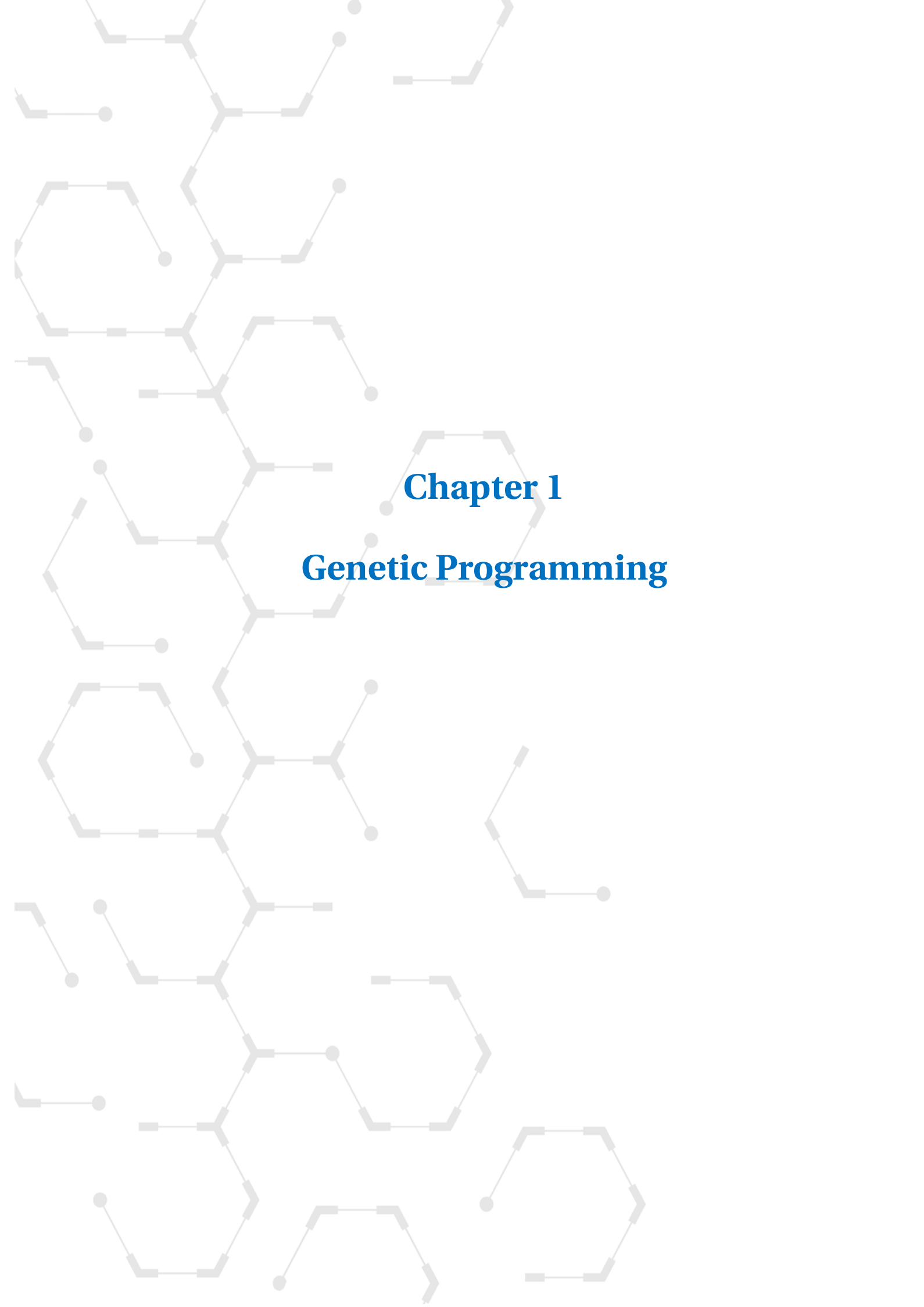
Chapter 1 describes Genetic Programming. It starts with a brief introduction to Evolutionary Computation describing the evolutive process and the key concepts as representation, fitness function, selection, and the variation operators: crossover and mutation. It is followed by a description of Genetic Programming, including its classical representation, applications, initialization, and operators. Moreover, we describe how Semantic Genetic Programming can be used for solving supervised learning problems. Finally, we describe EvoDAG, the GP system where the proposed selection heuristics are implemented.

Chapter 2 presents the related work. First, we describe how semantics have been used for improving the performance of GP, including the direct and indirect use of it. Inside of the direct use of semantics, we present Geometric Semantic Genetic Programming (GSGP) and some of the operators that have been inspired by it. Then, we review some of the work that changes the fitness function and performs the search using another objective. Also, we mention the research that has been focused on selection, specifically in GP. Finally, a review of GP classifiers is shown.

Chapter 3 introduces the proposed selection heuristics and the inspiration that we used. First, we describe the motivations of these selection schemes. Then, the chapter is divided into parent and negative selection techniques. For parent selection, we propose three selection heuristics: tournament selection based on cosine similarity (*sim*), tournament selection based on Pearson's correlation coefficient (*prs*), and tournament selection based on the accuracy (*acc*). We also describe, for parent and negative selection, tournament selection based on fitness (*fit*) and random selection (*rnd*).

Chapter 4 describes the experiments and results. The chapter starts with the experiment description, this is, the datasets, the computer equipment characteristics, and the performance metrics that were used. Then, the performance of EvoDAG

with different selection schemes is presented. First, an analysis of the classic selection technique based on fitness against random selection, and the proposed selection heuristics, is presented. Then, we include in the comparison the state-of-the-art selection schemes, Angle-Driven Selection [15], and Novelty Search [72]. Finally, we compare EvoDAG with the proposed heuristics against eighteen classifiers, sixteen of them from the python library scikit-learn [80]: Perceptron, MLPClassifier, BernoulliNB, GaussianNB, KNeighborsClassifier, NearestCentroid, LogisticRegression, LinearSVC, SVC, SGDClassifier, PassiveAggressiveClassifier, DecisionTreeClassifier, ExtraTreesClassifier, RandomForestClassifier, AdaBoostClassifier and GradientBoostingClassifier; and the others are two auto-machine learning libraries: autosklearn [23] and TPOT [76]. In all cases, the Wilcoxon test was used for validating our results.



Chapter 1
Genetic Programming

1 Genetic Programming

This chapter presents a brief introduction to Evolutionary Computation (EC) and their primary operations. Moreover, this chapter introduces Genetic Programming (GP), one of the youngest members of the evolutionary algorithm family, its applications, representation, and operators. Besides, we present the concept of *semantics* and how Semantic Genetic Programming has been used for solving supervised learning problems.

Finally, we present EvoDAG, a GP system, where our selection heuristics are implemented and tested.

1.1 A Brief Introduction to Evolutionary Computing

Evolutionary computing (EC) [21] is a research area within computer science, as the name suggests, it is a particular flavor of computing, which draws inspiration from the process of natural evolution. Darwin's theory of evolution is described in the book "Introduction to Evolutionary Computation" [21] as follows. Darwin's theory of evolution explains biological diversity and its underlying mechanisms. Given an environment that can host only a limited number of individuals, and the basic instinct of individuals to *reproduce*, selection becomes inevitable if the population size is not to grow exponentially. Natural *selection* favors those individuals that compete for the given resources most effectively; in other words, those that are adapted or fit to the environmental conditions best. This phenomenon is also known as the *survival of the fittest*. Competition based selection is one of the two cornerstones of evolutionary progress. The other primary force identified by Darwin results from *phenotypic variations* among members of the population. Phenotypic traits are those behavioral and physical features of an individual that directly affect its response to the environment, thus determining its fitness. Each individual represents a unique combination of phenotypic traits that is evaluated by the environment. If it evaluates favorably, then it is

propagated via the individual's offspring; otherwise, it is discarded by dying without offspring. Darwin's insight was that small, *mutations* in phenotypic traits occur during reproduction from generation to generation. Through these variations, new combinations of traits occur and get evaluated. The best ones survive and reproduce, and so evolution progresses. To summarise this basic model, a population consists of several individuals. These individuals are the "units of selection", that is to say that their reproductive success depends on how well they are adapted to their environment relative to the rest of the population. As the more successful individuals reproduce, occasional mutations give rise to new individuals to be tested. Thus, as time passes, there is a change in the constitution of the population, i.e., the population is the "unit of evolution".

The evolution theory has been used as an inspiration for several Evolutionary Algorithms (EAs) to solve hard problems. The connection between Darwin's theory and the algorithms is the following. First, the problem to be solved represents the environment, where the individuals, which are represented by the solutions, evolve. In this sense, the best solutions to the problem represent individuals that are well adapted to the environment. The key concepts in EC are:

- **Representation.** It is the structure of a solution in such a way that it can be used for the algorithm. It is divided into genotype and phenotype. Genotype comes from the word "gen"; it is the code representation that is used for the crossover and mutation operators. Phenotype is the solution that the genotype represents. Generally, it is used to calculate the individual's fitness.
- **Fitness function.** It is the way of calculating the individual's aptitude. It must receive the phenotype and return a numeric value that indicates how well the individual solves the problem.
- **selection.** Its objective is to warranty the quality of individuals. The main idea is to conserve the proper individuals and remove the bad ones.
- **Variation operators: crossover and mutation.** Crossover generates new individ-

uals, or offsprings, mixing the characteristics of parents. Mutation variates an individual to explore the search space.

The general evolution process of the EAs is described in the Figure 1.1.

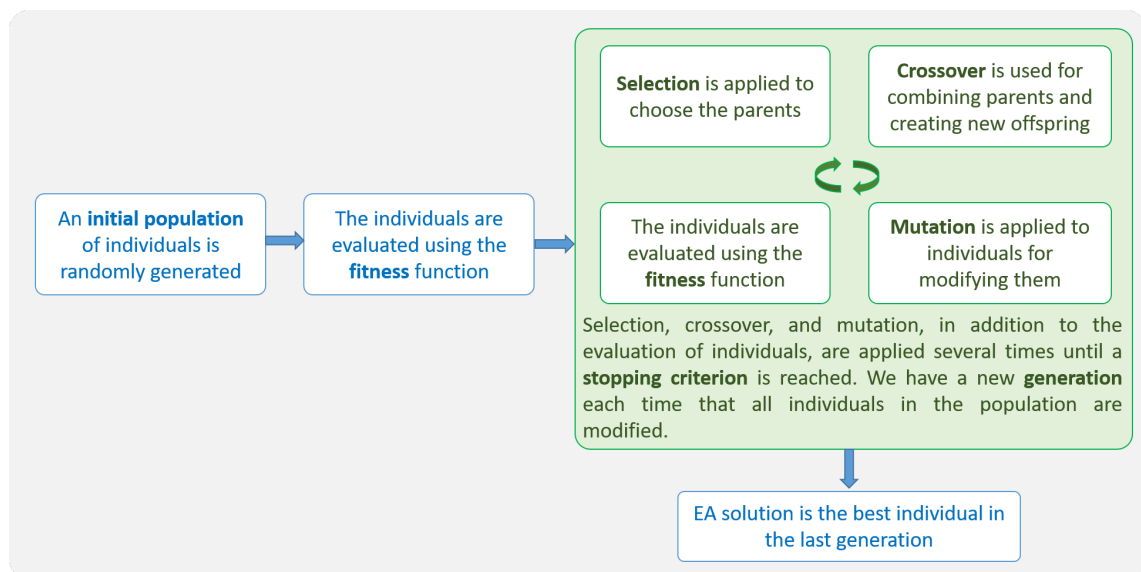


Figure 1.1: Diagram of evolutionary algorithms' evolution process. Source: Own elaboration.

The evolutive algorithms started appearing around the 1960s. Holland introduced *genetic algorithms* [42,43]. Maybe it is the most known and used algorithm from the EAs. The main characteristic of genetic algorithms is their inspiration on chromosomes, where a solution is represented by a bit string. Fogel proposed *evolutionary programming* [24], originally they evolved finite state machines. Rechenberg and Schwefel invented *evolution strategies* [84,89] to solve numerical optimization problems. Those techniques starting a research area that around the 1990s was called *evolutionary computing*. In the 1990s arose two new approaches. Storn and Price proposed *differential evolution* [91], a powerful algorithm for solving continuous optimization problems. *Genetic programming* was introduced by Koza [48], its main characteristic is its representation, where programs can be evolved. Commonly, individuals are represented as syntax trees. There are other EAs, but we mention only the most known and used ones.

1.2 General Concepts of Genetic Programming

Genetic programming (GP) [48], proposed by John Koza around the 1990s, is one of the youngest members of the evolutionary algorithm family. It automatically solves problems without requiring the user to know or specify the form or structure of the solution in advance [82]. The main idea of GP is to evolve a population of computer programs, where individuals are commonly represented as syntax trees, as it is shown in Figure 1.2. While the EAs are typically applied to optimization problems, GP could instead be positioned in machine learning [21]. For example, GP can solve the problems mentioned at the beginning of the Introduction.

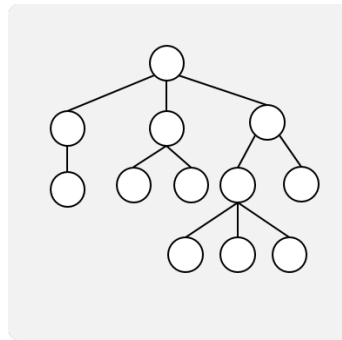


Figure 1.2: Example of a GP individual representation structure. Source: Own elaboration.

GP can be used to solve a wide variety of problems. For example, synthesis and design of circuits [49, 50, 71, 86], automatic synthesis of antennas [19, 63], design of autonomous controllers [2, 75], modeling the financial time series [45, 88], forecasting in the energy area [11, 61], and, sentiment analysis [31, 34].

Specifically, in Machine Learning, GP has been used in several phases of the process. As a pre-processing technique [46, 60, 70]. For improving the performance or structure of different algorithms, like decision trees [41, 59], support vector machines [94], or convolutional neural networks [93]. As a full model selection algorithm as done in TPOT (Tree-based Pipeline Optimization Tool) [76]. Alternatively, for evolving full models as EvoDAG [36], which is described in detail in Section 1.4.

Representation

In Genetic Programming, individuals are usually represented as syntax trees [82] (see Figure 1.3).

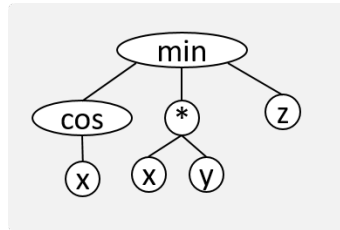


Figure 1.3: Genetic Programming syntax tree that represents the function $\min(\cos(x), xy, z)$. Source: Own elaboration.

The elements of a GP individual are:

- *Terminals*, which correspond to the leaf nodes in a syntax tree. They can be inputs, typically called variables (e.g., x and y), functions without arguments (e.g., $\text{rand}()$), or constants that can be predefined or randomly generated. The terminal set \mathcal{T} defines all the terminal elements.
- *Functions*, which correspond to the internal nodes in a syntax tree and can be seen as the operations. The function set \mathcal{F} is defined by the problem's nature. For example, for numeric problems, it could be formed by arithmetic operators and trigonometric functions.

Solution of a Problem

The first step for solving a problem with Genetic Programming is to define a **fitness function**. It needs to return a numeric value that indicates how well the individual solves the problem. For example, if we are using GP for recognizing faces, the fitness can be calculated as the number of images that the classifier predicts correctly. Specifically, for classification problems, defined at the beginning of the Introduction, the objective is to find an individual whose function matches inputs with outputs reducing an error function. Then, the fitness function could return the inverse of the error function, in this sense, an individual with a small error could have good fitness, and, in the

opposite, an individual with a big error will have poor fitness.

Once the fitness function is defined, several GP individuals are evolved using the following process (see Algorithm 1). First, an **initial population** of programs, as the one showed in Figure 1.2, is created. There are three classical techniques to generate the initial population [82]. The first one is called *full*, where all the trees are randomly created, and all the leaves have the same depth. It means, in all levels, except the last one, the nodes are internal nodes selecting elements randomly from the function set \mathcal{F} and only in the last level the elements are selected randomly from the terminal set \mathcal{T} . The second technique is called *grow*; in this case, each node selects an element randomly from either the function set \mathcal{F} or the terminal set \mathcal{T} . Finally, the technique *Ramped half-and-half* where half of the population is created with the *full* technique and the other half with *grow*. After creating the initial population, the fitness of all individuals is calculated. Also, the best program, based on fitness, and its fitness are stored.

The population is iteratively changed until a **termination criterion** is reached. The termination criterion may be a maximum number of iterations to be run as well as a problem-specific success predicate [82]. Using the example of faces recognition, the termination criterion could be to have as maximum 10,000 iterations, or, finding a program that can recognize the 90% of the images, whichever comes first.

Two different population models exist in EC [21]: the *generational* model and the *steady-state* model. In the first one, in each generation, we begin with a population of individuals, and several parents are selected from that population to generate the offspring by the application of variation operators. After each generation, the whole population is replaced by its offspring, which is called the next generation. On the other side, in the steady-state model, the entire population is not changed at once, but rather a part of it. Usually, in each iteration, an individual is generated, and it replaces another individual in the population (see Algorithm 1). Most of the recent GP implementation uses the steady-state model [21, 82]. In this dissertation, we use the steady-state model.

As it can be seen in Algorithm 1, in each iteration, a new program is created, it is the result of applying the variation operators (crossover and mutation) to two programs that are selected, based on fitness, as parents. The selection, crossover, and mutation are explained as follows:

- **Selection:** As most evolutionary algorithms, the genetic operators in GP are applied to individuals that are stochastically selected based on fitness. This is, more adapted individuals have more chances to be part of the evolutionary process than the ones who are less adapted. According to Mezura-Montes [66], in evolutive algorithms, we can distinguish between two selection techniques. The first one is *proportional selection*, where individuals are selected based on their contribution to the total amount of fitness of the population. Furthermore, *tournament selection* is a direct comparison of fitness among individuals. It can be either binary (two individuals) or with more than two individuals that are randomly selected from the population. The fittest one wins. In GP, tournament selection is the most used technique [22]. In Chapter 3, several selection heuristics are proposed to improve the performance of GP for solving classification problems.
- **Crossover:** The objective of crossover in GP, as well in other evolutive algorithms, is to combine the characteristics of two or more individuals, called parents, to generate the offspring. The classic crossover consists of randomly selecting a crossover point in each parent and create an offspring based on the first parent but replacing the selected node by the subtree of the second parent on the crossover point [82] (see Figure 1.4).
- **Mutation:** The goal of mutation is to change an individual. The most commonly used form of mutation in GP, called *subtree mutation*, randomly selects a mutation point in a tree and substitutes the subtree rooted there with a randomly generated subtree [82] (see Fig 1.5). Another common mutation is *point mutation*, where a function node is replaced for another one selecting an element from the

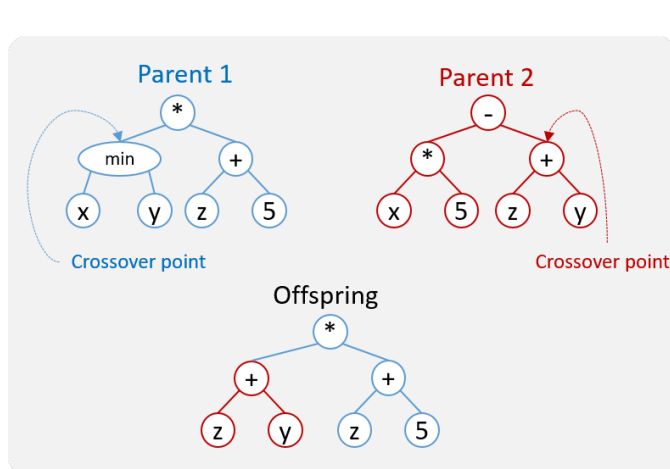


Figure 1.4: Illustration of Genetic Programming crossover. Source: Own elaboration.

function set \mathcal{F} with the same arity.

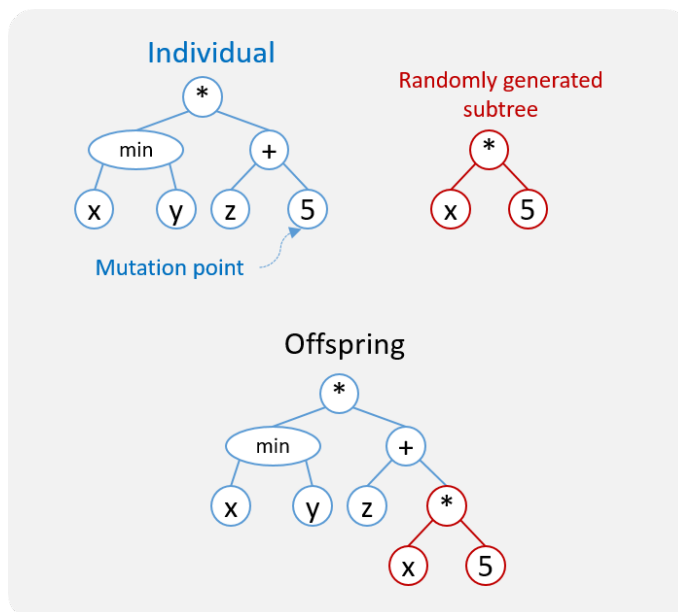


Figure 1.5: Illustration of Genetic Programming mutation. Source: Own elaboration.

Once the new program is created, it is added to the population, and to maintain the population size, it replaces another one that is chosen using negative selection (see Algorithm 1). In GP, the most used technique for negative selection is negative tournament, it consists of randomly selecting several programs from the population, and the worst of them, based on fitness, is the one that is going to be replaced. In addition, the new program is compared with the best one, and the fittest is kept as the best program.

At the end of the process, the best program based on fitness, is the one that is chosen as the problems' solution.

Algorithm 1: Evolution process of Genetic Programming. The underlined steps correspond to the ones that are analyzed in this dissertation.

```
Create an initial population  $\mathcal{P}$  of programs;  
Calculate the fitness of all programs in  $\mathcal{P}$ ;  
 $best, best_{fitness} \leftarrow$  The best program of  $\mathcal{P}$  and its fitness;  
while a termination criterion is not reached do  
     $parent_1, parent_2 \leftarrow$  Two programs that are selected from  $\mathcal{P}$  based on fitness;  
  
     $new \leftarrow$  The program that is the result of applying the variation  
        operators (crossover and mutation) to the programs  $parent_1$  and  
         $parent_2$ ;  
     $old \leftarrow$  A program that is chosen from  $\mathcal{P}$  using negative selection;  
    Remove  $old$  from  $\mathcal{P}$ ;  
    Add  $new$  to  $\mathcal{P}$ ;  
     $new_{fitness} \leftarrow$  The fitness of  $new$ ;  
    if  $new_{fitness} > best_{fitness}$  then  
        |  $best \leftarrow new$ ;  
        |  $best_{fitness} \leftarrow new_{fitness}$ ;  
    end  
end  
Return  $best$ ;
```

1.3 Semantic Genetic Programming for Supervised Learning Problems

We can define supervised learning as follows [97]: given a set of samples, or input-vectors, $\{x^{(1)}, x^{(2)}, \dots, x^{(n)}\}$, where $\forall_{i=1,2,\dots,n} : x^{(i)} \in \mathbb{R}^m$, this is, $x^{(i)} = [x_1^{(i)}, \dots, x_m^{(i)}]$, and a *target* vector $\vec{t} = \{t^{(1)}, t^{(2)}, \dots, t^{(n)}\}$, where $\forall_{i=1,2,\dots,n} : t^{(i)} \in \mathbb{R}$, a supervised learning problem can be defined as the problem of finding a function f that minimizes an error function, \mathcal{E} , that is, f is the function that minimizes $\sum_{i=1}^n \mathcal{E}(f(x^{(i)}), t^{(i)})$ where the ideal scenario would be $\forall_{i=1,2,\dots,n} f(x^{(i)}) = t^{(i)}$. In this way, a GP individual P can be seen as a function that, for each input vector $x^{(i)}$ returns the scalar value $P(x^{(i)})$, and the objective of GP is to find the GP individual that minimizes $\sum_{i=1}^n \mathcal{E}(P(x^{(i)}), t^{(i)})$.

We call *semantics* of P to the vector whose entries are all the responses to the input-vectors, this is, $\vec{S}_P = [P(x^{(1)}), P(x^{(2)}), \dots, P(x^{(n)})]$. *Semantic Genetic Programming (SGP)* is called in that way because it uses semantics in its evolutionary process. We can imagine the existence of two spaces: the *genotype space*, where individuals are represented by their structures, and the *phenotype* or *semantic space*, where individuals are represented by points, which are their semantics. Remark that the target vector \vec{t} itself is a point in the semantics space. The dimensionality of the semantics space is the number of input-vectors. Figure 1.6 shows an example of the genotype and phenotype spaces of an unrealistic case with only three input-vectors and five GP individuals. As we can see, the aim of *Semantic Genetic Programming (SGP)* in supervised learning problems is to find the tree structure of the individual whose semantics \vec{S}_P be as close as possible to the target \vec{t} in the semantic space. Most of the GP implementations for supervised learning use as fitness function the distance between individuals' semantics and the target vector.

It could seem easy to find the individual whose semantics is as close as possible to the target. However, the problem is that the search is performed in the genotype space, and the results are observed in the semantic space. Pawlak *et al.* affirmed in [78]

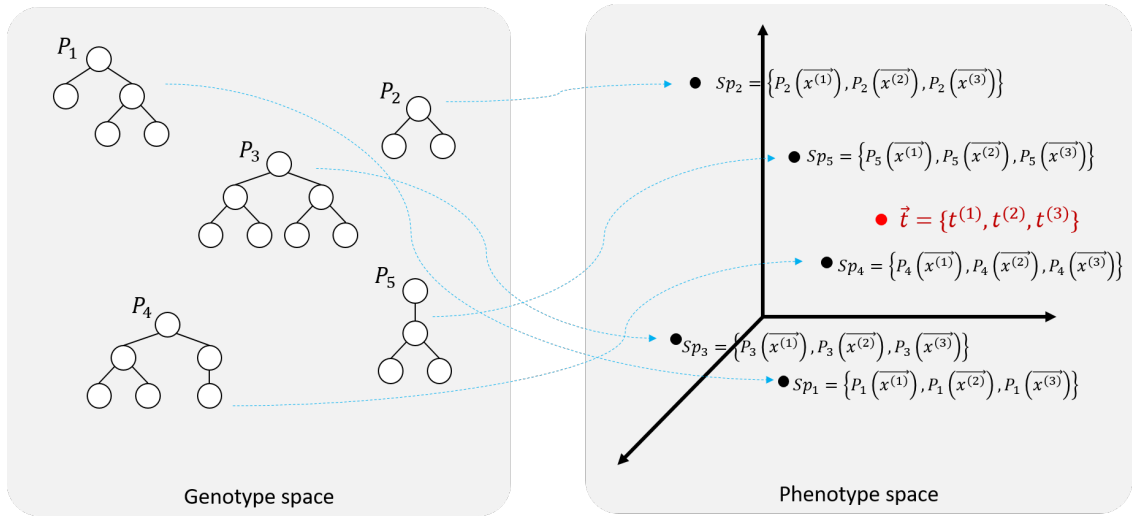


Figure 1.6: Example of genotype and phenotype spaces. Source: Own elaboration.

that focus on the syntax trees is a complex process because a small variation in the individual syntax can result in a dramatic change in its response (semantic space). On the other hand, a significant difference in the syntax cannot affect the response. Nevertheless, Semantic Genetic Programming changes this because it gives information about the individuals' behavior.

1.4 EvoDAG

In this section we describe EvoDAG [36], a GP system proposed by Graff *et al.* that is used for testing our proposed selection heuristics (see Chapters 3 and 4). EvoDAG is tailored to tackle classification and regression problems, but, we focus in using and describing how EvoDAG constructs classification models.

Its name means Evolving Directed Acyclic Graph. It is a python library that implements a steady-state Genetic Programming with tournament selection. It was inspired by the geometric semantic crossover [67] proposed by Moraglio *et al.* and the implementation performed by Castelli, Vanneschi, and Silva [10, 100]. Specifically, it has been used for solving complex problems of text categorization [32–34, 65, 77].

The code of EvoDAG has been uploaded in the public repository:

<https://github.com/mgraffg/EvoDAG>.

Model

As we mentioned in the first chapter, the starting point would be a set of samples, this is, input-output pairs. In each sample, the input vector $x^{(i)} \in \mathbb{R}^m$ is related to its output $t^{(i)}$, in this case, $x^{(i)} = [x_1^{(i)}, \dots, x_m^{(i)}]$. The set of samples, input-output pairs, is called training set, i.e, $\mathcal{X} = \{(x^{(1)}, t^{(1)}), \dots, (x^{(n)}, t^{(n)})\}$, that is conformed by n samples and m variables. EvoDAG uses the training set to find a function f such that for all samples, a performance metric \mathcal{M} is maximized. Formally, EvoDAG solves the optimization problem defined in Equation 1.1. The ideal scenario would be $\forall_{(\vec{x}, t) \in \mathcal{X}} f(\vec{x}) = t$. Once the function f is learned, it can be use for predicting new inputs. Specifically, for solving classification problems, EvoDAG uses as performance metric the macro-F1 score, described in detail in Section 4.3.

$$\arg_f \max \sum_{(\vec{x}, t) \in \mathcal{X}} \mathcal{M}(f(\vec{x}), t) \quad (1.1)$$

As we mentioned in Section 1.2, a GP individual represents a solution, in this case a function f . GP individuals' functions are formed by the combination of elements in the terminal set \mathcal{T} and the function set \mathcal{F} . In EvoDAG, the terminal set \mathcal{T} is composed only by the input variables of the problem; this is $\mathcal{T} = \{x_1, \dots, x_m\}$. On the other hand, the function set for classification problems is defined as $\mathcal{F} = \{\Sigma_{60}, \Pi_{60}, \max_5, \min_5, \text{NB}_5, \text{MN}_5, \text{NC}_5, \sin, \tan, \text{atan}, \tanh, \sqrt{\cdot}, |\cdot|, \text{hypot}_2\}$, where the subscript indicates the number of arguments. The procedure that was used for defining the number of functions' arguments is explained below. Table 1.1 describes the function set \mathcal{F} of EvoDAG for classification problems. As it can be observed, in the function set \mathcal{F} , the Naive Bayes and Nearest Centroid classifiers are included, they are described as follows.

Table 1.1: Function set of EvoDAG

Symbol	Number of arguments	Description
Σ	60	Sum of all arguments
Π	20	Product of all arguments
max	5	Maximum of all arguments
min	5	Minimum of all arguments
$\sqrt{\cdot}$	1	Square root of the argument
$ \cdot $	1	Absolute value of the argument
sin	1	Sine of the argument
tan	1	Tangent of the argument
atan	1	Arc tangent root of the argument
tanh	1	Hyperbolic tangent of the argument
hypot	2	Given the “legs” of a triangle, it return its hypotenuse, $\sqrt{x_1^2 + x_2^2}$
NB	5	Naive Bayes classifier with Gaussian distribution
MN	5	Naive Bayes classifier with Multinomial distribution
NC	5	Nearest Centroid classifier

Naive Bayes is a classifier based on the Bayes’ theorem with the “naive” assumption of independence between every pair of features. Let be x_1, \dots, x_m the input features and y the output class, the Bayes’ theorem say that we can predict the output y given the input variables based on the Equation 1.2.

$$P(y|x_1, \dots, x_m) = \frac{P(y)P(x_1, \dots, x_m|y)}{P(x_1, \dots, x_m)} \quad (1.2)$$

Using the “naive” assumption of independence among features, we have the Equation 1.3.

$$P(y|x_1, \dots, x_m) = \frac{P(y)P(x_1|y)\dots P(x_m|y)}{P(x_1, \dots, x_m)} = \frac{P(y)\prod_{i=1}^m P(x_i|y)}{P(x_1, \dots, x_m)} \quad (1.3)$$

Since $P(x_1, \dots, x_m)$ is constant for all classes, we can say that $P(y|x_1, \dots, x_m)$ is proportional to the numerator term in Equation 1.3. It is showed in Equation 1.4.

$$P(y|x_1, \dots, x_m) \propto P(y) \prod_{i=1}^m P(x_i|y) \quad (1.4)$$

To calculate the predicted class \hat{y} given the variables x_1, \dots, x_m , it is needed to find the class that maximizes Equation 1.4, this is, Equation 1.5.

$$\hat{y} = \arg_y \max P(y) \prod_{i=1}^m P(x_i|y) \quad (1.5)$$

For avoiding numeric errors in Equation 1.5, the sum of logarithms is used (see Equation 1.6).

$$\hat{y} = \arg_y \max \log(P(y)) + \sum_{i=1}^m \log(P(x_i|y)) \quad (1.6)$$

On Equation 1.6, the term $P(y)$ is obtained based on the frequency of classes. However, the term $P(x_i|y)$ is calculated based on the type of the features x_1, \dots, x_m . The versions of Naive Bayes, Naive Bayes with Gaussian distribution (NB) and Naive Bayes with Multinomial distribution (MN), are described as follows:

- *Naive Bayes with Gaussian distribution* (NB) is for continuous variables. The term $P(x_i|y)$ is calculated using a normal distribution. Formally, it is defined as Equation 1.7, where μ and σ represent the average and the standard deviation of variable x_i only for the samples of class y .

$$P(x_i|y) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(x_i - \mu)^2}{2\sigma^2}\right) \quad (1.7)$$

- *Naive Bayes with Multinomial distribution* (MN) is for discrete variables. The term $P(x_i|y)$ is calculated for each value u of variable x_i . The probability $P(x_i == u|y)$ corresponds to the proportion of class y 's samples with the value u . Formally, it is defined as Equation 1.8, where N_y is the number of class y 's samples, and $\delta(\cdot)$ returns 1 if its input is true and 0 otherwise.

$$P(x_i == u|y) = \frac{\sum_{x^{(j)} \in y} \delta(x_i^{(j)} == u)}{N_y} \quad (1.8)$$

Nearest Centroid (NC) is a simple classifier based on distances. It calculates the centroid of each class as the average of all its samples in the training set. Then, for predicting a new sample \vec{x} , it assigns the class that corresponds to the nearest centroid.

Formally, it can be defined as Equation 1.9, where $d(\cdot, \cdot)$ is a distance metric, in EvoDAG, d is implemented as the Euclidean distance, and $\vec{\mu}_y$ represents the centroid of class y .

$$\hat{y} = \arg_y \min d(\vec{\mu}_y, \vec{x}) \quad (1.9)$$

In order to provide an idea of the type of models produced by EvoDAG, Figure 1.7 presents a model of the Iris data set. The inputs ($x_0, \dots, x_3, \text{NB}, \text{MN}, \text{NC}$) are at the bottom of the figure. The computation flow goes from bottom to top, being the output the node in the top of the figure, i.e., Naive Bayes with Gaussian distribution.

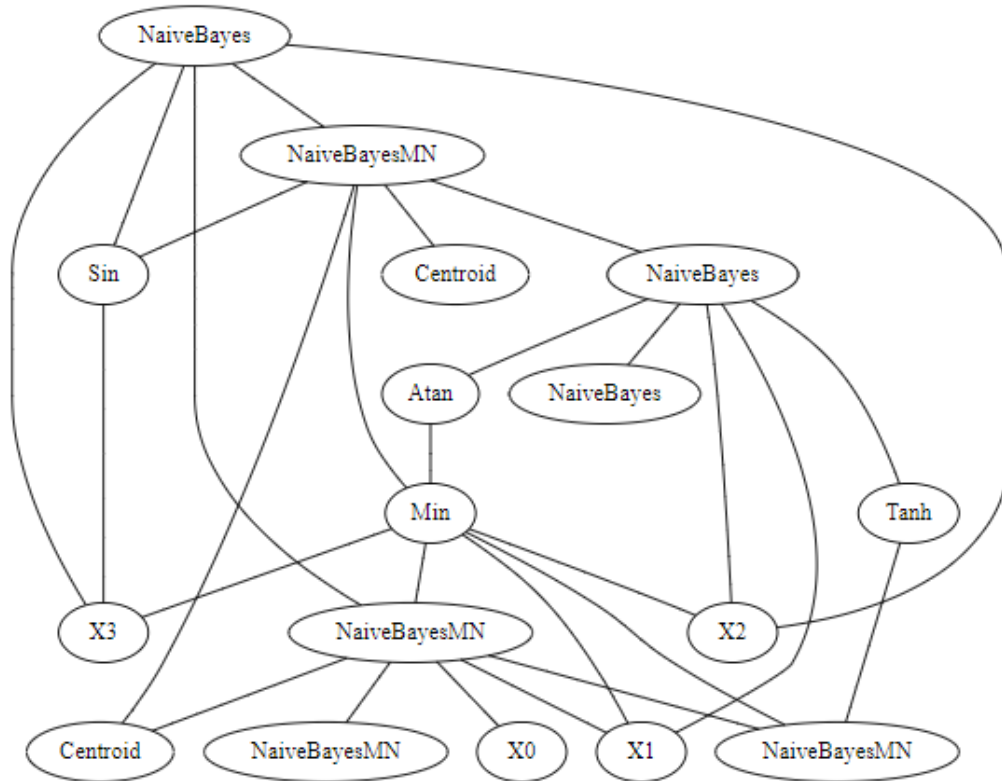


Figure 1.7: Example of a model evolved by EvoDAG on the Iris dataset. The inputs are at the bottom of the figure, and the output is on the top. Source: Own elaboration.

Evolution process

This subsection contains the EvoDAG's process for evolving a classifier. Algorithm 2 presents a summary of this process. The first **input** of EvoDAG is the training set \mathcal{X} .

The second input is the fitness function $fit(individual, \mathcal{X})$ whose aim is to solve the optimization problem defined in Equation 1.1. The fitness function $fit(individual, \mathcal{X})$ has two arguments, the first one is the individual, and the second one is the training set \mathcal{X} where the individual's function is tasted. The **output** is the EvoDAG's model that corresponds to the function f that solves the optimization problem defined in Equation 1.1.

On the first step, the training set $\mathcal{X} = \{(x^{(1)}, t^{(1)}), \dots, (x^{(n)}, t^{(n)})\}$ is randomly split into a smaller training set $\mathcal{X}_{training}$, with the 50% of the samples, and a validation set $\mathcal{X}_{validation}$ that contains the remaining elements. The objective of the training set is to fit the function f solving the optimization problem showed in Equation 1.1, on the other hand, the validation set aims to avoid the problem of overfitting. This is related to the stopping criterion, which is explained below.

EvoDAG searches a function f that optimizes Equation 1.1 in a search space, Ω , that is formed by all the combinations of elements in the function \mathcal{F} and terminal \mathcal{T} sets. It is not possible to know and test all the elements in Ω . For that reason, a sample of Ω is tested using GP to enhance the search.

Based on [35, 37], each function in the function set \mathcal{F} is associated with a set of parameters θ that are estimated with ordinary least squares (OLS) using the target and the individual' semantics calculated over the training set $\mathcal{X}_{training}$. The aim is minimizing the difference between the individual's semantics and the target semantics. The background of the use of parameters θ is described in Section 2.2. The addition is defined as $\sum_k \theta_k p_k$, where p_k is an individual from population \mathcal{P} . The classifiers are defined as $NB(\theta_1 p_1, \dots, \theta_k p_k)$, $MN(\theta_1 p_1, \dots, \theta_k p_k)$, and $NC(\theta_1 p_1, \dots, \theta_k p_k)$. The rest of the arithmetic functions, trigonometric functions, min and max are defined as $\theta f(p_1, \dots, p_k)$ where f is the function at hand, and p_1, \dots, p_k are individuals in \mathcal{P} .

Specifically, in classification, the initial population \mathcal{P}_0 contains the classifiers Naive Bayes with Gaussian distribution (NB), Naive Bayes with Multinomial distribution (MN), and Nearest Centroid (NC). Those classifiers take as arguments all the inputs $\{\text{NB}(\theta_1 x_1, \dots, \theta_m x_m), \text{MN}(\theta_1 x_1, \dots, \theta_m x_m), \text{NC}(\theta_1 x_1, \dots, \theta_m x_m)\}$. In addition, \mathcal{P}_0 contains all the inputs $\{\theta x | x \in \mathcal{T}\}$. In case that the number of elements in \mathcal{P}_0 is lower than the population size, other elements are added using the following procedure. A function f_k is randomly selected from the function set \mathcal{F} , and its k arguments are randomly chosen from the terminal set \mathcal{T} without replacement. This process continues until the population size has been reached. For example, let $\mathcal{F} = \{\text{NB}, +, | \cdot |\}$ and $\mathcal{T} = \{x_1, x_2, x_3\}$, then \mathcal{P}_0 starts with $\{\text{NB}(\theta_1 x_1, \theta_2 x_2, \theta_3 x_3), \theta_4 x_1, \theta_5 x_2, \theta_6 x_3\}$. This is followed by randomly selecting a function and its parameters from \mathcal{T} , assuming $+$ is selected, then $\mathcal{P}_0 = \{\text{NB}(\theta_1 x_1, \theta_2 x_2, \theta_3 x_3), \theta_4 x_1, \theta_5 x_2, \theta_6 x_3, \theta_7 x_3 + \theta_8 x_1\}$, assuming the next function is $| \cdot |$, then, $\mathcal{P}_0 = \{\text{NB}(\theta_1 x_1, \theta_2 x_2, \theta_3 x_3), \theta_4 x_1, \theta_5 x_2, \theta_6 x_3, \theta_7 x_1 + \theta_8 x_3, \theta_9 | x_2 |\}$. This process is repeated until the number of elements in \mathcal{P}_0 reaches the population size. All the parameters θ are estimated with ordinary least squares (OLS) using the target and the individual' semantics calculated over the training set $\mathcal{X}_{training}$.

Once the initial population \mathcal{P}_0 is created, the evolution starts using the steady-state population model. Consequently, at the beginning, $\mathcal{P} = \mathcal{P}_0$, and in each iteration, \mathcal{P} is updated by replacing a current individual, selected using a negative selection, with an offspring that can be selected as a parent just after being inserted in \mathcal{P} (see Algorithm 2). The offspring creation process is similar to the one used to create the initial population, but the difference is on the procedure used to select the arguments. That is, a function f_k is randomly selected from \mathcal{F} , k represents the number of arguments, which are selected from the population \mathcal{P} using tournament selection or any of the heuristics analyzed in this dissertation (see Chapter 3). The parameters θ associated to f are optimized using OLS.

The process continues until the stopping criteria are met. Traditionally in EAs, the evolution stops when a maximum number of generations is reached, or the fitness

reaches a particular value. In this case, EvoDAG stops the evolutionary process using early stopping. The validation set $\mathcal{X}_{validation}$ is used to perform the early stopping and to keep the individual with the best performance. The evolution stops when the fittest individual, on the validation set $\mathcal{X}_{validation}$, has not been updated in a defined number of evaluations (see Algorithm 2). The final model corresponds to the fittest individual on the validation set.

Algorithm 2: Evolution process of EvoDAG. The underlined steps correspond to the ones that are analyzed in this dissertation.

Input: The training set, \mathcal{X}

Input: The fitness function, $fit(individual, \mathcal{X})$

Output: An EvoDAG's model that represents the function f that optimizes Equation 1.1

$\mathcal{X}_{training}, \mathcal{X}_{validation} \leftarrow$ randomly division of \mathcal{X} ;

$\mathcal{P}_0 \leftarrow$ the initial population ;

$p_{validation} \leftarrow$ the fittest individual based on $\mathcal{X}_{validation}$;

$rounds \leftarrow 0$;

while $rounds < MaxEarlyStoppingRounds$ **do**

$f_k \leftarrow$ A function that is randomly selected from the function set \mathcal{F} ;

for i **from** 1 **to** k **do**

$parent_i \leftarrow$ An individual selected from the population \mathcal{P} for being a parent;

Add $parent_i$ as argument of f_k ;

end

$\theta \leftarrow$ The parameters of f_k that are calculated using OLS and $\mathcal{X}_{training}$;

$new \leftarrow$ The new offspring created using f_k and its arguments;

$old \leftarrow$ An individual selected from the population \mathcal{P} using negative selection;

Delete old from the population \mathcal{P} ;

Add new to the population \mathcal{P} ;

if $fit(new, \mathcal{X}_{validation}) > fit(p_{validation}, \mathcal{X}_{validation})$ **then**

$p_{validation} \leftarrow new$;

$rounds \leftarrow 0$;

else

$rounds \leftarrow rounds + 1$;

end

end

Return $p_{validation}$;

Implementation

EvoDAG's implementation is based on the proposal of Castelli, Vanneschi, and Silva [10, 100]. Only the semantics of individuals in the current population are stored, but the trace of all individuals is stored.

We said that the initial population is created using the inputs of the problem and the classifiers. Based on the same example, where $\mathcal{F} = \{\text{NB}, +, | \cdot |\}$ and $\mathcal{T} = \{x_1, x_2, x_3\}$, the first individuals from the initial population \mathcal{P}_0 are $\{\text{NB}(\theta_1 x_1, \theta_2 x_2, \theta_3 x_3), \theta_4 x_1, \theta_5 x_2, \theta_6 x_3\}$. The idea of using as first individuals the inputs and classifiers is because by doing this, it is simpler to calculate their semantics, only it is needed to compute the parameters θ and the classifiers' outputs. For creating each one of the remaining individuals in \mathcal{P}_0 , the following process is executed. A function f_k is randomly selected from the function set \mathcal{F} , and its k arguments are randomly chosen from the terminal set \mathcal{T} without replacement. For all individuals in \mathcal{P}_0 the semantics vectors are stored in memory. Assuming $\mathcal{P}_0 = \{\text{NB}(\theta_1 x_1, \theta_2 x_2, \theta_3 x_3), \theta_4 x_1, \theta_5 x_2, \theta_6 x_3, \theta_7 x_1 + \theta_8 x_3, \theta_9 |x_2|, \dots\}$, a representation of individuals and their semantics can be seen in Figure 1.8.

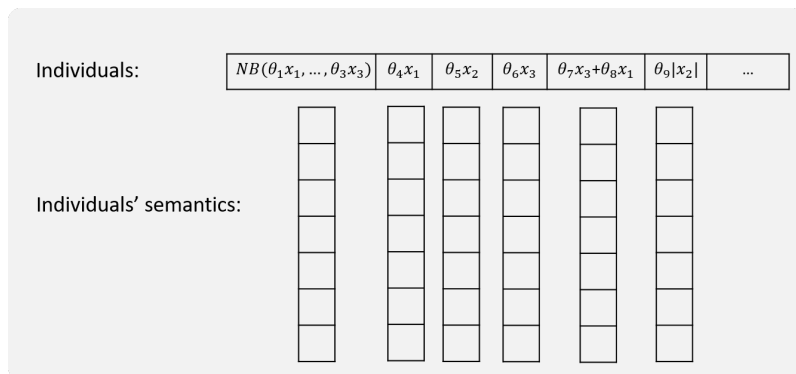


Figure 1.8: Example of EvoDAG's individuals and their semantics in the initial population \mathcal{P}_0 . Source: Own elaboration.

Once the initial population is created, in each iteration, one offspring is generated, and it replaces another one from the population. Figure 1.9 shows an example where the blue and the red individuals are added and removed from \mathcal{P} , respectively. As it was explained above, for creating an offspring a function f_k is chosen from \mathcal{F} , and

its k arguments are selected from the population \mathcal{P} , then, the parameters θ associated to f_k are optimized using OLS. EvoDAG calculates the semantics of offspring based on the application of their root function and the semantics of their parents or arguments. It is not needed to evaluate the whole tree structure. Having the semantics of parents, it is only necessary to apply the function that the offspring has in its root node. It is represented by the arrows in Figure 1.9. Also, when an offspring is created, its semantics is stored. For avoiding memory problems, the individual's semantics that is removed from \mathcal{P} is deleted. Indeed, the new offspring semantics is stored in the memory space of the individual's semantics that will be removed. Then, the memory used is always constant, and it depends on the training samples and the population size.

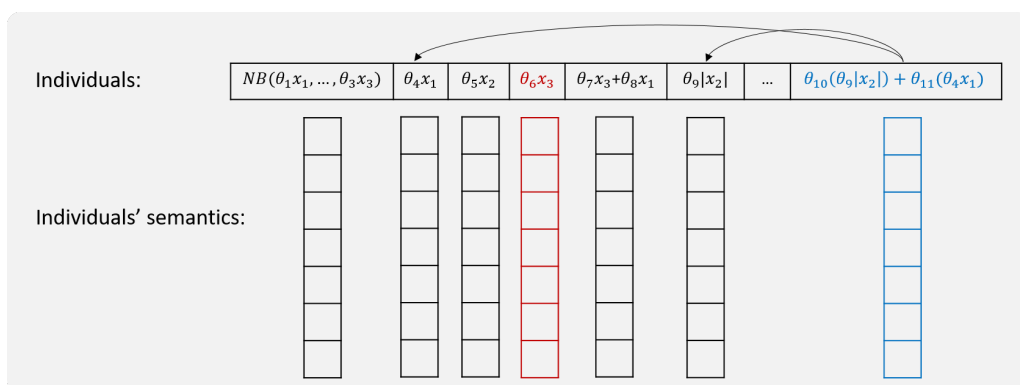


Figure 1.9: Example of EvoDAG's individuals and their semantics when a new offspring is created. Source: Own elaboration.

Representation of Classification Problems

Specifically, for solving classification problems, EvoDAG uses the one-vs-all scheme. We affirmed that in a supervised learning problem, the dataset is composed of tuples of data samples $(x^{(i)}, t^{(i)})$, where $x^{(i)}$ represents the input and $t^{(i)}$ the output, which corresponds to a categorical value in a classification problem. Then, for transforming the classification problem in k different ones, where k is the class number, the output is represented as a vector $\vec{t}^{(i)}$ with k entries, where we assign 1 to the entry that corresponds to the real class and -1 to the other classes.

Instead of evolving one tree per problem, as done, for example, in [69], Graff *et al.* decided to use only one tree and optimize k different θ parameters, one for each class. The result is that each node outputs k values, and the class is the one with the highest value. In the case of the classifiers, Naive Bayes and Nearest Centroid, the output is the log-likelihood.

This representation also changes the semantics of individuals. As we explained in Section 1.3, the individual's semantics is the vector whose entries are all the responses of the individual's function to all the input-vectors. In this sense, the semantics of the individual P corresponds to a vector $\vec{S}_p \in \mathbb{R}^n$, where n represents the number of input-vectors. However, in EvoDAG, each individual contains an array of semantics vectors, one per class, where each vector contains the semantics for a specific class.

Ensemble

It is well known that in evolutionary algorithms, there are runs that do not produce an acceptable result. Then, to improve the stability and also the performance of EvoDAG, Graff *et al.* decided to use Bagging [6] in their approach.

Thirty models are fitted, each one uses only half training samples. The samples are randomly selected, and different seeds are used in the random function. A bagging estimator can be expected to perform similarly by either drawing n elements from the training set with-replacement or selecting $\frac{n}{2}$ elements without-replacement (see [26]). The ensemble prediction corresponds to the average of the thirty models. In Chapter 4, an analysis of the number of models in the ensemble is presented.

Optimization of Parameters

EvoDAG, as all evolutive algorithms, has some parameters that need to be defined. EvoDAG's authors decided to use the technique random search for the optimization

of hyper-parameters [5], proposed by Bergstra and Bengio.

For adjusting the parameters' values, they used thirteen binary classification datasets that were divided into the training and test sets. Those datasets are described in [36]. First, they defined possible values for each parameter (see Table 1.2), including the functions' number of arguments (see Table 1.3). Then, they tasted different combinations of parameters' values. They used *Balanced Error Rate* (BER) as performance metric of EvoDAG's classifiers. $BER = \frac{100}{K} \sum_{i=1}^K \frac{fn_i}{t_i}$, where K is the number of classes, i represents the i -th class, fn_i is the number of false negative of class i , and finally, t_i is the number of elements of class i . For each dataset, they executed 734 experiments where the parameters' values were randomly selected from the ones presented in Tables 1.2 and 1.3. In each experiment, they executed EvoDAG three times and calculated the median of BER. For each dataset, the best parameters correspond to the ones with the smallest median BER. Finally, for having a generalization of parameters that work well in several datasets, they calculated the parameters' values as the average of the best values of the thirteen datasets. Based on that procedure, they defined the EvoDAG's parameters as the ones presented in Table 1.4.

Table 1.2: The possible values of EvoDAG's parameters

Parameter	Possible values
Population size	100, 200, 500, 1000, 2000, 4000
Tournament size	2, 5, 10, 20
Early stopping rounds	500, 1000, 2000, 4000

1.5 Summary

In this Chapter, we introduced the key concepts of Evolutionary computing (EC), Genetic Programming (GP), Semantic Genetic Programming (SGP), and EvoDAG, the GP system that we used for implementing and testing our proposed selection heuristics.

Evolutionary computing (EC) is a research area within computer science, as the name suggests, it is a particular flavor of computing, which draws inspiration from the

Table 1.3: The possible number of arguments of EvoDAG’s functions. When the number of arguments is equal to zero, it indicates that the function is not included in the function set.

Function	Possible number of arguments	Function	Possible number of arguments
Σ	2, 5, 10, 20, 30, 40, 50, 60, 70	sin	0, 1
Π	0, 2, 5, 10, 20, 40	cos	0, 1
max	0, 2, 5, 10, 20, 40	tan	0, 1
min	0, 2, 5, 10, 20, 40	asin	0, 1
NB	0, 2, 5, 10, 20, 40	acos	0, 1
MN	0, 2, 5, 10, 20, 40	atan	0, 1
NC	0, 2, 5, 10, 20, 40	tanh	0, 1
$\sqrt{\cdot}$	0, 1	atan2	0, 2
$ \cdot $	0, 1	hypot	0, 2

Table 1.4: EvoDAG’s parameters. The functions’ subscripts represent the number of arguments.


Parameter	Value
Population size	4000
Tournament size	2
Stop criterion	Early stopping, 4000 rounds
Function set \mathcal{F}	$\Sigma_{60}, \Pi_{60}, \max_5, \min_5, \text{NB}_5, \text{MN}_5, \text{NC}_5, \sin, \tan, \text{atan}, \tanh, \sqrt{\cdot}, \cdot , \text{hypot}_2$
Number of models in the Ensemble	30

process of natural evolution. The key aspects to take into account when the implementation of an evolutive algorithm is performed are the representation, fitness function, selection, and variation operators: crossover and mutation.

GP evolves a population of computer programs and can be used for solving a variety of problems. While evolutive algorithms are typically applied to solve optimization problems, GP could instead be positioned in machine learning. The *semantics* of an individual is a vector whose entries are all the responses to the input vectors. The term Semantic Genetic Programming (SGP) is used when GP uses semantics in the evolutionary process.

EvoDAG, a GP system proposed by Graff *et al.* [36], is a python library that solves supervised learning problems. EvoDAG searches a model combining elements from

the terminal set \mathcal{T} and the function set \mathcal{F} . The terminal set \mathcal{T} is composed only by the input variables of the problem; this is $\mathcal{T} = \{x_1, \dots, x_m\}$. The function set for classification problems is defined as $\mathcal{F} = \{\sum_{60}, \prod_{60}, \max_5, \min_5, \text{NB}_5, \text{MN}_5, \text{NC}_5, \sin, \tan, \text{atan}, \tanh, \sqrt{\cdot}, |\cdot|, \text{hypot}_2\}$, where the subscript indicates the number of arguments. We used EvoDAG for implementing and testing the proposed selection heuristics, in addition to classical and state-of-the-art selection techniques.



Chapter 2
Related Work

2 Related Work

This chapter presents the literature review related to the topic of this dissertation. In Chapter 1, we described Semantic Genetic Programming, Vanneschi divided the work that is related to semantics in indirect and direct [99]. The indirect methods are the ones that apply operators or methodologies on the trees' structures and use semantics for validating the results. The most representative work of indirect methods is described in the first section of this chapter. Section 2.2 presents the methodologies that work directly in the semantic space. This approach is more recent than indirect methodologies. According to Vanneschi [99], these techniques can improve the learning process and be executed efficiently. A review of how fitness and selection have been implemented in GP is presented in Section 2.3. This dissertation has as objective apply GP for solving classification problems, Section 2.4 presents the work that also use GP in classification.

2.1 Indirect Semantic Genetic Programming

In Chapter 1, we explained individuals' semantics and how the supervised learning problems can be seen from the point of view of Semantic Genetic Programming (SGP). The use of individuals' semantics and also target's semantics is very useful for designing operators and GP systems. Krawiec [52] affirmed that aware semantic methods make search algorithms better informed. In this section, we review some work that uses SGP for improving GP in supervised learning problems.

Bryan proposed in [8] a technique that calculates the canonical representation of syntax trees of GP individuals for binary problems using truth tables. Therefore, individuals with similar semantics have the same representation. Beadle and Johnson [3] used that representation to propose a crossover operator that measures the semantic equivalence between parents and their offspring. Each time the crossover operator is applied, they reject the offspring that is semantically equivalent to their parents and

repeat the crossover. They also proposed a mutation operator [4] that measures the individual's changes after mutation, and it is allowed only if the individual is semantically different after the mutation. The result of these operators is the increment of population diversity, and as a consequence, the improvement of the GP performance for binary problems.

Nguyen *et al.* [73] proposed two distance metrics between individuals, one based on the syntax tree structure, and the other using individuals' semantics. In the first one, the same tree structure is constructed adding nodes if necessary. Then, the distance between the nodes in the same position is calculated, if the nodes have the same symbol, the distance is 0, otherwise it is 1. Finally, the distance between individuals is the sum of all node distances. The second distance is calculated base on the *Sampling Semantics (SS)* of individuals and it is called *Sampling Semantics Distance (SSD)*. They defined the Sampling Semantics of an individual T as $S_T = \{s_1, s_2, \dots, s_n\}$, and it is calculated by the evaluation of the function f that the individual T represents in a point series $P = \{p_1, p_2, \dots, p_n\}$, so $s_i = f(p_i)$, $i = 1, 2, \dots, n$. The SSD between two individuals is defined as Equation 2.1, where T_1 and T_2 are the individuals, and $\{u_i\}_{i=1,2,\dots,n}$ y $\{v_i\}_{i=1,2,\dots,n}$ are the sampling semantics of each individual.

$$SSD(T_1, T_2) = (|u_1 - v_1| + |u_2 - v_2| + \dots + |u_n - v_n|) / n \quad (2.1)$$

Quang Uy *et al.* used the Sampling Semantics Distance to create the *Semantic Similarity based Crossover* operator [96] and the *Semantic Similarity based Mutation* operator [83]. First, they defined the concepts of *Semantically Equivalent (SE)* and *Semantic Similarity (SSi)* between two individuals T_1 and T_2 , whose sampling semantics are St_1 and St_2 respectively, as follows. The individuals are SE if their sampling semantics distance, SSD, is less than a threshold. It is formally defined in Equation 2.2, where ϵ is a predefined constant. On the other hand, the individuals are SSi if their sampling semantics distance is between a range of values, in other words, it is small but no too

much. It is formally defined in Equation 2.3, where α and β are predefined constants. The crossover operator searches for a crossover point in each parent in such a way that the subtrees are semantically different but not too much, it means, the subtrees need to be semantically similar. The mutation operator allows replacing a subtree of the individual only if the new subtree is semantically similar to the one being replaced. Using these operators, they could control the changes in individual fitness.

$$SE(T_1, T_2) = \text{true if } SSD(St_1, St_2) < \epsilon, \text{ else false} \quad (2.2)$$

$$SSi(St_1, St_2) = \text{true if } \alpha < SSD(St_1, St_2) < \beta, \text{ else false} \quad (2.3)$$

Semantic Control Crossover [40], introduced by Hara *et al.*, used semantic similarity between subtrees to control the global and local search. The operator combines the individuals A and B as follows. First, a point crossover is randomly selected from A , and the semantics of the subtree at that point is calculated. Then, the semantics of all subtrees in B are calculated, and the semantic distances between all the subtrees in B and the subtree selected in A are calculated. If it is a regression problem, they use Euclidean distance for measuring semantic similarity, and if it is a Boolean problem, the Hamming distance is used. The selection of the crossover point in B uses roulette selection based on the semantic distance d power to α , this is, d^α , where α changes at each iteration based on the Equation 2.4. α_{max} and α_{min} are the minimum and maximum values of α respectively (in the experiments, the values 2 and -2 were used), G is the maximum number of generations, and t is the current generation. Their results were that a global search is performed at the beginning of the process and local search in the last generations. In other words, in the beginning, the crossover between trees with different semantics is prioritized, while late in the process, the opposite is promoted, i.e., crossover between trees with similar semantics.

$$\alpha(t) = \alpha_{max} - \frac{t(\alpha_{max} - \alpha_{min})}{G} \quad (2.4)$$

Graff *et al.* proposed the *Semantic Crossover Operator based on the Partial Derivative Error* [30]. The operator combines two individuals as follows. First, it randomly selects a crossover point in the first parent. Based on the back-propagation algorithm, the partial derivative of the fitness function in the crossover point is calculated with the aim of knowing whether the values in the node need to be higher or lower than the current ones. Then, a subtree in the second parent is searched, in such a way that when it is used as a crossover point increases or decreases the value of the function. Graff *et al.* also proposed the *Semantic Point Mutation operator based on the Partial Derivative Error* [29]. This operator randomly selects a mutation point in the individual tree with the restriction that it must be a function node. The partial derivative of the fitness function in that point is calculated in order to know whether the values in it need to be higher or lower than the current ones. Finally, a function that increases or decreases the value in the selected point, and with the same arity is searched in the function set. Furthermore, to promote the use of their operators, they proposed differential functions that simulate the behavior of not differential functions like if, max, min y argmax. Following the same direction, Suárez *et al.* proposed the *Semantic Crossover operator based on the Second Partial Derivative of the Error Function* [92]. The objective of the second partial derivative is to calculate the optimum value of the node using the Newton method (see Equation 2.5). The process is very similar. First, a crossover point is randomly selected. The first and second partial derivatives are calculated. Finally, the crossover point in the second parent whose semantics are most similar to the calculated values with the Newton method is selected. Their results showed that the use of these operators improves the performance of GP for symbolic regression problems.

$$X^{n+1} = X^n - \frac{f'(X^n)}{f''(X^n)} \quad (2.5)$$

2.2 Direct Semantic Genetic Programming

Vanneschi affirmed in [99] that the main problem of methods that use the semantics in an indirectly way, as the operators presented in the previous section, is related with the rejection of children during the evolutive process. Typically, the individuals are constructed using the operators based on syntactic operations, and the semantics criteria require individual evaluation. It can affect the learning speed of GP. Nevertheless, the operators that use the semantic information in a directed way, like the ones presented in this section, never reject individuals once they are evaluated, in this sense, they can be executed efficiently. Moreover, Vanneschi said that indirect methods have been widely studied, so it could be said it is a mature area. On the other hand, the directed methods are more recent, and much of its potential is to be seen.

One of the first geometric semantic crossover operator was proposed by Krawiec and Lichocki in [53]. The operator tries to generate offspring whose semantics is similar to a linear combination of their parents. It uses another crossover operator, which can be the classic one described in Section 1.4. Given two parents P_1 and P_2 , the crossover between them is applied k times, and all the possible children are stored. The expression in Equation 2.6 is calculated for each offspring O , where $d(\cdot, \cdot)$ represents a distance between the individuals' semantics. The two children with the smaller values are selected as offspring. The idea is to generate offspring whose semantics are close to their parents' semantics, but also, they promoted equidistance between parents and offspring semantics.

$$d(P_1, O) + d(P_2, O) + |d(P_1, O) - d(P_2, O)| \quad (2.6)$$

Krawiec extended his work in [51]. He defined the *geometric crossover operators* as operators that generate offspring O given two parents P_1 and P_2 with the charac-

teristic showed in Equation 2.7, where $d(\cdot, \cdot)$ is a distance measure. In addition, he defined the *medial crossover operators* as the ones that minimize the divergence functions showed in Equation 2.8 and 2.9. In the same work, Krawiec proposed the crossover operator PMX which considers some subtrees interchanges between parents and selects the interchange that minimizes the divergence functions d_G and d_E .

$$d(O, P_1) + d(O, P_2) = d(P_1, P_2) \quad (2.7)$$

$$d_G(o, p_1, p_2) = d(o, p_1) + d(o, p_2) - d(p_1, p_2) \quad (2.8)$$

$$d_E(o, p_1, p_2) = \left| d(o, p_1) - d(o, p_2) \right| \quad (2.9)$$

Moraglio *et al.* [67,68] proposed *Geometric Semantic Genetic Programming (GSGP)*. They proposed two operators, crossover and mutation. Given two parents, P_1 and P_2 , the crossover operator generates an offspring as Equation 2.10, where r is a value between 0 and 1. The mutation operator for P was defined as Equation 2.11, where R_1 and R_2 are random trees, and ms is a real value within the range $[0, 1]$. Their work called the attention of the GP scientific community because the crossover operator produces an offspring that stands in the segment joining the parents' semantics. Therefore, offspring fitness cannot be worse than the worst fitness of the parents, and this property transforms the fitness landscape into a cone. Unfortunately, the offspring is always bigger than the sum of the size of its parents. It makes the use of this operator impossible in practice, and it compromises the GP essence: the ability to construct legible solutions and interpretable by humans [99]. Figure 2.1 shows an example of the GSGP crossover operator.

$$r \cdot P_1 + (1 - r) \cdot P_2 \quad (2.10)$$

$$P + ms \cdot (R_1 - R_2) \quad (2.11)$$

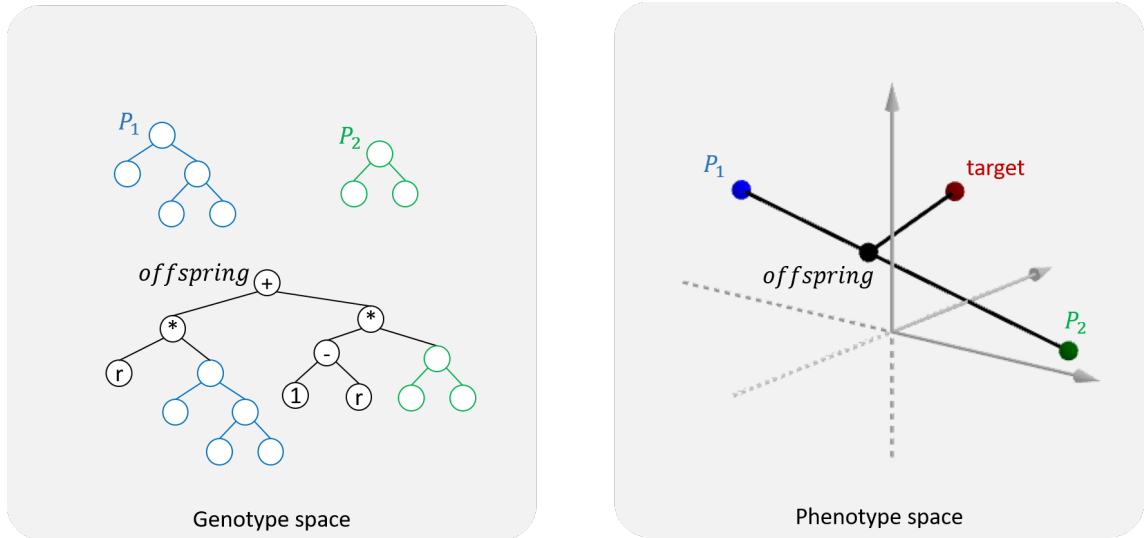


Figure 2.1: Illustration of Geometric Semantic Genetic Programming crossover operator. Source: Own elaboration.

Krawiec and Pawlak proposed *Locally geometric semantic crossover (LGX)* [54, 55, 58]. It is an operator that approximates the crossover operator of GSGP but with the idea of reducing bloat. LGX first identifies the structurally common region [81] in the parents P_1 and P_2 , this is, the set of tree node locations that occur in both parents. Then, in each parent, a crossover point is randomly selected from that region, prioritizing intern nodes. The semantics of the subtrees p'_1 and p'_2 rooted in the crossover points are calculated. The midpoint between those subtrees in the semantic space also is calculated as $s_m = \frac{s(p'_1) + s(p'_2)}{2}$. s_m represents the program that is perfectly geometric to p'_1 and p'_2 . Finally, the procedure p' whose semantics $s(p')$ is the closest to s_m is searched in a library of procedures \mathcal{L} , it means, $p' = \operatorname{argmin}_{p' \in \mathcal{L}} \|s(p') - s_m\|$. After some analysis, the authors noticed the election of p' in a deterministic way causes premature convergence in the searching process, so, they decided to randomly choose p' of the k procedures whose semantics are closer to s_m . For searching the k nearest neighbors, they used the data structured *kd-trees*.

Krawiec, Pawlak, and Wieloch introduced *Approximating geometric crossover (AGX)* [56, 79]. AGX aims to replace subtrees in individuals for generating offsprings whose semantics are close to the center of the segment joining the parents' semantics

using the backpropagation algorithm. Given two parents, P_1 and P_2 , AGX calculates the parents' semantics $s(p_1)$ and $s(p_2)$. The point in the center of the segment joining $s(p_1)$ and $s(p_2)$ is calculated as $m = \frac{s(p_1)+s(p_2)}{2}$. A crossover point is randomly selected in both parents. Then, in an independent way for each parent, the backpropagation algorithm is used to propagate the expected semantics m and calculate the desired semantics in the crossover point. Finally, from a library of procedures \mathcal{L} , the ones whose semantics are the closets to the desired semantics are selected, and they replace the subtrees in the crossover points to generate the offsprings.

Pawlak *et al.* affirmed in [78] that the use of crossover and mutation operators focusing on the syntax trees is a complex process because a small change in the individual syntax can result in a dramatic change in its response. On the other hand, a big change in the syntax cannot affect the response. Nevertheless, Semantic Genetic Programming changes this because it gives information about the individuals' behavior. Furthermore, Geometric Semantic Genetic Programming not only gives information about the individuals' behavior but also gives a multidimensional metric space (the semantic space), which is unimodal. However, despite this attractive property, the searching process is not easy because the semantic space is not the searching space; the searching space is the genotypic space. In the same work, they compared the Moraglio's crossover operator SGX [67, 68] with their operators LGX [54, 55, 58] and AGX [56, 79]. Of the comparison, they concluded that although SGX is attractive for their potentiality to find a perfect solution in a short time, it has the inconvenient of generating huge trees. While on the other hand, LGX and AGX produce smaller trees with a good generalization capacity.

Pawlak *et al.* proposed the *Random Desired Operator (RDO)* [79]. It is a mutation operator that aims to propagate the target semantics. RDO first randomly selects the mutation node, then a backpropagation algorithm calculates the desired semantics in that node as follows. The propagation starts in the root node with the target semantics. The inverse of the function is used for calculating the desired semantics in the child

node. Then, the algorithm moves to that child, and the propagation is also applied there. This process is repeated until the desired semantics for the node of interest is calculated (see Figure 2.2). Once the desired semantics is calculated in the mutation node, a procedure whose semantics are the closest to the desired semantics is searched in a procedure library \mathcal{L} . Finally, the found procedure replaces the subtree positioned in the mutation node. Virgolin et al. proposed in [101] a linear scaling for RDO. Let the MSE between the dependent variable y and the tree output o be the fitness function for regression (see Equation 2.12), they introduced a scaled version of MSE incorporating the scalar values a and b as Equation 2.13.

$$MSE(y, o) = \frac{1}{n} \sum_i (y_i - o_i)^2 \quad (2.12)$$

$$MSE(y, o) = \frac{1}{n} \sum_i (y_i - (a + bo_i))^2 \quad (2.13)$$

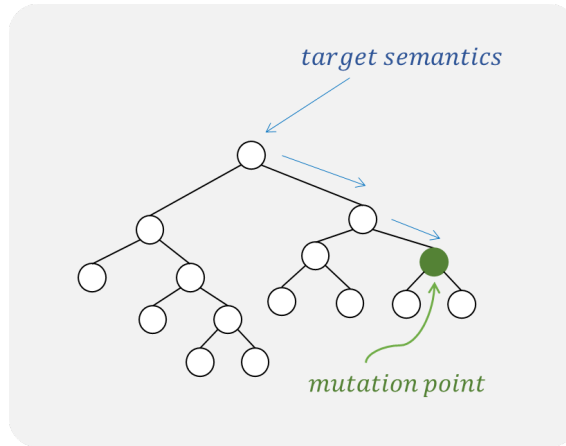


Figure 2.2: Illustration of Random Desired Operator (RDO). Source: Own elaboration.

In [74], Nguyen *et al.* compared the operators: SSGX [74], RDO [79], AGX [57], and Moraglio's geometric semantic crossover operator (SGX) [67]. Based on their experiments, they concluded that: (1) RDO was the best for regression problems and SSGX the second better; (2) AGX, RDO and SSGX had smaller solutions than SGX; and (3) AGX and RDO had an execution time higher than SSGX. They were usually 8 or 10

times slower than SSGX.

RDO was extended by Szubert *et al.* when *Forward Propagation Mutation (FPM)* [95] was introduced. FPM uses a combination of forward and back-propagation to find a combination of unitary and binary functions that are the most similar to the desired behavior. First, it randomly selects a node in the tree and separates it into a subtree p_1 positioned in the selected node and a context p_2 . The subtree p_1 is extracted from the tree and used as the initial point from the new individual. For generating the new context, it assumes a structure with four new nodes: (1) u that represents a unitary function as sin, cos, log or exp; (2) b that is a binary function as +, -, * or /; (3) x that represents a multiplication; and finally, (4) c that is a real value. Figure 2.3 shows FPM structure. An exhaustive search of all possible combinations between unitary and binary functions (u, b) is performed. For each pair of functions (u, b), the semantics of the subtree p_1 is forward propagated to the root node. Then, the target semantics and the invert function of b are used for calculating the desired semantics d for the node x . A tree whose semantics is the closets to d is searched in a procedure library \mathcal{L} , this is: $p_i^* = \operatorname{argmin}_{p_i} \arccos \frac{s(p_i)d}{\|s(p_i)\| \|d\|}$. The real value c is used to scale the semantics of p_i^* to bring it closer to d . Finally, the combination (u, b) whose individual has better fitness is selected.

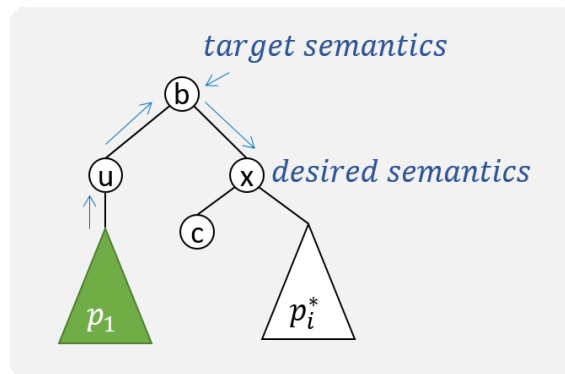


Figure 2.3: Illustration of Forward Propagation Mutation (FPM). Source: Own elaboration.

Graff *et al.* proposed a new crossover operator based on projections in the phe-

notype space [37]. It creates a plane in the semantic space using the parent semantics and the origin of the space. The offspring is calculated as the projection of the target in that plane. Given the parents semantics P_1 and P_2 , and the target semantics T , the offspring is calculated as Equation 2.14 where α and β are real values that are calculated solving the equation $A[\alpha, \beta]' = T$ where $A = (P_1, P_2)$. It implies the offspring will be at least as good as the best parent. An application is presented in [34], where the objective is determined whether a text has a positive, negative, or neutral opinion according to a specific topic. Figure 2.4 shows the comparison between Moraglio's and Graff's crossover operators in the semantic space.

$$O = \alpha P_1 + \beta P_2 \quad (2.14)$$

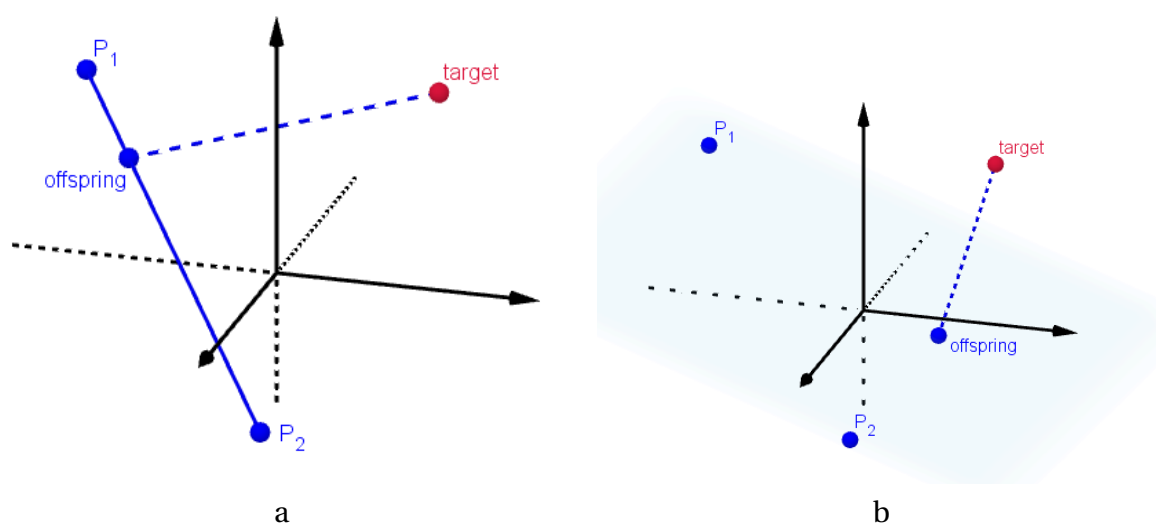


Figure 2.4: Comparison of Moraglio's and Graff's crossover operators. a) Moraglio's geometric crossover. b) Graff's projection crossover. Source: Own elaboration.

Memetic Genetic Programming based on orthogonal projections in the phenotype space [35] was also proposed by Graff *et al.* In that work, they used a linear combination of k parents as $\sum_k \alpha_k P_k$, with the idea of optimizing the coefficients $\{\alpha_i\}$ with ordinary least squares (OLS) to guarantee the offspring is the best of its family. As a result, the fitness of the generated tree is always *better* or equal than any internal tree.

It was not the first time where parameters are added to GP nodes, Smart and Zhang defined the *Inclusion Factors* [90] as numeric values between 0 and 1 assigned to each node in the tree structure, except the root node. This value represents the inclusion proportion of the node in the tree. In their work, they only used the sum and multiplication functions. The sum function without inclusion factors can be seen as $a_1 + a_2$ and with inclusion factors as $x_1 a_1 + x_2 a_2$. For calculating the inclusion factors, they used gradient descent applied to the cost function $C = \frac{\sum_{j=1}^N (y_j - Y_j)^2}{2}$, where Y_j and y_i are the desired and the actual outputs for the training sample j , and N is the number of training samples.

Castelli *et al.* presented in their work *Geometric Semantic Genetic Programming with Local Search* [12] a mutation operator, based on Moraglio's mutation operator, that also uses parameters. In this operator, an individual P is modified with the Equation 2.15, where R_1 and R_2 are random trees, and, $\alpha_i \in \mathbb{R}$. $\{\alpha_i\}$ are calculated using the target semantics and OLS for getting the better linear combination of the original and the random trees. Castelli *et al.* extended their work in [9] applying Local Search to all the individuals during a separate step after mutation and crossover. For a GP tree T , they calculated another tree as $T' = \alpha T + \beta$, where α and β are optimized with OLS minimizing the error between the individual semantics and target semantics. Moreover, they generalized the idea and transformed it into a regression problem $T' = \sum_j \alpha_j f_j(T)$, where $f_j : \mathbb{R} \rightarrow \mathbb{R}$.

$$O = \alpha_0 + \alpha_1 P + \alpha_2 (R_1 - R_2) \quad (2.15)$$

Nguyen *et al.* proposed the *Subtree Semantic Geometric Crossover (SSGX)* [74] operator, which uses the semantic similarity between subtrees to approximate the geometric property of the Moraglio's operators for producing smaller individuals' trees. The operator works as follows. First two parents P_1 and P_2 , and a probability value ϵ are selected. Whether a random number $r \in [0, 1]$ is smaller than ϵ , the new subtree geometric operator is used; if not, the traditional operator is applied. In the case of the new

operator is selected, several subtrees which satisfy a defined size are randomly selected from P_1 , except the whole tree P_1 . The selected subtree St_1 is the one whose semantics are the closest to P_1 semantics. A subtree St_2 is selected from P_2 following the same process. The offspring O_1 and O_2 are generated by combining subtrees St_1 and St_2 as Equations 2.16 and 2.17, where T_R is a real value between 0 and 1 or a random tree with co-domain $[0, 1]$. The size constant helps to control the size of individuals, and it needs to be inside of the established range $[\alpha, \beta]$. They conclude that the use of this operator approximates the geometric property at a subtree level, and it helps to reduce the exponential growth of individuals.

$$O_1 = T_R St_1 + (1 - T_R) St_2 \quad (2.16)$$

$$O_2 = (1 - T_R) St_1 + T_R St_2 \quad (2.17)$$

Hara *et al.* proposed *Deterministic Geometric Semantic Crossover* [39]. It is based on the Moraglio's geometric semantic crossover operator. Given the parents' semantics P_1 and P_2 , and the target semantics T , they calculate a new offspring O based on the geometric crossover operator defined in Equation 2.10. But, instead of calculating r as a random number between $[0, 1]$, they calculate it as $r = \frac{|P_1 P_2| - |P_1 T| \cos \theta}{|P_1 P_2|}$, where θ is the angle between the semantics of the target and the second parent using the first parent' semantics as reference, this is, $\theta = \angle T P_1 P_2$. In other words, the offspring is the projection of the target in the line formed by the parents. This operator used the target semantics effectively to determine the optimal combination of parents, and as a result, the performance of GSGP is improved. However, the problem of the bloat phenomenon continues.

Chen *et al.* proposed *Angle-Driven Geometric Semantic Genetic Programming (ADGSGP)* [14–16]. Their work attempts to further explore the geometry of geometric operators in the search space to gain an improvement in genetic programming for

symbolic regression. The angle-awareness brings new geometric properties that are expected to provide greater leverage for approximating the target semantics in each operation, and more importantly, be resistant to overfitting. The angle-distance between the semantics of two individuals is calculated as the angle γ between the two semantics' vectors \vec{V}_1 and \vec{V}_2 , Equation 2.18, where $\|\cdot\|$ represents the vector norm. However, they used the angle between the relative vectors. A relative vector is the one between one parent's semantics and the target semantics (see Equation 2.19). Based on this, they proposed Angle-Driven Selection, Perpendicular Crossover, and Random Segment Mutation. *Angle-Driven Selection (ADS)* selects a pair of parents that not only have good fitness values but also are far away from each other regarding the angle-distance of their relative semantics. It is described in detail in Section 2.3. *Perpendicular Crossover (PC)* generates a child point standing on the line crossing the two parents, which follows the theoretical framework of GSGP. Moreover, the relative vector of this child and the target semantics is perpendicular to the vector defined by the two parents, similar to Hara's proposal [39]. *Random Segment Mutation (RSM)* was inspired in RDO [79]. In RSM, the offspring stands on the intervals between the parents and the target semantics. In this operator, the offspring is calculated as Equation 2.20, where P and T represents the semantics vectors of the parent and the target, respectively, and $k \in [0, 1]$. The experiments show that the angle-driven geometric operators not only drive the evolutionary process to fit the target semantics more efficiently but also improve the generalization performance.

$$\gamma = \arccos\left(\frac{\vec{V}_1}{\|\vec{V}_1\|} \cdot \frac{\vec{V}_2}{\|\vec{V}_2\|}\right) \quad (2.18)$$

$$\gamma_r = \arccos\left(\frac{\vec{t} - \vec{V}_1}{\|\vec{t} - \vec{V}_1\|} \cdot \frac{\vec{t} - \vec{V}_2}{\|\vec{t} - \vec{V}_2\|}\right) \quad (2.19)$$

$$O = P + k(T - P) \quad (2.20)$$

2.3 Fitness and Selection in Genetic Programming

Vanneschi *et al.* affirmed that since the beginning of GP, many contributions have shown the importance of the population diversity because it can drastically affect the GP performance. Also, the semantic diversity is more important than structural diversity [99]. Some work has been proposed to change the fitness function for promoting population diversity. *Novelty Search* [62] presents an extreme case where the fitness function is replaced by individuals' novelty. The novelty of the individual P is computed as the average of the distances between P and its k -nearest neighbors in the semantic space. The main idea of Novelty Search is to promote population diversity. Rather than viewing open-ended evolution as an adaptive competition, it can be viewed simply as a passive drift through the lattice of novelty. Nguyen *et al.* proposed *Fitness Sharing* [73], a technique that promotes dispersion and diversity of individuals. Their proposal consisted of calculating the individual fitness as Equation 2.21, where m_i is approximately equal to the number of individuals that behave similarly to individual i . For calculating the individuals that behave similarly, they used a distance based on semantics.

$$f'_i = f_i(m_i + 1) \quad (2.21)$$

Ruberto *et al.* defined in [85] the *Error Vector* and *Error Space*. The individual error vector \vec{e}_p is defined as Equation 2.22, where P represents the individual semantics and t the target semantics.

$$\vec{e}_p = P - t \quad (2.22)$$

It can be conceived as a point in an n -dimensional space called error space where vector t is the origin. There are two important properties in this space: *Optimally Aligned Individuals* and *Optimally Coplanar Individuals*. The first one is defined as two individuals A , and B are optimally aligned if it exists a scalar k such that Equation 2.23 is accomplished. In other words, the individuals are optimally aligned if their respective

error vectors are directly proportional, with a proportional constant k .

$$\vec{e}_A = k\vec{e}_B \quad (2.23)$$

Furthermore, if we find two aligned individuals we can calculate the optimal global solution as $P^* = \frac{1}{1-k}A - \frac{k}{1-k}B$. The second property says that three GP individuals A , B , and C are optimally coplanar if the bi-dimensional plane on which \vec{e}_A , \vec{e}_B , and \vec{e}_C lie also intersects the origin of the Cartesian system in the error space. In this case, three equations are proposed in order to calculate the optimal global solution. They extended their work in [13], where two GP systems for exploiting alignment in the error space were proposed. The objective of the first one, ESAGP-1, is to find two individuals that are optimally aligned. For that reason, they used a fitness function that has no relationship with the distance to the target in the semantic space. To define this new fitness function, ESAGP-1 calculates a particular point in the error space that they called center of attraction. The fitness of an individual is the angle between its error vector and the attractor, and it has to be minimized. In other words, small angles are better than large ones. The second GP system was called ESAGP- μ . It can be seen as a generalization of ESAGP-1 aimed at finding three optimally coplanar individuals. In a recent document, Vanneschi *et al.* presented the first usable alignment-based genetic programming system, called *Nested alignment genetic programming (NAGP)* [98]. They used individuals that contain several programs. They called them multi-individuals. The idea is to have pairs of programs where the fitness was the angle between their respective error vectors. For accomplishing that, they proposed a selection scheme based on five selection criteria, which had been organized into a nested tournament. The objectives were: (1) promote diversity; (2) maximize the value of the k constant in Equation 2.23; (3) minimize the sum of the errors; (4) minimize the angle between the error vectors; and, (5) reduce the error of the reconstructed expression.

The most used parent selection scheme in GP is tournament selection based on fitness [22]. However, in this section, we discuss some works that have proposed new methodologies of parent selection for improving the GP performance.

Galvan-Lopez *et al.* [27] applied crossover only to those individuals whose difference in behavior is greater than a defined threshold for every element of the training set. Chu *et al.* proposed two tournament selection techniques that use a statistical test to compare the error vectors of individuals [17, 18]. The first technique is called Statistics-TS1. Similar to the standard tournament selection, several individuals are randomly selected and compared. Wilcoxon signed-rank test is applied to the error vectors of these individuals. For a pair of individuals, if the test shows that they are statistically different, the individual with better fitness is the winner. Conversely, if the test confirms that those individuals are not statistically different, a random individual is selected from the pair. The winner is tested against other individuals, and the process is repeated for all individuals in the tournament. The second tournament selection is called Statistics-TS2; it is similar to Statistics-TS1, but it aims at reducing code growth in the GP population. In Statistics-TS2, if the individuals are not statistically different, then the individual with the smallest size is selected from the pair. They tested their tournament selection techniques using eighteen multivariate regression problems and observed that the proposed method helps to reduce code bloat and the generalization error.

As we mentioned in Section 2.2, Hara *et al.* proposed Deterministic Geometric Semantic Crossover [39]. Later, they introduced *Deterministic Geometric Semantic Genetic Programming with Optimal Mate Selection* [38], a methodology for selecting parents in such a way that the line connecting them is close to the target in the semantic space. The first parent T_p is selected by tournament selection based on fitness. The second parent is selected based on the positional relationship, with the aim that the line connecting both parents will be closest to the target point in the semantic space. For choosing the second parent, for all the individuals T_i , the cosine distance on $\angle PT_p T_i$ is calculated, where P represents the target semantics vector. The individual with the largest cosine distance is selected as a mate of T_p . Then, they combine the parents using the Deterministic Geometric Semantic Crossover operator. Their results

confirm that their method has better performance than Moraglio’s geometric crossover in several symbolic regression problems.

Angle-Driven Selection (ADS) was proposed by Chen *et al.* [14–16]. It aims to select a pair of parents that not only have good fitness values but also are far away from each other regarding the angle-distance of their relative semantics. The first parent is selected by tournament selection based on fitness. The second parent is selected using an algorithm that tests several candidates, chosen by tournament selection based on fitness, and chooses the one with the large angle-distance between its relative semantics and the first parent relative semantics. According to the authors, ADS brings several benefits to the evolutionary process. First, ADS helps to decrease semantic duplicates. Since these parents generally have different semantics and the segment between their points in the semantic space is much larger than nearby parents. It can potentially maintain/increase the semantic diversity of the population. Second, the convex hull of the far-away parents becomes larger, which will increase the probability of covering the target semantics, and has a more accurate fitting to the target semantics.

2.4 Classification in Genetic Programming

In this section, we present the work that uses GP for solving classification problems.

First, we list the work that does not use semantics in the evolutive process. Loveard and Ciesielski [64] proposed five different techniques for representing classification problems in GP. (1) Binary decomposition, where a GP tree is generated for each class, this is equivalent to the strategy one-vs-all. (2) Static range selection, as most of the genetic programs return real values, they divided those values into different intervals where each one represents one class. (3) Dynamic range selection, in this case, instead of using static ranges, it is allowed for each individual to determine those ranges dynamically. (4) Class Enumeration, they add a new node function that rep-

represents the instruction IF with the goal of that each individual returns the class type, which is a categorical value that indicates the class assigned to samples. Finally, (5) Evidence Accumulation. This representation contains a vector with k entries, where k represents the number of classes. Before program execution, the vector is initialized to zero. As the program executes, values are added (or subtracted) from certain elements of the vector. At the end, the element with the highest value in the vector is declared to be the most certain outcome for classification. Based on their results, they found that dynamic range selection is more appropriate for both binary and multi-class problems as it is capable of producing classifiers of a higher accuracy degree. Muni *et al.* [69] proposed to evolve a GP tree for each class following an equivalent strategy of one-vs-all approach. For a two-class problem, a single tree T can represent a solution, and the label assignation is defined in Equation 2.24. They extended it to a multi-category classification problem. In their design, a solution can be represented for k trees, one per class, and the label assignation is described in Equation 2.25. Jaben and Baig [47] developed a two-stage method for constructing multi-class classifiers based on GP trees. They used the principle one-vs-all. The first stage creates a classifiers population where individuals are trained to discriminate between the absence and presence of a particular class amongst many classes. The second stage takes the individual discriminators from the first stage and combines them in a single solution. This stage eliminates the need for any conflict resolution mechanism that requires extra computation.

$$\text{class } 1 \text{ if } T(x) \geq 0, \text{ else class } 2 \quad (2.24)$$

$$\text{class } i \text{ if } T_i(x) \geq 0 \text{ and } T_j(x) < 0 \quad \forall j \neq i \quad i, j \in \{1, 2, \dots, k\} \quad (2.25)$$

Ingallli, Silva, Castelli, and Vanneschi affirmed in [46] that GP was never regarded as a good method to perform multi-class classification. They also proposed a GP framework called Multi-dimensional Multi-class Genetic Programming (M2GP). The main idea is to transform the original space into another one using functions evolved with GP, then, a centroid is calculated for each class, and the vectors are as-

signed to the class that corresponds to the nearest centroid using the Mahalanobis distance. M2GP takes as argument the dimension of the transformed space; this parameter is evolved in M3GP [70] by including specialized search operators that can increase or decrease the number of feature dimensions produced by each tree. They extended M3GP and proposed M4GP [60] that uses a stack-based representation in addition to new selection methods, namely lexicase selection, and age-fitness Pareto survival. They end their document with the following statement: “It may also be of interest to explore whether or not variation operators with semantic guarantees can be developed in this context”.

Naredo *et al.* [72] used Novelty Search for evolving genetic programming classifiers based on M3GP. They used semantics for computing the novelty of individuals. To the best of our knowledge, they are the first that used novelty search and semantics for evolving GP classifiers. Each GP individual is represented as a binary vector whose length is the training set size, where every entry is set to 1 if the classifier assigns the class label correctly and 0 otherwise. Then, they used those binary vectors to measure the sparseness among individuals, and the more the sparseness, the higher the fitness value. Their results show that all their NS variants achieve competitive results relative to the traditional objective-based. Galván-López *et al.* [28] also used semantics in Multi-objective GP for solving unbalanced binary classification problems.

Auto machine learning consists of obtaining a model automatically, e.g., classifier or regressor, that includes the steps of preprocessing, feature selection, classifier selection, and hyperparameters tuning. Feurer *et al.* [23] developed a robust automated machine learning (AutoML) technique using Bayesian optimization methods. It is based on scikit-learn [80], using 15 classifiers, 14 feature preprocessing methods, and 4 data preprocessing techniques, giving rise to a structured hypothesis space with 110 hyperparameters. Olson *et al.* [76] proposed the use of GP to develop a robust algorithm that automatically constructs and optimizes machine learning pipelines through a Tree-based Pipeline Optimization Tool (TPOT). On classification, the objective con-

sists of maximizing accuracy score performing a searching of the combinations of 14 preprocessors, 5 feature selectors, and 11 classifiers; all these techniques implemented in scikit-learn [80].

2.5 Summary

Recent proposed operators in Genetic Programming							
Year	Authors	Use of semantics	Operator(s) name(s)	Main idea	Function set	Application	Main results
2008 - 2009	Beadle and Johnson [3, 4]	Indirect	Semantically driven crossover and mutation operators	The crossover is applied several times, and the offspring that are semantically equivalent to their parents are rejected. The mutation allows the operation only if the individual is semantically different from its parent.	IF, AND, OR, NOT	Simple problems: bit Multiplexer and even 5 parity	Increase performance of GP and decrease code bloat
2009 - 2012	Quang Uy <i>et al.</i> [83, 96]	Indirect	Semantic Similarity based Crossover and Mutation operators	The crossover combines individuals at a subtree level trying that subtrees are semantically different but no so much, which means the subtrees need to be semantically similar. The mutation allows replacing a subtree of the individual only if the new subtree is semantically similar to the one that is going to be replaced.	+, -, *, sin, cos, exp, and, log	Real-valued symbolic regression. Unidimensional functions taking as reference 20 or 100 points.	The operators were better than traditional standard GP crossover and mutation operators
2012	Hara <i>et al.</i> [40]	Indirect	Semantic Control Crossover	It combines individuals at a subtree level and performs a global search at the beginning of the evolutive process and a local search in the end.	+, -, *	A real-valued symbol regression problem and even 5 parity	The proposal got better performance than conventional crossover
2014 - 2015	Graff <i>et al.</i> [29, 30, 92]	Indirect	Semantic crossover and mutation operators based on the Partial Derivative Error	They propagate the partial derivative error to know the optimal values in the crossover or mutation points.	+, -, *, /	Symbolic regression problems	Best performance than traditionally GP systems
2004 - 2012	Moraglio <i>et al.</i> [67, 68]	Direct	Geometric Semantic Genetic Programming (GSGP)	The crossover operator SX generates an offspring as $r \cdot P_1 + (1 - r) \cdot P_2$, where $r \in [0, 1]$. The mutation operator is defined as $P + ms * (R_1 - R_2)$, where R_1 and R_2 are random trees, and $ms \in [0, 1]$.	And, Or, Not / +, -, * / if then else	Boolean functions / Polynomial regression / Classification	Advantages: Offspring fitness cannot be worse than the worst fitness of the parents. The problem is converted into a convex one. Disadvantages: The offspring is always bigger than the sum of the size of its parents. This makes the use of this operator hard in practice.

Recent proposed operators in Genetic Programming

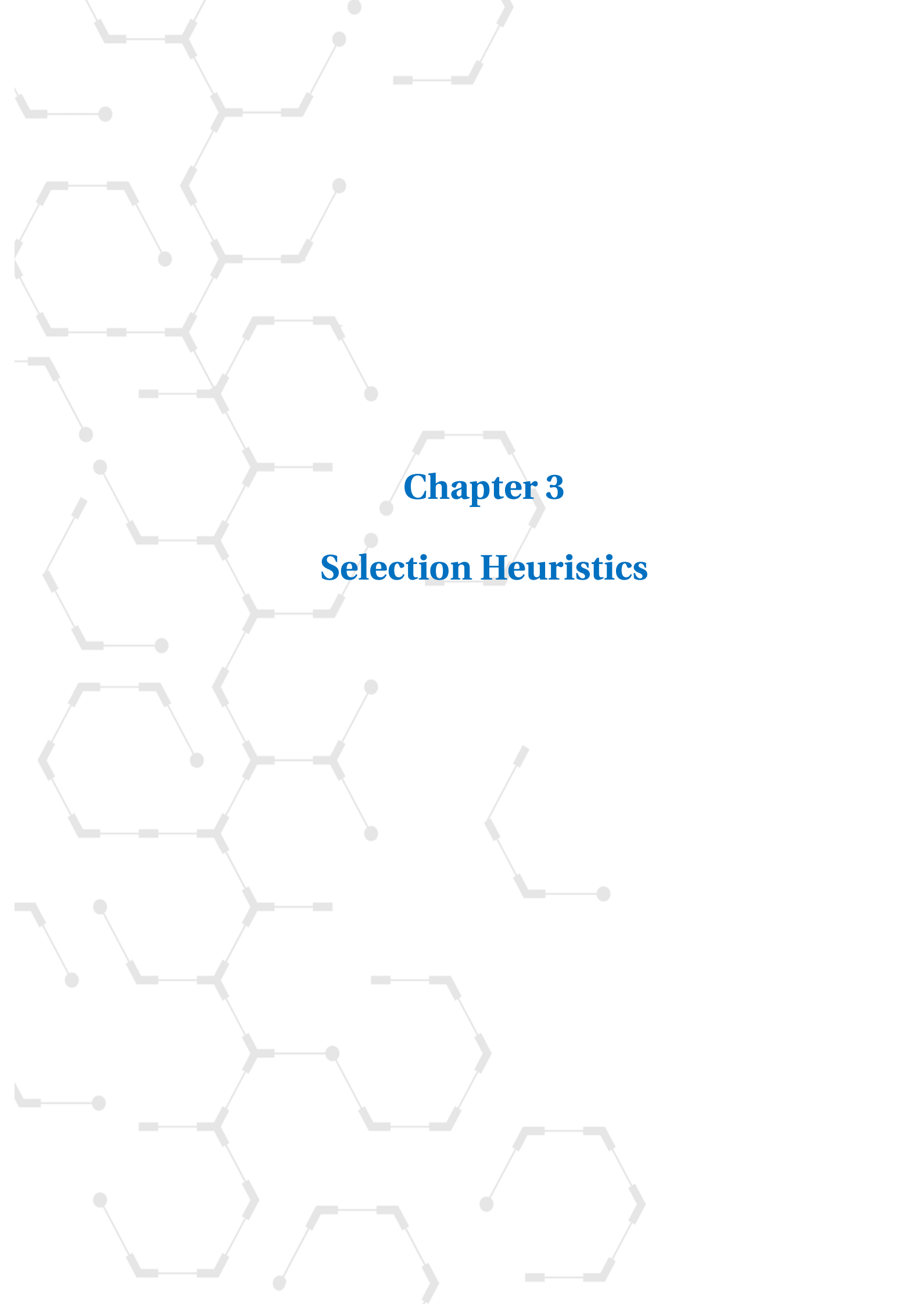
Year	Authors	Use of semantics	Operator(s) name(s)	Main idea	Function set	Application	Main results
2009 - 2012	Krawiec <i>et al.</i> [51, 53]	Direct	Approximating geometric crossover in semantic space	The crossover operator tries to minimize $d(P_1, O) + d(P_2, O) + d(P_1, O) - d(P_2, O) $ to generate offspring whose semantics are close to their parents' semantics, but also, they promoted equidistance between parents and offspring semantics. He defined the <i>geometric crossover operators</i> as operators that generate offspring with the following characteristic $\ O, P_1\ + \ O, P_2\ = \ P_1, P_2\ $.	$+, -, *, /$	Symbolic regression problems	Reduce the computational cost of SX
2012 - 2013	Krawiec and Pawlak <i>et al.</i> [54, 55, 58]	Direct	Locally geometric semantic crossover (LGX)	It approximates the crossover operator of GSGP at a subtree level. It uses a library of procedures.	$+, -, *, /$	Read-valued symbolic regression	Get the same results of RX but reducing the bloat code
2013 - 2015	Krawiec, Pawlak, and Wieloch [56, 79]	Direct	Approximating geometric crossover by semantic backpropagation (AGX) and Random Desired Operator (RDO)	It approximates the crossover operator of GSGP at a subtree level. AGX calculates the point in the center of the segment joining the parents semantics $s(p_1)$ and $s(p_2)$ as $m = \frac{s(p_1) + s(p_2)}{2}$. It uses backpropagation for propagating the expected semantics in the crossover points, and it searches in a library of procedures the ones whose semantics are the closests to the desired ones. RDO is a mutation operator that propagates the target semantics until the mutation point to calculate the desired semantics in that node. Then, a procedure whose semantics is the closests to the desired semantics is searched in a procedure library.	$+, -, *, /, \sin, \cos, \exp, \log$	Read-valued symbolic regression	AGX and RDO converges faster than LGX and classic GPX for symbolic regression benchmarks.
2015	Graff <i>et al.</i> [35, 37]	Direct	Crossover operators based on projections in the phenotype space	The first crossover operator creates a plane in the semantic space using the parents' semantics and the origin of the space. The offspring is calculated as the projection of the target in that plane. The second operator uses a linear combination of k parents as $\sum_k \alpha_k P_k(x)$, with the idea of optimizing the coefficients $\{\alpha_i\}$ with ordinary least squares (OLS).	$+, -, *, /, \sin, \cos, \exp, \log$	Read-valued symbolic regression	The offspring will be at least as good as the best parent.
2016	Nguyen <i>et al.</i> [74]	Direct	Subtree Semantic Geometric Crossover (SSGX)	It approximates the crossover operator of GSGP at a subtree level. It searches in the parents' trees the subtrees whose semantics are the closests to the parents' semantics. Then, the offspring is generated using an equation similar to the one proposed in GSGP but with the subtrees.	$+, -, *, /, \sin, \cos, \exp, \log$	Read-valued symbolic regression, 4 UCI regression problems	SSGX reduces the code growth of SX

Recent proposed operators in Genetic Programming							
Year	Authors	Use of semantics	Operator(s) name(s)	Main idea	Function set	Application	Main results
2015 - 2019	Castelli <i>et al.</i> [9, 12]	Direct	Geometric Semantic Genetic Programming with Local Search (GSGP-LS)	In this mutation operator, an individual P is modified with the equation $O = \alpha_0 + \alpha_1 P + \alpha_2(R_1 - R_2)$ where R_1 and R_2 are random trees, and, $\alpha_i \in \mathbb{R}$. $\{\alpha_i\}$ are calculated using the target semantics and OLS for getting the better linear combination of the original and the random trees. In the extension of their work, they proposed to apply local search for all the individuals during a separate step after mutation and crossover. For a GP tree T , they calculated an optimized one T' as $T' = \sum_j \alpha_j f_j(T)$, where $f_j: \mathbb{R} \rightarrow \mathbb{R}$, and the values of α_i are optimized with OLS minimizing the error between the individual semantics and target semantics.	$+, -, *, /$	3 real life multivariable regression problems	GSGP-LS outperforms GSGP on the training set, but overfits it. On the test set GSGP is better than GP. The hybrid proposal, integrating GSGP and GSGP-LS, got excellent results on the training and test set.
2016	Hara [39]	Direct	Deterministic Geometric Semantic Crossover	They calculated the parameter r of the GSGP crossover operator as $r = \frac{ P_1 P_2 - P_1 T \cos \theta}{ P_1 P_2 }$ where $\theta = \angle T P_1 P_2$, and P_1 , P_2 and T are the semantics of the parents and the target.	$+, -, *$	Real-valued symbolic regression	The offspring is the projection of the target in the line formed by the parents to determine the optimal combination. It improves the performance of GSGP.
2016	Szubert <i>et al.</i> [95]	Direct	Forward Propagation Mutation (FPM)	It defines a structure of four new nodes for the individual. FPM uses a combination of forward and back-propagation to find a combination of unitary and binary functions that is the most similar to the desired behavior.	$+, -, *, /, \sin, \cos, \exp, \log$	Real-valued symbolic regression	FPM produced shorter programs than RDO, also it obtained significantly lower error on the unseen test dataset than RDO.
2019	Virgolin <i>et al.</i> [101]	Direct	Linear scaling of RDO	They introduced a scaled version of MSE incorporating the scalar values a and b as follows $MSE(y, o) = \frac{1}{n} \sum_i (y_i - (a + b o_i))^2$, where a and b are optimized.	$+, -, *, /$	Real-world benchmark regression problems	The incorporation of linear scaling within semantic backpropagation-based GP leads to much lower errors, and outperforms the use of traditional variation operators
2017 - 2019	Chen <i>et al.</i> [14-16]	Direct	Angle-Driven Geometric Semantic GP (ADGSGP)	They defined the angle between the relative vectors of two individuals' semantics as $\gamma_r = \arccos\left(\frac{\vec{r} - \vec{v}_1}{\ \vec{r} - \vec{v}_1\ } \cdot \frac{\vec{r} - \vec{v}_2}{\ \vec{r} - \vec{v}_2\ }\right)$. They proposed three techniques. (1) Angle-Driven Selection (ADS) selects a pair of parents that have good fitness values as well as are distant to each other regarding the angle-distance of their relative semantics. (2) Perpendicular Crossover (PC) generates a child point standing on the line crossing the two parents, as GSGP; but in this case, its semantics vector is perpendicular to the vector defined by the two parents. (3) Random Segment Mutation (RSM) where the offspring stands on the interval between the parent and the target semantics.	$+, -, *, /$	Multivariable symbolic regression	ADGSGP has a faster convergence rate than traditionally GP and GSGP; a good interpret ability, and requires less computational effort.

Fitness function				
Year	Authors	Proposal name	Main idea	Main results
2011	Lehman and Kenneth [62]	Novelty Search (NS)	It presents an extreme case where the fitness function is replaced by individuals' novelty. The novelty of the individual P is calculated as the average of the distances between P and its k -nearest neighbors in the semantic space.	It promotes population diversity and gets diverse solutions.
2012	Nguyen <i>et al.</i> [73]	Fitness Sharing	Their proposal consisted of calculating the individual fitness as $f_i' = f_i(m_i + 1)$, where m_i is approximately equal to the number of individuals that behave similarly to the individual i .	It promotes the dispersion and diversity of individuals
2014 - 2019	Ruberto <i>et al.</i> [13, 85, 98]	Nested alignment genetic programming (NAGP)	They defined the error space where each individual can be represented as the vector $\vec{e}_p = P - t$, where P represents the individual semantics and t the target semantics. In this space, the target vector t is the origin. Also, they defined two properties: Optimally Aligned Individuals and Optimally Coplanar Individuals . Those properties said that if two or three individuals are founded whose error vectors accomplish those properties, then the problem is solved. Later, they proposed two GP systems where the fitness function is changed with the aim of searching individuals that satisfy those properties. In 2019, they presented the first usable alignment-based genetic programming system, called Nested alignment genetic programming (NAGP).	NAGP outperforms GSGP and GSGP-LS on four complex real-life applications. Its models are not only more effective but also significantly smaller.

New schemes for parent selection					
Year	Authors	Proposal name	Main idea	Application	Main results
2013	Galvan-Lopez <i>et al.</i> [27]	Using semantics in the selection mechanism in GP	They applied crossover only to those individuals whose difference in behavior is greater than a defined threshold for every element of the training set.	Artificial Ant Problem, Even-n Parity problem, and Real-Valued Symbolic Regression problems.	The semantics in selection approach has shown promising results, in many cases achieving superior results compared to the crossover-semantics based approach.
2016 - 2018	Chu <i>et al.</i> [17, 18]	Semantic tournament selection for GP based on statistical analysis of error vectors	They used the Wilcoxon signed-rank test for comparing whether individuals are statistically different or not. If they are statistically different, they select the one with the best fitness as a parent. Otherwise, they choose the smaller one.	GP benchmark problems and UCI repository regression problems.	The proposed techniques were better than standard tournament selection and neatGP (the state of the art method for controlling GP code bloat) in improving GP generalisation and reducing GP code growth.
2016	Hara <i>et al.</i> [38]	Deterministic Geometric Semantic Genetic Programming with Optimal Mate Selection	They proposed a selection scheme for choosing parents in such a way that the line connecting them is closed to the target in the semantic space.	Real-valued symbolic regression.	The proposed selection scheme can improve search performance of deterministic GSGP by using the appropriate application rate.
2017 - 2019	Chen <i>et al.</i> [14–16]	Angle-Driven Selection (ADS)	They proposed a selection scheme to choose a pair of parents that have good fitness values as well as are distant to each other regarding the angle-distance of their relative semantics.	Multivariable symbolic regression	In general, ADGSGP has a faster convergence rate than traditionally GP and GSGP, a good interpretability, and requires less computational effort.
2019	Vanneschi <i>et al.</i> [98]	Nested alignment genetic programming (NAGP)	They proposed a selection scheme based on five selection criteria, which had been organized into a nested tournament. The objectives were: (1) promote diversity; (2) maximize the value of the k constant, explained in Section 2.3; (3) minimize the sum of the errors; (4) minimize the angle between the error vectors; and, (5) reduce the error of the reconstructed expression.	UCI regression problems	NAGP outperforms GSGP and GSGP-LS on four complex real-life applications. Its models are not only more effective but also significantly smaller.

Genetic Programming classifiers					
Year	Authors	Use of semantics	Proposal name	Main idea	Main results
2001	Loveard and Ciesielski [64]	No	Representing classification problems in Genetic Programming	They proposed five different techniques for representing classification problems in GP. (1) Binary decomposition, (2) Static range selection, (3) Dynamic range selection, (4) Class Enumeration, and (5) Evidence Accumulation. Dynamic range selection scheme dynamically divided those values into different intervals where each one represents one class.	Dynamic range selection is more appropriate for both binary and multi-class problems as it is capable of producing classifiers of a higher degree of accuracy when comparable training times are allowed.
2004	Muni <i>et al.</i> [69]	No	A Novel Approach to Design Classifiers using Genetic Programming	They proposed to evolve a GP tree for each class following an equivalent strategy of one-vs-all approach.	They proposed a comprehensive scheme for classifier design for multiclass problems using a multitree concept of GP, moreover suitable crossover and mutation operators.
2013	Jaben and Baig [47]	No	Two-stage learning for multi-class classification using Genetic Programming	They developed a two-stage method for constructing multi-class classifiers based on GP trees. The first stage creates a population of GP classifiers for particular classes, one-vs-all. The second one combines those classifiers to create a multi-class classifier.	The two-stage learning scheme has yielded better results when compared to one-versus-all or binary decomposition method.
2016	Naredo <i>et al.</i> [72]	Yes	Evolving Genetic Programming classifiers with novelty search	Their proposal is based on M3GP, but in this case, novelty search (NS) and semantics are used instead of fitness for guiding the evolutive process.	In terms of performance, results show that all NS variants achieve competitive results relative to the standard approach in GP. Moreover, NS variants got smaller program trees.
2014 - 2019	Ingalalli, Castelli, Muñoz, Silva, Trujillo, La Cava, Vanneschi <i>et al.</i> [46, 60, 70]	No	M2GP, M3GP and M4GP	They proposed several Multi-dimensional Multi-class Genetic Programming algorithms where the main idea is to transform the original space into another one using functions evolved with GP, then, a centroid is calculated for each class, and the vectors are assigned to the class that corresponds to the nearest centroid using the Mahalanobis distance.	M4GP produces better results than classical classifiers as LR, RF, MLP, KNN and RF but it is not better than TPOT.



Chapter 3
Selection Heuristics

3 Selection Heuristics

In this chapter, we describe in detail the main contribution of this thesis: the parent selection heuristics that are inspired in function properties and use semantics for guiding the selection process. We used EvoDAG, which was described in Section 1.4, for testing all the selection schemes.

Section 3.1 describes the motivation of our heuristics. It is followed by Section 3.2 that describes the schemes for parent selection. It starts with the description of the classical technique for parent selection in GP: tournament selection based on fitness, and how it is implemented in EvoDAG (Subsection 3.2.1). Random selection is described in Subsection 3.2.2. Subsections 3.2.3 and 3.2.4 describe our selection heuristics: (1) *Tournament selection based on cosine similarity* (sim), (2) *Tournament selection based on Pearson's correlation coefficient* (prs) and (3) *Tournament selection based on the accuracy* (acc). Finally, Section 3.3 presents the techniques for negative selection that are negative tournament selection based on fitness and random negative selection.

3.1 Motivation

This thesis proposes new selection heuristics for GP tailored to classification problems based on the idea that function properties and individual semantics can guide parent selection.

Let us recall that in a steady-state evolution, which is used in EvoDAG, there are two stages where selection takes place, on the one hand, the selection is used to choose the parents, and on the other hand, the selection is applied to decide which individual, in the current population, is replaced by the offspring. This last one is called negative selection. The most popular parent selection method in GP is tournament selection

based on fitness [22] (see Subsection 3.2.1), and also, negative selection is commonly performed using the same selection scheme (see Subsection 3.3.1).

Nonetheless, as we reviewed in Chapter 2, there have been approaches that investigate the behavior of EAs when the fitness function is replaced or combine with other techniques in the task of parent selection. The extreme case would be to replace the fitness function, in all the evolutionary process, as done in Novelty Search [62] where the fitness of an individual is related to its novelty. The novelty of an individual is computed as the average of the distances between it and its k -nearest neighbors. GP with Novelty Search has been used in classification problems (see [72]) where the novelty is computed with the 0-1 loss function, i.e., it counts the differences between the behavior of any pair of individuals. Novelty Search has proved that it can be useful in the evolutionary process, and further, to evolve classifiers. However, random selection has not been evaluated, and it is the less expensive of any selection schemes. So, we include in this dissertation the comparison of random selection for parent selection and negative selection.

Moreover, some selection schemes that use semantics have been proposed for improving GP performance. They are described in detail in Section 2.3. The most significant for us are Nested alignment genetic programming (NAGP) [98] and Angle-Driven Selection (ADS) [14–16]. In NAGP, the objective is to minimize the angle between the error vectors of programs in multi-individuals, and the authors proposed a selection scheme based on five selection criteria, which had been organized into a nested tournament. The objectives were: (1) promote diversity; (2) maximize the value of the constant k in Equation 2.23; (3) minimize the sum of the errors; (4) minimize the angle between the error vectors; and, (5) reduce the error of the reconstructed expression. On the other hand, the aim of ADS is selecting a pair of parents that have good fitness values as well as are distant to each other regarding the angle-distance of their relative semantics. Those approaches are interesting because they used semantics to guide parent selection. However, to the best of our knowledge, no one has proposed

selection schemes based on functions' properties. Our primary motivation is that the selection scheme used for one function cannot be useful in another one. For example, the parent selection scheme used in the function Σ cannot perform as well as for the function \min . For that reason, we proposed new selection schemes based on functions' properties and individual semantics.

3.2 Parent Selection

In the case of EvoDAG, for the creation of an offspring, a function f is randomly selected from the function set \mathcal{F} , and then parent selection needs to be performed to choose each one of the k arguments (or parents). For example, Figure 3.1 shows an example of parent selection where the function Σ was selected, and, 3 individuals from the population \mathcal{P} need to be selected as arguments for creating the new offspring. This thesis compares different schemes for parent selection; in other words, how the arguments of functions need to be chosen. Parent selection is generally applied by tournament selection based on fitness [22] (see Subsection 3.2.1). The original implementation of EvoDAG uses that scheme.

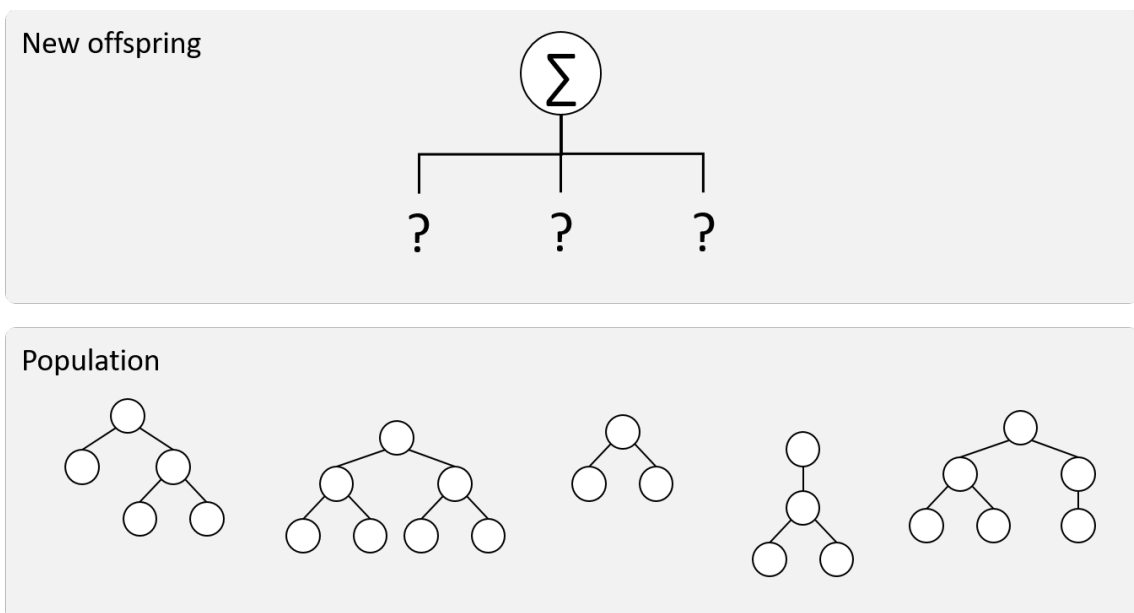


Figure 3.1: Diagram of parent selection in EvoDAG. Source: Own elaboration.

This section presents the classical parent selection scheme, tournament selection based on fitness (Subsection 3.2.1); random selection (Subsection 3.2.2) and our proposed parent selection heuristics based on: cosine similarity (Subsection 3.2.3), Pearson's correlation coefficient (Subsection 3.2.3) and accuracy (Subsection 3.2.4).

3.2.1 Parent Selection based on Fitness (fit)

Parent Selection based on Fitness

It tries to select fittest individuals as arguments. Each argument is independently chosen using tournament selection based on fitness.

In GP, as well as in EvoDAG, the classical parent selection scheme is tournament selection based on fitness. This tournament randomly selects several individuals from the population and chooses as the parent the fittest one. Specifically, in EvoDAG, each one of the k arguments (or parents) are independently selected from the population \mathcal{P} using a tournament selection based on fitness. Figure 3.2 shows a simple example of parent selection in EvoDAG. In this case, an offspring is created with the function Σ . Three parents need to be chosen from the population \mathcal{P} for becoming arguments. For each one of the arguments, two individuals are randomly selected from the population and the fittest one is chosen as parent. In this case, the number of individuals in the tournament ts is two. Algorithm 3 describes the selection of arguments in EvoDAG based on fitness.

3.2.2 Random Selection of Parents (rnd)

Random Selection of Parents

It randomly selects the individuals.

The most straightforward and less expensive strategy for selecting parents, or in this case, arguments, is random. The proposal of Novelty Search [62] produces good re-

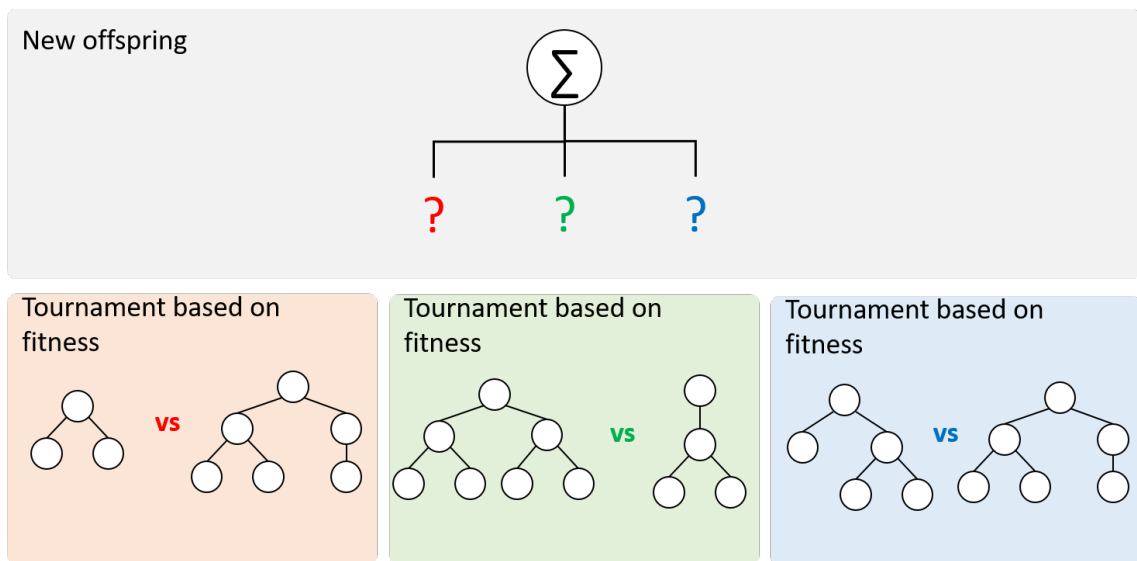


Figure 3.2: Diagram of parent selection based on fitness (in EvoDAG). Source: Own elaboration.

sults, in that case, individuals' novelty replaces the fitness function. It makes us wonder what happens if parent selection is performed randomly. For that reason, we include in the comparison random selection. In this case, the process consists of selecting all the arguments of functions randomly from the population \mathcal{P} . Figure 3.3 shows a simple example of random parent selection in EvoDAG. In this case, function Σ is selected and it needs three arguments. But, as we said, they are randomly selected from the population. Algorithm 4 describes random selection of parents for selecting the arguments of functions in EvoDAG.

Algorithm 3: Selection of arguments based on fitness (fit) in EvoDAG

Input: k , the number of arguments

Input: ts , the tournament size

Output: The list of individuals that will be passed as arguments

$args \leftarrow$ an empty list of individuals;

for i **from** 1 **to** k **do**

$A \leftarrow$ an individual randomly selected from the population \mathcal{P} ;

$fitness_A \leftarrow$ fitness of A ;

for j **from** 2 **to** ts **do**

$B \leftarrow$ an individual randomly selected from the population \mathcal{P} ;

$fitness_B \leftarrow$ fitness of B ;

if $fitness_B > fitness_A$ **then**

$A \leftarrow B$;

$fitness_A \leftarrow fitness_B$;

end

end

 Add the individual A to $args$;

end

Return $args$;

Algorithm 4: Random selection of arguments (rnd) in EvoDAG

Input: k , the number of arguments

Output: The list of individuals that will be passed as arguments

$args \leftarrow$ an empty list of individuals;

for i **from** 1 **to** k **do**

$A \leftarrow$ an individual randomly selected from the population \mathcal{P} ;

 Add the individual A to $args$;

end

Return $args$;

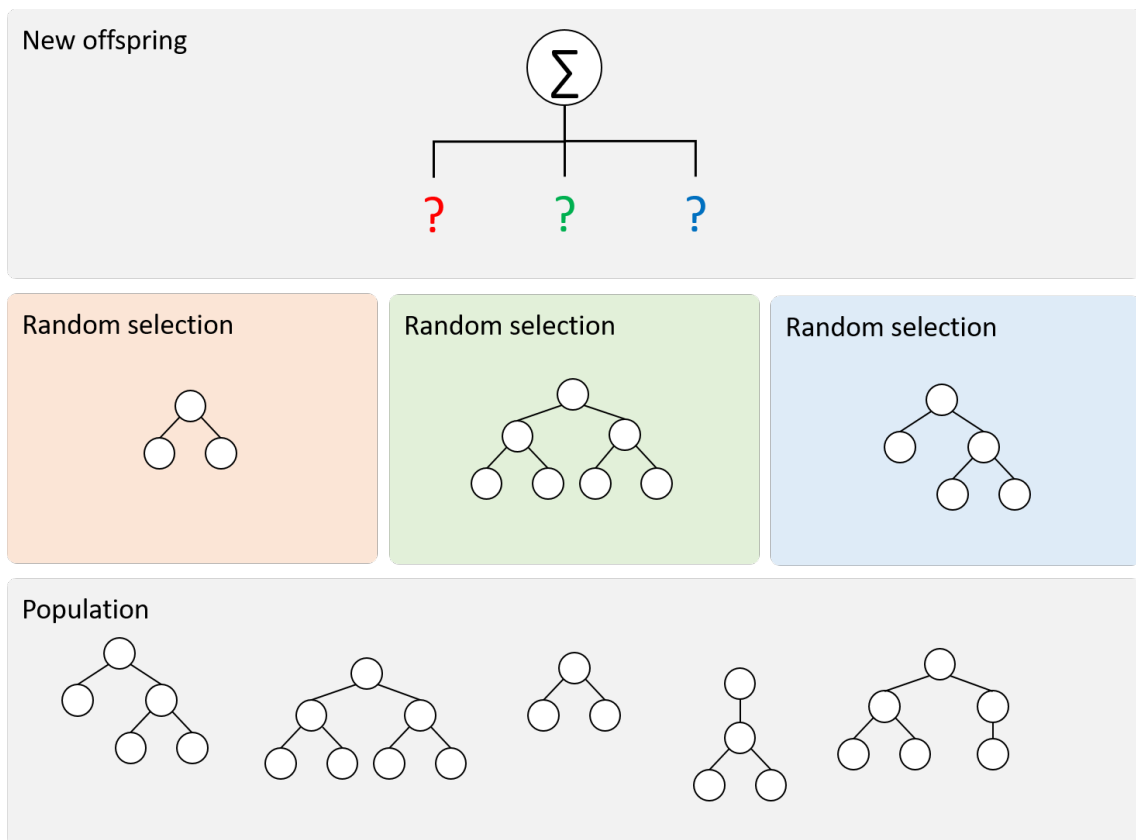


Figure 3.3: Diagram of random selection of parents (in EvoDAG). Source: Own elaboration.

3.2.3 Parent Selection based on Cosine Similarity (sim) and Pearson's Correlation Coefficient (prs)

Parent Selection based on Cosine Similarity

It consists of selecting individuals whose semantics' vectors ideally have rectangle angles. The absolute cosine similarity is used to measure the angle between individuals' semantics. This heuristic is designed based on the properties of the function Σ and the classifiers Naive Bayes (NB and MN) and Nearest Centroid (NC).

$$sim(\vec{v}_1, \vec{v}_2) = \frac{|\vec{v}_1 \cdot \vec{v}_2|}{\|\vec{v}_1\| \|\vec{v}_2\|}$$

Parent Selection based on Pearson's Correlation Coefficient

It aims to select parents whose semantics vectors are uncorrelated among them. We use the absolute Pearson's correlation coefficient for measuring the correlation among inputs. This heuristic is designed based on the properties of the function Σ and the classifiers Naive Bayes (NB and MN) and Nearest Centroid (NC).

$$prs(v_1, v_2) = \frac{|(v_1 - \bar{v}_1) \cdot (v_2 - \bar{v}_2)|}{\|v_1 - \bar{v}_1\| \|v_2 - \bar{v}_2\|}$$

As we said at the beginning of this chapter, the main idea of our heuristics is to be inspired by functions' properties and use semantics for guiding the parent selection. The first heuristics, based on cosine similarity and Pearson's correlation coefficient, are related to the relationship of individuals' semantics in the semantic space.

On one hand, we consider the function Σ that in EvoDAG is defined as $\sum_k \theta_k p_k$. It can be seen as a linear combination of individuals. It is well known that a set of linearly independent vectors generates a base for a specific space. Then, we can enhance the performance of the function Σ if its arguments are linearly independents among them.

On the other hand, we consider the classifiers Naive Bayes and Nearest Centroid. Naive Bayes is based on Bayes's theorem with the "naive" assumption of independence between every pair of variables; this is, it assumes that its inputs are uncorrelated. Then, if Naive Bayes receives uncorrelated variables, it will work better. The Nearest Centroid classifier works as follows: it calculates the centroids of all classes, then, for predicting a new sample, the assigned label will be the class that corresponds to the nearest centroid. It is common to use Euclidean distance that is defined as $d(\vec{v}_1, \vec{v}_2) = \sqrt{\sum_i (v_{1i} - v_{2i})^2}$. It can be seen that it calculates the distance using each variable independently. For that reason, we assume that the performance of Nearest Centroid could increase if it receives uncorrelated inputs, as well as Naive Bayes. Fig-

Figure 3.4 shows an example of the Nearest Centroid performance with uncorrelated and correlated variables. It can be seen that at least in that example, Nearest Centroid performs better using uncorrelated variables. Based on that, for both classifiers, we tried to select parents whose semantics are uncorrelated.

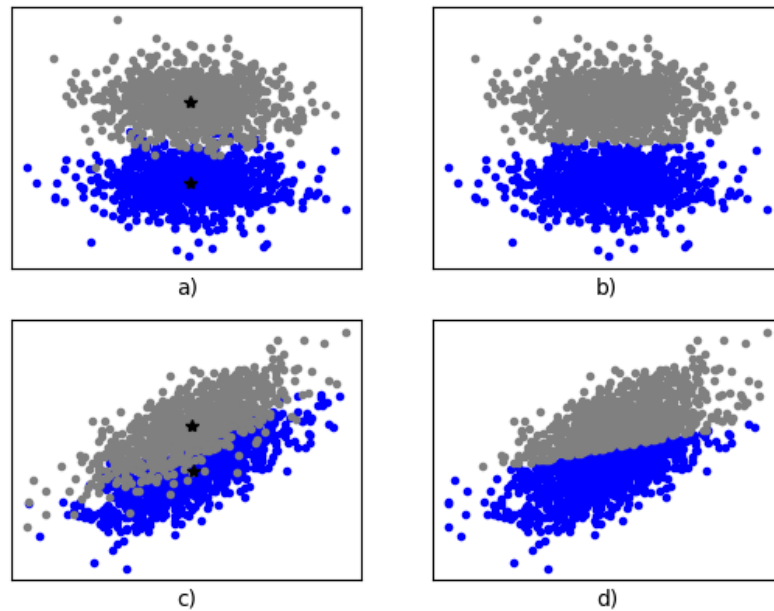


Figure 3.4: Example of Nearest Centroid classifier with uncorrelated and correlated variables. a) and c) Classes and their centroids. b) and d) Predicted classes using the classes centroids and the algorithm Nearest Centroid. Source: Own elaboration.

Brereton reviewed and compared the properties: orthogonality, uncorrelatedness, and linear independence of vectors in [7]. He affirmed that these concepts are commonly confused because they are related.

Some vectors are *linearly independent* among them if they accomplish the following rules. (1) If the number of entries and vectors are the same, they are linearly independent if the vectors are placed together in a matrix X and its determinant is nonzero. (2) If the number of vectors is less than the number of entries, they are linearly independent if the vectors are placed together in a matrix X , and the determinant of $X'X$ is nonzero [7].

Two vectors are *orthogonal* if the angle between them is equal to 90° or $\frac{\pi}{2}$ rad [7]. *Cosine similarity* measures the cosine of the angle between two vectors in the space. It only uses the direction of vectors without taking into account the magnitude. Its value ranges between -1 and 1. It is 1 if the vectors have exactly the same direction, -1 if their direction is exactly opposite, and 0 if they are orthogonal, as it can be seen in Figure 3.5. The cosine similarity between the vectors \vec{v}_1 and \vec{v}_2 is defined as Equation 3.1, where \cdot represents the dot product and $\|\vec{v}\|$ the norm of \vec{v} .

$$CS(\vec{v}_1, \vec{v}_2) = \cos(\theta) = \frac{\vec{v}_1 \cdot \vec{v}_2}{\|\vec{v}_1\| \|\vec{v}_2\|} \quad (3.1)$$

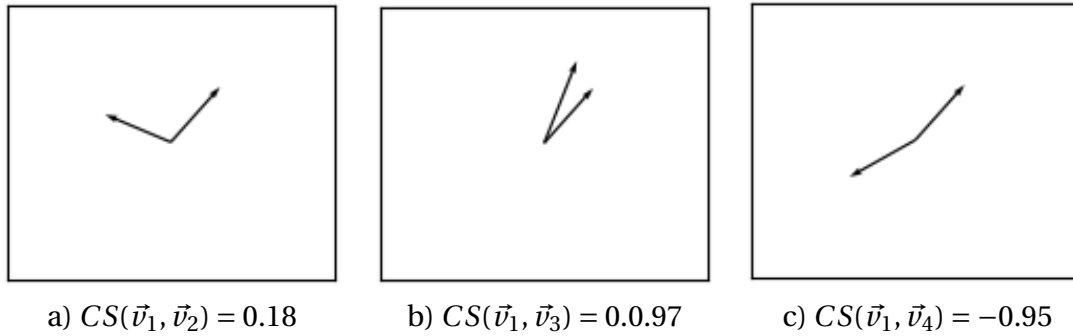


Figure 3.5: Example of Cosine Similarity (CS), $\vec{v}_1 = [3, 6]$, $\vec{v}_2 = [-4, 3]$, $\vec{v}_3 = [2, 9]$, and $\vec{v}_4 = [-4, -4]$. a) Cosine similarity of \vec{v}_1 and \vec{v}_2 , b) Cosine similarity of \vec{v}_1 and \vec{v}_3 , and c) Cosine similarity of \vec{v}_1 and \vec{v}_4 . Source: Own elaboration.

Pearson's correlation coefficient has been widely used for measuring the correlation among variables. The Pearson's coefficient between the input variables, v_1 and v_2 , is defined as Equation 3.2, where *cov* is the covariance and σ represents the standard deviation of variables. When, Pearson's coefficient is calculated using a sample, it can be rewritten as Equation 3.3, where v_1 and v_2 are vectors with the values of the variables, \cdot represents the dot product and $\|\vec{v}\|$ the norm of \vec{v} . It is essential to notice that, if we had centered data, the equation would be the same as cosine similarity. Pearson's correlation coefficient value ranges between -1 and 1. It is 1 if the variables are total positive correlated, 0 represents no linear correlation among variables, and -1 represents a total negative linear correlation.

$$\rho_{v_1, v_2} = \frac{cov(v_1, v_2)}{\sigma_{v_1} \sigma_{v_2}} \quad (3.2)$$

$$\rho_{v_1, v_2} = \frac{(v_1 - \bar{v}_1) \cdot (v_2 - \bar{v}_2)}{\|v_1 - \bar{v}_1\| \|v_2 - \bar{v}_2\|} \quad (3.3)$$

Besides, Brereton affirmed that uncorrelated and orthogonal vectors are linearly independent, as can be seen in the Venn diagram of Figure 3.6.

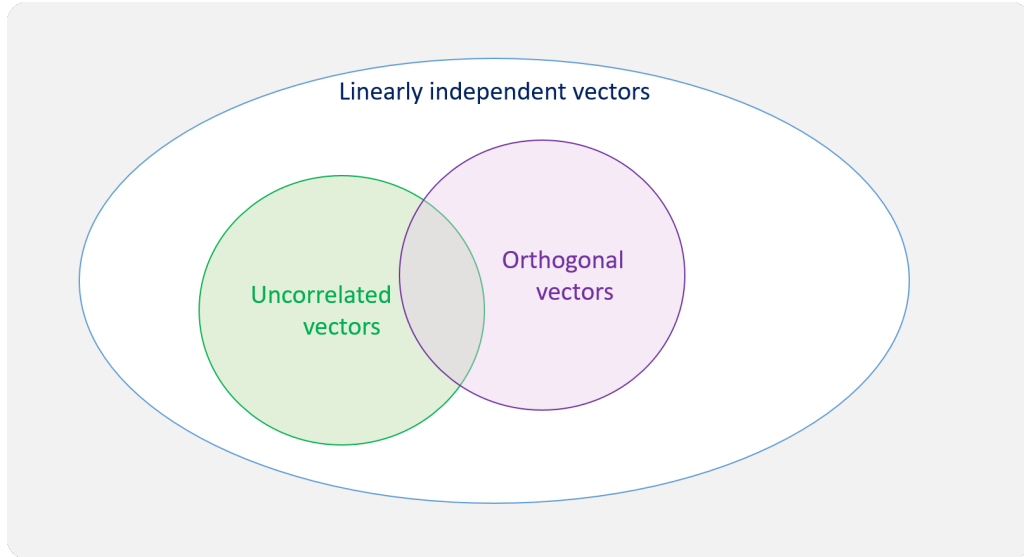


Figure 3.6: Diagram of the relationship between linearly independent, uncorrelated, and orthogonal vectors. Source: Own elaboration.

To sum up, we want linearly independent vectors as arguments for the function Σ and uncorrelated inputs for the classifiers Naive Bayes and Nearest Centroid. Also, as we said, cosine similarity and Pearson's correlation coefficient, which are related, have been used for measuring the angle between vectors and the correlation of vectors, respectively. We decided to create two parent selection heuristics based on these properties. Our first heuristic consists of selecting individuals whose absolute cosine similarity between their semantics vectors is as close as possible to zero. For that reason, we use the absolute of the cosine distance (see Equation 3.4). The second one, the selection heuristic based on Pearson's correlation coefficient, tries to select as parents individuals whose semantics are uncorrelated. Moreover, the direction of uncorrelatedness does not affect, then, we use the absolute of Pearson's coefficient (see Equation 3.5).

$$sim(\vec{v}_1, \vec{v}_2) = \frac{|\vec{v}_1 \cdot \vec{v}_2|}{\|\vec{v}_1\| \|\vec{v}_2\|} \quad (3.4)$$

$$prs(v_1, v_2) = \frac{|(v_1 - \bar{v}_1) \cdot (v_2 - \bar{v}_2)|}{\|v_1 - \bar{v}_1\| \|v_2 - \bar{v}_2\|} \quad (3.5)$$

The selection of arguments based on the absolute of cosine similarity is described using a simple example showed in Figure 3.7. In this case, the function Σ is selected for creating the offspring and it needs three arguments. The first parent is randomly selected from the population \mathcal{P} . Then, the next arguments are independently selected. For each one, a tournament is performed, which is described as follows. First, two individuals are randomly selected from the population \mathcal{P} , and the absolute cosine similarity between their semantics and the first argument semantics is calculated. The individual with the minimum value of the absolute cosine similarity between its semantics and the first parent semantics is selected as argument. Algorithm 5 describes the selection of arguments based on cosine similarity in EvoDAG. It is important to mention that, as we affirmed in Section 1.4, the individuals' semantics in EvoDAG when solving classification problems is an array of semantics vectors, then, the cosine similarity between individuals' semantics is calculated as the sum of similarities among pair of vectors. This process is described in the algorithm.

The selection of arguments based on Pearson's correlation coefficient is described using a simple example showed in Figure 3.8. In this case, the function Σ is selected for creating the offspring and it needs three arguments. The first parent is randomly selected from the population \mathcal{P} . Then, the next arguments are independently selected. For each one, a tournament is performed, which is described as follows. First, two individuals are randomly selected from the population \mathcal{P} , and the absolute Pearson's correlation coefficient between their semantics and the first argument semantics is calculated. The individual with the minimum value of the absolute Pearson's correlation coefficient between its semantics and the first parent semantics is selected as argument. Algorithm 6 describes the selection of arguments based on Pearson's corre-

Algorithm 5: Selection of arguments based on the absolute of cosine similarity in EvoDAG

Input: k , the number of arguments
Input: ts , the tournament size
Output: The list of individuals that will be passed as arguments
 $args \leftarrow$ an empty list of individuals;
 $F \leftarrow$ an individual randomly selected from the population \mathcal{P} ;
 $semantics_F \leftarrow$ the array of individual F 's semantics vectors;
for i **from** 2 **to** k **do**
 $A \leftarrow$ an individual randomly selected from the population \mathcal{P} ;
 $semantics_A \leftarrow$ the array of individual A 's semantics vectors;
 $sim_A \leftarrow 0$;
 for each element in $semantics_F$ **and** $semantics_A$ **do**
 $sim_A \leftarrow sim_A + \frac{|(semantics_{Fi}) \cdot (semantics_{Ai})|}{\|semantics_{Fi}\| \|semantics_{Ai}\|}$;
 end
 for j **from** 2 **to** ts **do**
 $B \leftarrow$ an individual randomly selected from the population \mathcal{P} ;
 $semantics_B \leftarrow$ the array of individual B 's semantics vectors;
 $sim_B \leftarrow 0$;
 for each element in $semantics_F$ **and** $semantics_B$ **do**
 $sim_B \leftarrow sim_B + \frac{|(semantics_{Fi}) \cdot (semantics_{Bi})|}{\|semantics_{Fi}\| \|semantics_{Bi}\|}$;
 end
 if $sim_B < sim_A$ **then**
 $A \leftarrow B$;
 $sim_A \leftarrow sim_B$;
 end
 end
 Add the individual A to $args$;
end
Return $args$;

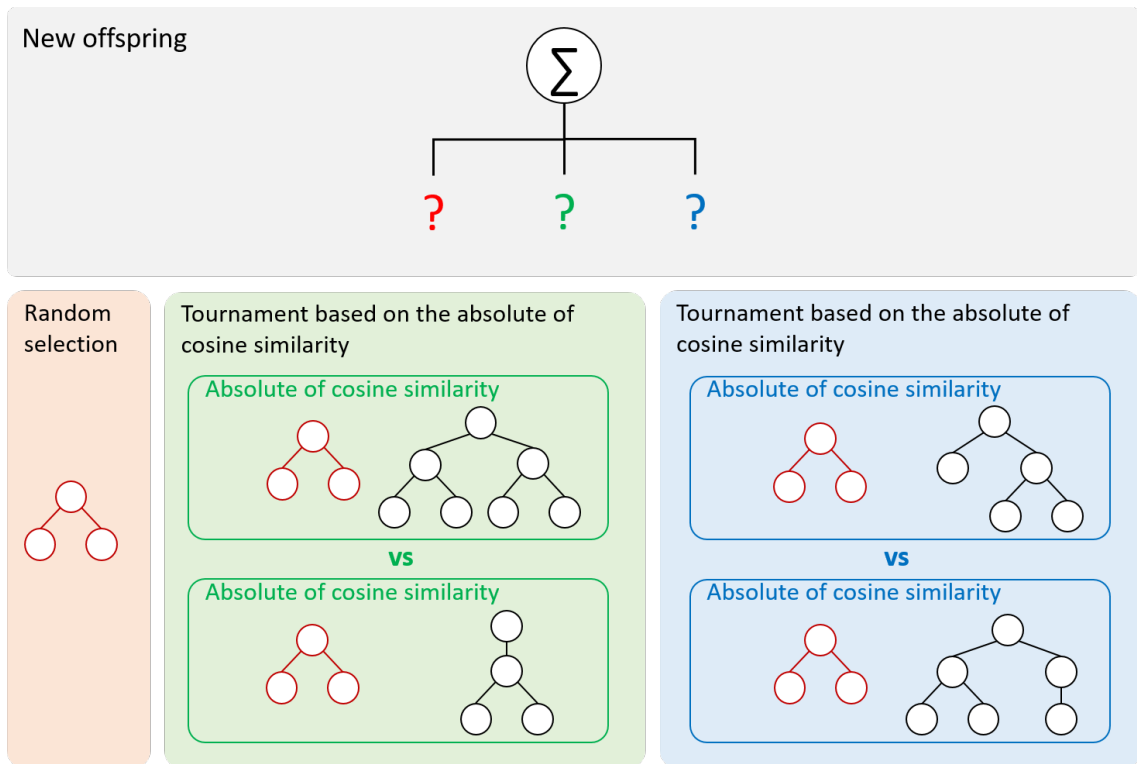


Figure 3.7: Diagram of parent selection based on cosine similarity (in EvoDAG). Source: Own elaboration.

lation coefficient in EvoDAG. It is important to mention that, as we affirmed in Section 1.4, the individuals' semantics in EvoDAG when solving classification problems is an array of semantics vectors, then, the Pearson's correlation coefficient between individuals' semantics is calculated as the sum of correlations among pairs of vectors. This process is described in the algorithm.

Algorithm 6: Selection of arguments based on Pearson's correlation coefficient in EvoDAG

Input: k , the number of arguments
Input: ts , the tournament size
Output: The list of individuals that will be passed as arguments
 $args \leftarrow$ an empty list of individuals;
 $F \leftarrow$ an individual randomly selected from the population \mathcal{P} ;
 $semantics_F \leftarrow$ the array of individual F 's semantics vectors;
for i **from** 2 **to** k **do**
 $A \leftarrow$ an individual randomly selected from the population \mathcal{P} ;
 $semantics_A \leftarrow$ the array of individual A 's semantics vectors;
 $prs_A \leftarrow 0$;
 for each element in sF **and** $semantics_A$ **do**
 $prs_A \leftarrow prs_A + \frac{|(semantics_{Fi} - \bar{semantics}_{Fi}) \cdot (semantics_{Ai} - \bar{semantics}_{Ai})|}{\|semantics_{Fi} - \bar{semantics}_{Fi}\| \|semantics_{Ai} - \bar{semantics}_{Ai}\|}$;
 end
 for j **from** 2 **to** ts **do**
 $B \leftarrow$ an individual randomly selected from the population \mathcal{P} ;
 $semantics_B \leftarrow$ the array of individual B 's semantics vectors;
 $prs_B \leftarrow 0$;
 for each element in sF **and** $semantics_B$ **do**
 $prs_B \leftarrow prs_B + \frac{|(semantics_{Fi} - \bar{semantics}_{Fi}) \cdot (semantics_{Bi} - \bar{semantics}_{Bi})|}{\|semantics_{Fi} - \bar{semantics}_{Fi}\| \|semantics_{Bi} - \bar{semantics}_{Bi}\|}$;
 end
 if $prs_B < prs_A$ **then**
 $A \leftarrow B$;
 $prs_A \leftarrow prs_B$;
 end
 end
 Add the individual A to $args$;
end
Return $args$;

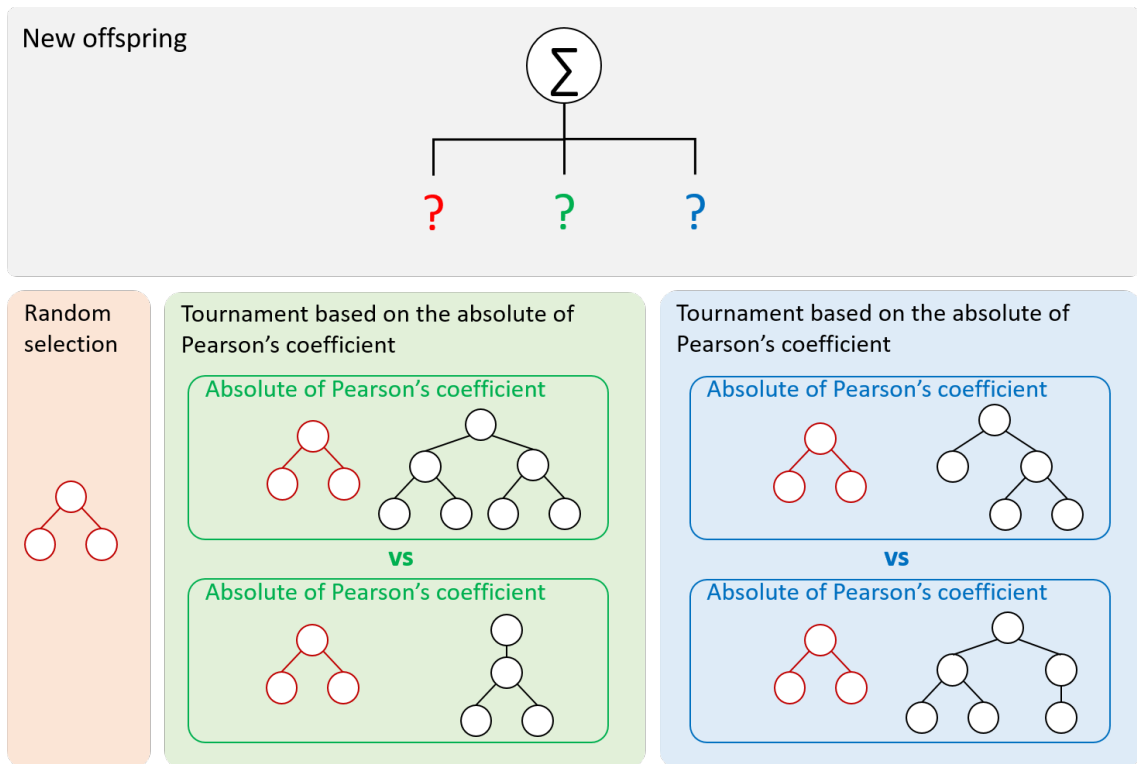


Figure 3.8: Diagram of parent selection based on Pearson's coefficient (in EvoDAG). Source: Own elaboration.

3.2.4 Parent Selection based on the Accuracy (acc)

Parent Selection based on Accuracy

It consists of selecting parents minimizing the accuracy among them. The idea is to select parents with different behaviors. This heuristic is designed based on the properties of the function Σ and the classifiers Naive Bayes (NB and MN) and Nearest Centroid (NC).

$$acc(\vec{P}_1, \vec{P}_2) = \frac{1}{n} \sum_i \delta(P_{1i} == P_{2i})$$

As we explained in Section 1.4, each EvoDAG individual contains an array of semantics vectors, one per class, where each vector contains the semantics for a specific class. A value close to 1 in the i -th entry of the j -th semantic vector represents that the i -th input vector belongs to the class j . Opposite, a value of -1 represents that the input vector do not belongs to that class. Further, for each input vector, the assigned

class corresponds to the one with the highest value in the semantics vectors. Therefore, another way for representing the individuals' semantics is by using a vector with the assigned labels to the input-vectors. We call this vector *labels vector*.

On the other hand, accuracy is widely used for calculating the performance of classifiers. It counts the number of correct prediction between the target and the classifier. If \hat{y}_i and y_i are the predicted and the real value of the i -th sample, the accuracy over n input vectors is defined as $accuracy(y, \hat{y}) = \frac{1}{n} \sum_i \delta(\hat{y}_i == y_i)$, where n is the number of samples, and $\delta(\cdot)$ returns 1 if its input is true and 0 otherwise.

Our selection heuristic based on accuracy aims to select parents whose semantics are different, and this can be measured using as metric the accuracy among individuals' labels vectors. Let be v_1 and v_2 the labels vectors of two individuals, the accuracy among them is defined as Equation 3.6, where n is the number of input vectors, and $\delta(\cdot)$ returns 1 if its input is true and 0 otherwise.

$$acc(v_1, v_2) = \frac{1}{n} \sum_i \delta(v_{1i} == v_{2i}) \quad (3.6)$$

In Subsection 3.2.3, we described that Naive Bayes and Nearest Centroid work better with independent variables, if we use the accuracy between the labels vectors of two individuals, we can say that two individuals' semantics are correlated among them if their accuracy is equals to 1. In addition, the smaller the value, the less the correlation among individuals' semantics. On the other hand, for the sum function, if two identical or similar individuals are combined, the resultant one will be almost the same as its parents. The objective of this heuristic is to select individuals whose accuracy among their labels vectors is the smallest.

The selection of arguments based on accuracy is described with a simple example showed in Figure 3.9. In this case, the function Σ is selected for creating the offspring and it needs three arguments. The first parent is randomly selected from the

population \mathcal{P} . Then, the next arguments are independently selected. For each one, a tournament is performed, which is described as follows. First, two individuals are randomly selected from the population \mathcal{P} , and the accuracy between their labels vectors and the first argument labels vector is calculated. The individual with the minimum accuracy between its labels vector and the first parent labels vector is selected as argument. Algorithm 7 describes the selection of arguments based on accuracy in EvoDAG.

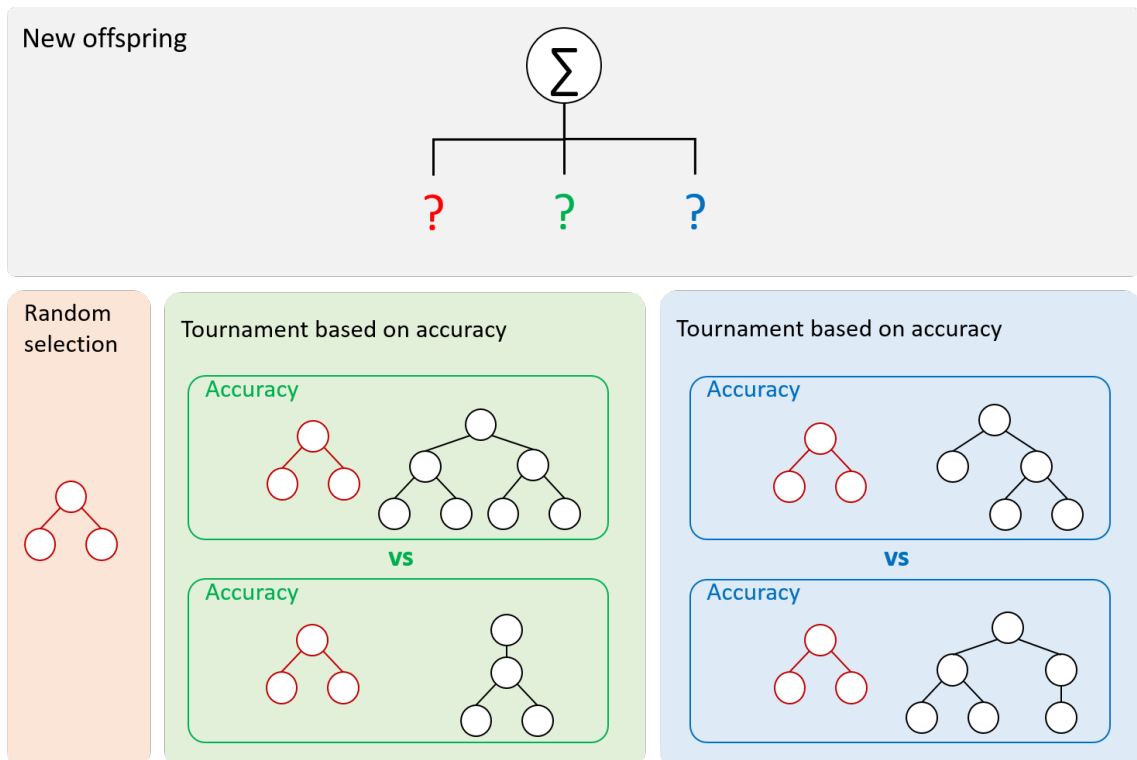


Figure 3.9: Diagram of parent selection based on accuracy (in EvoDAG). Source: Own elaboration.

3.3 Negative Selection

Negative selection takes place when the new individual, the offspring, needs to be inserted in the population. In this case, an individual is selected to be replaced by the new one. The classical negative selection scheme consists of choosing a non-fitted individual. In GP, generally, a tournament selection where the worst individual is selected for being replaced is used for this purpose. Besides, we compare this technique with random selection, because it is the most straightforward and less expensive strategy.

Algorithm 7: Selection of arguments based on the accuracy in EvoDAG

Input: k , the number of arguments

Input: ts , the tournament size

Output: The list of individuals that will be passed as arguments

$args \leftarrow$ an empty list of individuals;

$F \leftarrow$ an individual randomly selected from the population \mathcal{P} ;

$labels_F \leftarrow$ the labels vector of individual F ;

for i **from** 2 **to** k **do**

$A \leftarrow$ an individual randomly selected from the population \mathcal{P} ;

$labels_A \leftarrow$ the labels vector of individual A ;

$acc_A \leftarrow \frac{1}{n} \sum_j \delta(labels_{Fj} == labels_{Aj})$;

for j **from** 2 **to** ts **do**

$B \leftarrow$ an individual randomly selected from the population \mathcal{P} ;

$labels_B \leftarrow$ the labels vector of individual B ;

$acc_B \leftarrow \frac{1}{n} \sum_j \delta(labels_{Fj} == labels_{Bj})$;

if $acc_B < acc_A$ **then**

$A \leftarrow B$;

$acc_A \leftarrow acc_B$;

end

end

 Add the individual A to $args$;

end

Return $args$;

This section is divided into two subsections. The first one describes the negative selection based on fitness, and the second one, random negative selection.

3.3.1 Negative Selection based on Fitness (fit)

Negative Selection based on Fitness

It tries to remove non-fitted individuals from the population. Those are replaced by the new offspring.

The objective of negative selection based on fitness is to remove non-fitted individuals from the population \mathcal{P} with the idea of improving the quality of individuals generation by generation. EvoDAG performs this scheme using negative tournament selection. ts individuals are randomly chosen from the population \mathcal{P} , and the worst of them, based on fitness, is the one which is removed from the population \mathcal{P} and the new offspring takes its place. Algorithm 8 shows this selection scheme.

Algorithm 8: Negative selection of individuals based on fitness

Input: ts , the tournament size
Output: The individual that will be removed from the population \mathcal{P}
 $A \leftarrow$ an individual randomly selected from the population \mathcal{P} ;
 $fitness_A \leftarrow$ fitness of A ;
for i **from** 2 **to** ts **do**
 $B \leftarrow$ an individual randomly selected from the population \mathcal{P} ;
 $fitness_B \leftarrow$ fitness of B ;
 if $fitness_B < fitness_A$ **then**
 $A \leftarrow B$;
 $fitness_A \leftarrow fitness_B$;
 end
end
Return A ;

3.3.2 Random Negative Selection (rnd)

Random Negative Selection

It randomly removes individuals from the population. Those are replaced by the new offspring.

Random negative selection aims to perform negative selection randomly because it is the most straightforward. Besides, as in random parent selection, we want to see what happens if the negative selection is performed randomly. Algorithm 9 shows this selection scheme. It can be seen that random negative selection is much simpler than negative selection based on fitness.

Algorithm 9: Random negative selection

Output: The individual that will be removed from the population \mathcal{P}

$A \leftarrow$ an individual randomly selected from the population \mathcal{P} ;

Return A ;

3.4 Summary

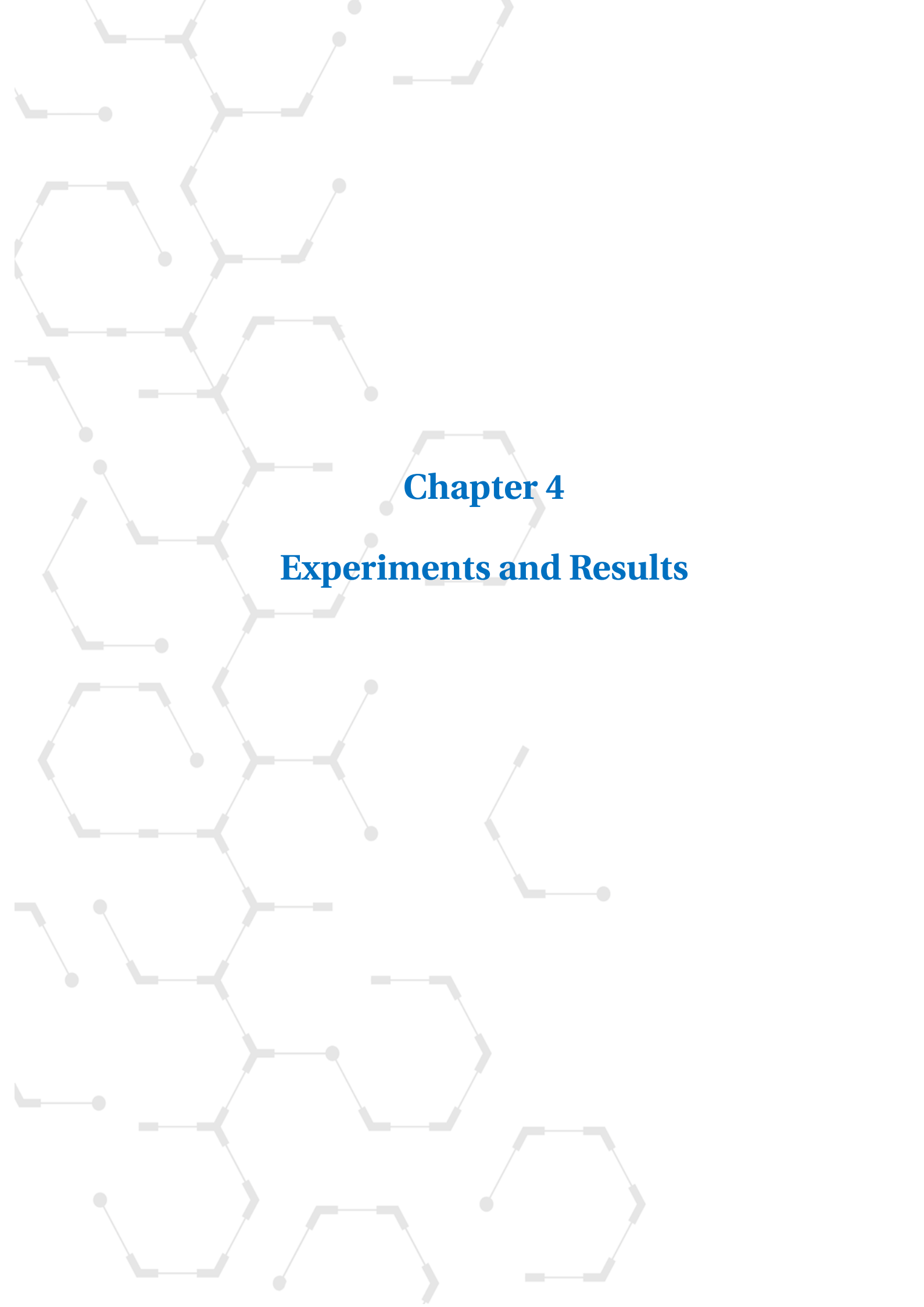
This chapter presented the main contribution of this dissertation: the proposed selection heuristics and their motivation. We divided the selection schemes into two groups: parent selection and negative selection. In GP, **parent selection** consists of selecting the parents of the new offspring. In EvoDAG, it can be seen as the selection of the arguments for a function. **Negative selection** is when an individual needs to be chosen for being removed from the population, and the new offspring takes its place. For both selection steps, the classical selection scheme is based on fitness, where the best (for parent selection) and the worst (for negative selection) individual is selected from a group of individuals randomly chosen from the population.

For guiding parent selection, we proposed the following selection heuristics that use functions' properties, in addition to individuals' semantics. The heuristics

were designed for solving classification problems.

- *Parent selection based on cosine similarity (sim)* that consists of selecting individuals whose semantics' vectors ideally have rectangle angles. The absolute cosine similarity was used to measure the angle between individuals' semantics. This heuristic was designed based on the properties of the function Σ and the classifiers Naive Bayes (NB and MN) and Nearest Centroid (NC).
- *Parent selection based on Pearson's correlation coefficient (prs)* that aims to select parents whose semantics vectors are uncorrelated among them. We used the absolute Pearson's correlation coefficient for measuring the correlation among inputs. This heuristic was designed based on the properties of the function Σ and the classifiers Naive Bayes (NB and MN) and Nearest Centroid (NC).
- *Parent selection based on the accuracy (acc)* that consists of selecting parents with the objective of minimizing the accuracy among their labels vectors. The labels vector of an individual is the vector whose entries correspond to the labels that the individual assigns to all the input vectors. The idea is to select parents with different behaviors. This heuristic was designed based on the properties of the function Σ and the classifiers Naive Bayes (NB and MN) and Nearest Centroid (NC).

Besides, we proposed to test the use of random selection in both stages: parent selection and negative selection.



Chapter 4
Experiments and Results

4 Experiments and Results

In this chapter, we use thirty classification datasets taken from the UCI repository for testing the performance of our proposed selection heuristics. They are described in Section 4.1. The datasets were selected to be heterogeneous in terms of the number of samples, variables, and classes. Two measures are used for analyzing the performance of classifiers: macro-F1 and time (in seconds) per sample. They are described in detail in Section 4.3.

In Section 4.4, we show and compare the performance of EvoDAG with different selection techniques. For parent selection, we analyze the results of the proposed selection heuristics against random selection or tournament based on fitness, the most used technique. Those parent selection schemes were combined with random negative selection or negative selection based on fitness. The performance of different parent and negative selection combinations was analyzed in terms of macro-F1 and time. As we mentioned in the previous chapter, our heuristics were designed to be used in the function Σ and the classifiers Naive Bayes (NB and MN) and Nearest Centroid (NC), then, we analyze what happens if our heuristics were used in all functions that need more than one argument (Σ , Π , max, min, hypot, NB, MN, and NC). Besides, we show the results of the different selection schemes taking into account the properties of the datasets: class imbalance, number of variables, and number of samples.

We also compare our heuristics against state-of-the-art selection schemes. In Chapter 2, we said that Angle-Driven-Selection (ADS) [15] and Novelty Search (NVS) [62] had been recently proposed as new selection schemes. For that reason, we decided to implement those techniques in EvoDAG for comparing our heuristics with them. In Section 4.5, first, we explain the main idea of those schemes and how they were implemented in EvoDAG. Finally, we present the results of the comparison of ADS and NVS against our results in terms of macro-F1 and time.

Besides, in Section 4.6, we present a comparison of EvoDAG plus our heuristics against different state-of-the-art classifiers. We decided to compare EvoDAG against sixteen classifiers of the scikit-learn python library [80]. Specifically, these classifiers are Perceptron, MLPClassifier, BernoulliNB, GaussianNB, KNeighborsClassifier, NearestCentroid, LogisticRegression, LinearSVC, SVC, SGDClassifier, PassiveAggressiveClassifier, DecisionTreeClassifier, ExtraTreesClassifier, RandomForestClassifier, AdaBoostClassifier and GradientBoostingClassifier. It is also included in the comparison two auto-machine learning libraries: autosklearn [23] and TPOT [76].

In all cases, we employ the Wilcoxon statistical test for validating our results.

Finally, we use our visualization technique, Liking Product Landscape [87], for analyzing the macro-F1 results of selection schemes and classifiers. The idea is to perform a comparison based on datasets and classification techniques.

4.1 Datasets

The classification problems used as benchmarks are thirty datasets taken from the UCI repository [20]. Table 4.1 shows the datasets' information. It can be seen that the datasets are heterogeneous in terms of the number of samples, variables, and classes. Additionally, some of the classification problems are balanced, and others are imbalanced.

We use Shannon's entropy to indicate the degree of the class-imbalance in the problem. It is defined as $H(X) = -\sum_i p_i \log(p_i)$, where p_i represents the probability of the category i . We calculate those probabilities by counting the frequencies of each category. Besides, for normalizing, we use the logarithm base the number of categories. For example, if the classification problem has four categories, we calculate the Shan-

Table 4.1: Datasets used to compare the performance of the algorithms. These problems are taken from the UCI repository. The table includes Shannon’s entropy to indicate the degree of the class-imbalance, where the value 1.0 indicates that the samples are perfectly balanced. In opposite, the smaller the value, the bigger the imbalance.

Dataset	Train samples	Test samples	Variables	Classes	Classes entropy
ad	2295	984	1557	2	0.58
adult	32561	16281	14	2	0.8
agaricus-lepiota	5686	2438	22	7	0.81
aps-failure	60000	16000	170	2	0.12
banknote	960	412	4	2	0.99
bank	31647	13564	16	2	0.52
biodeg	738	317	41	2	0.91
car	1209	519	6	4	0.6
census-income	199523	99762	41	2	0.34
cmc	1031	442	9	3	0.98
dota2	92650	10294	116	2	1.0
drug-consumption	1319	566	30	7	0.44
fertility	69	30	9	2	0.43
IndianLiverPatient	407	175	10	2	0.85
iris	105	45	4	3	1.0
krkopt	19639	8417	6	18	0.84
letter-recognition	14000	6000	16	26	1.0
magic04	13314	5706	10	2	0.93
ml-prove	4588	1530	56	2	0.98
musk1	333	143	166	2	0.99
musk2	4618	1980	166	2	0.61
optdigits	3823	1797	64	10	1.0
page-blocks	3831	1642	10	5	0.27
parkinsons	135	59	22	2	0.79
pendigits	7494	3498	16	10	1.0
segmentation	210	2100	19	7	1.0
sensorless	40956	17553	48	11	1.0
tae	105	45	5	3	0.99
wine	123	53	13	3	0.99
yeast	1038	446	9	10	0.76

non's entropy as $H(X) = -\sum_i p_i \log_4(p_i)$. In this sense, if the value is equal to 1.0, it indicates a perfect balanced problem. In contrast, the smaller the value, the bigger the imbalance.

The performance of the classifiers is measured in a test set. In the repository, some of the problems are already split between a training set and a test set. For those problems that this partition is not present, we performed cross-validation; that is, we randomly split the dataset using the 70% of the samples for the training set, and 30% for the test set.

4.2 Computer Equipment

The characteristics of the computer where the experiments were executed are shown in Table 4.2. For a valid comparison, the experiments were executed using only one core.

Table 4.2: Characteristics of the computer where the experiments were executed

Operating system	Ubuntu 16.04.2 LTS
Processor (CPU)	Intel (R) Xeon(R) CPU E5-2680 v4
Processor (CPU) speed	2.5GHz
Computer memory size	256 GB
Hard disk size	1 TB
Cores number	14

4.3 Performance Metrics

For analyzing the performance of classifiers from the point of view of precision and the time that they expend training the model, we use two metrics: macro-F1 and time (in seconds) per sample.

Accuracy is maybe the most used metric for measuring the performance of classifiers. Its value ranges from 0 to 1, being one the best performance and zero the worst.

It can be seen as the percentage of samples that are correctly predicted. However, if the classes are imbalanced, as the problems in this experiment, accuracy can not be very reliable. Imagine that you have a problem where the 95% of the samples belong to class 1, and only the 5% belong to class 2, if a classifier assigns to all samples the label that corresponds to class 1, it will get a high value on accuracy, 0.95, but this classifier does not do anything worthy. On the other hand, the F1 score, also known as balanced F-score, measures the performance of a binary classifier taking into account that it be able to predict both classes. It is robust to imbalanced problems. F1 score, for a binary classification problem, is defined as $F1 = 2 \frac{precision \cdot recall}{precision + recall}$, where $recall = \frac{tp}{tp + fn}$, $precision = \frac{tp}{tp + fp}$, tp is the number of true positives, fp the number of false positives, and fn the number of false negatives. For a multi-class problem, the F1 score can be extended as the macro-F1 score that corresponds to the average of the F1 score per class. Besides, most of the comparisons are performed based on the rank of macro-F1. It means, for each dataset, the classifiers are ranked according to their performance. The number 1 is assigned to the classifier with the highest value in macro-F1, number 2 corresponds to the one with the second-highest value in macro-F1, and so on. In the case of several classifiers have the same value in macro-F1, they got the same rank, and the next rank number will be increased the number of repeated values. For example, if the 5 classifiers have the following macro-F1 values: a-0.97, b-0.94, c-0.94, d-0.94, and, e-0.89, their ranks will be: a-1, b-2, c-2, d-2, e-5.

In addition to the performance of a classifier for predicting the samples correctly, time is an essential factor in algorithms. When the number of samples in the training set is small, generally, all the algorithms learn the model very fast. However, if the number of samples grows, some algorithms spend considerably more and more time, and in some cases, it could be impossible to wait until the algorithm converges. As we mention in the previous section, the datasets vary on the number of samples, and, logically, algorithms spend more time learning big datasets. In that sense, to normalize the time, and with the idea of making comparisons based on this measure, we divided the time (in seconds) that algorithms spend in the training phase by the num-

ber of samples in the datasets.

4.4 Comparison of the Proposed Selection Heuristics and Classic Selection Techniques

In this section, we show a comparison of our proposed selection heuristics for parent selection against random selection and the classical selection technique based on fitness. In other words, for **parent selection**, we compare the use of the following techniques: (1) tournament selection based on fitness (*fit*), (2) random selection (*rnd*), and our proposed selection heuristics, (3) tournament selection based on the absolute of cosine similarity (*sim*), (4) tournament selection based on the absolute of Pearson's correlation coefficient (*prs*), and, (5) tournament selection based on accuracy (*acc*). Besides, for **negative selection**, we analyze the use of the classical scheme, negative tournament selection based on fitness (*fit*), and, random selection (*rnd*). All these selection techniques were described in Chapter 3.

We used EvoDAG for testing different combinations of parent and negative selection schemes. To improve the reading of tables and figures, we use the following notation. The selection scheme that is used for parent selection is followed by the symbol “-”, and then comes the abbreviation of the negative selection scheme. For example, *sim-fit* means that EvoDAG uses a tournament based on the absolute of cosine similarity for parent selection and negative tournament based on fitness for negative selection. In total, we analyze the performance of eight combinations: *fit-fit*, *rnd-rnd*, *sim-fit*, *sim-rnd*, *prs-fit*, *prs-rnd*, *acc-fit*, and, *acc-rnd*. Furthermore, for analyzing whether the behavior of EvoDAG is better when the heuristics are applied for the functions that they were designed for (Σ , Naive Bayes, NB and MN, and Nearest Centroid NC) than using them in all functions with more than one argument (Σ , Π , max, min, hypot, NB, MN, and NC), we tested our heuristics applying them to all functions with more than one argument. We identify those versions with the symbol *. For example, *acc-rnd*

indicates that the parent selection heuristic based on accuracy is applied only to the functions Σ , NB, MN, and NC, however, acc-rnd* means that the heuristic is applied to the functions: Σ , Π , max, min, hypot, NB, MN, and NC.

Tables 4.3, 4.4, and Figure 4.1 shows the EvoDAG's performance for the classification task using different techniques for parent and negative selection. Table 4.3 and Figure 4.1.a show the results based on macro-F1 and macro-F1 ranks, respectively, over the test sets. It can be seen that the best performance can be reached when the heuristic based on accuracy for parent selection and random negative selection (acc-rnd) are used, followed by the use of the same scheme for parent selection and negative tournament selection based on fitness (acc-fit). The combinations acc-rnd, acc-fit, sim-fit, prs-fit*, prs-rnd, prs-fit, prs-rnd*, sim-fit*, sim-rnd, and, sim-rnd* are better than fit-fit in terms of macro-F1 average rank. It means that the use of our proposed heuristics improves the performance of the classical selection schemes, fit-fit. Surprisingly, random selection for parent and negative selection, rnd-rnd, also improves the performance of EvoDAG using the classical selection schemes based on fitness (fit-fit). Besides, in general, the use of negative random selection is better than negative selection based on fitness. Moreover, there is a trend that when the heuristics are applied to all functions with more than one argument (Σ , Π , max, min, hypot, NB, MN, and NC), the combinations with the symbol *, the results are worse than when they are applied only to the functions that they were designed for (Σ , Naive Bayes and Nearest Centroid). Based on the colors of Table 4.3, it can be observed that acc-rnd* and acc-fit* produce considerably worse results than the rest of the combinations. The results over the training sets (Table 4.4 and Figure 4.1.b) are similar to the ones in the test sets (Table 4.3 and Figure 4.1.a), but in this case prs-rnd is the one that produces the best results in the training sets. Comparing by the time that the classifiers spend in the training phase (Figure 4.1.c), it can be seen that rnd-rnd is the fastest, this is because it is the most straightforward.

Table 4.3: Comparison of EvoDAG’s performance using different techniques for parent and negative selection. The values represent the macro-F1 value obtained over the test sets. Columns are ordered based on the macro-F1 average rank. The symbol * represents that the heuristic for parent selection was applied to all functions with arity greater than one. The best performance in each problem is indicated in boldface. Colors indicate the performance of the selection techniques combination, the bluer, the better, in the opposite, the more gray, the worse.

	acc-rnd	acc-fit	rnd-rnd	sim-fit	prs-fit*	prs-rnd	prs-fit	prs-rnd*	sim-fit*	sim-rnd	sim-rnd*	fit-fit	acc-rnd*	acc-fit*
ad	0.940	0.934	0.932	0.933	0.928	0.933	0.932	0.921	0.934	0.934	0.923	0.934	0.554	0.548
adult	0.793	0.792	0.792	0.791	0.790	0.791	0.793	0.792	0.791	0.791	0.791	0.791	0.685	0.685
agaricus-lepiota	0.684	0.684	0.682	0.684	0.683	0.682	0.685	0.682	0.684	0.682	0.682	0.677	0.037	0.037
aps-failure	0.828	0.840	0.864	0.839	0.838	0.830	0.836	0.850	0.839	0.825	0.836	0.847	0.736	0.736
banknote	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	0.821	0.821
bank	0.762	0.760	0.765	0.763	0.760	0.762	0.762	0.760	0.758	0.762	0.762	0.756	0.713	0.713
biodeg	0.841	0.835	0.851	0.842	0.834	0.823	0.849	0.834	0.842	0.838	0.834	0.829	0.626	0.626
car	0.866	0.914	0.860	0.829	0.870	0.848	0.868	0.858	0.885	0.830	0.839	0.843	0.294	0.294
census-income	0.767	0.510	0.770	0.767	0.768	0.767	0.511	0.765	0.767	0.765	0.767	0.753	0.419	0.419
cmc	0.536	0.548	0.526	0.537	0.541	0.549	0.547	0.550	0.547	0.534	0.541	0.531	0.467	0.467
dota2	0.595	0.594	0.594	0.595	0.594	0.592	0.593	0.594	0.594	0.594	0.594	0.595	0.474	0.475
drug-consumption	0.227	0.204	0.197	0.224	0.229	0.209	0.209	0.214	0.211	0.223	0.210	0.196	0.203	0.203
fertility	0.453	0.453	0.453	0.453	0.453	0.453	0.442	0.453	0.453	0.453	0.453	0.442	0.396	0.396
IndianLiverPatient	0.661	0.711	0.694	0.649	0.654	0.639	0.654	0.667	0.657	0.642	0.647	0.642	0.631	0.631
iris	0.980	0.980	0.980	0.980	0.980	0.980	0.960	0.980	0.960	0.980	0.960	0.980	0.960	0.960
krkopt	0.188	0.183	0.198	0.143	0.160	0.160	0.185	0.164	0.145	0.142	0.145	0.147	0.124	0.124
letter-recognition	0.653	0.652	0.658	0.646	0.646	0.646	0.652	0.646	0.646	0.646	0.646	0.646	0.646	0.646
magic04	0.848	0.849	0.846	0.847	0.843	0.848	0.849	0.844	0.839	0.846	0.840	0.834	0.655	0.655
ml-prove	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	0.999	0.704	0.704
musk1	0.883	0.856	0.867	0.877	0.890	0.883	0.860	0.876	0.868	0.891	0.897	0.862	0.366	0.366
musk2	0.943	0.906	0.940	0.942	0.942	0.945	0.942	0.937	0.938	0.948	0.936	0.951	0.455	0.455
optdigits	0.953	0.949	0.955	0.946	0.953	0.947	0.955	0.950	0.943	0.946	0.944	0.945	0.733	0.733
page-blocks	0.825	0.791	0.761	0.772	0.803	0.804	0.793	0.805	0.765	0.773	0.766	0.773	0.757	0.757
parkinsons	0.747	0.752	0.730	0.734	0.734	0.718	0.653	0.734	0.701	0.718	0.718	0.667	0.672	0.672
pendigits	0.945	0.948	0.940	0.933	0.940	0.950	0.940	0.937	0.915	0.933	0.918	0.937	0.802	0.802
segmentation	0.910	0.904	0.909	0.896	0.917	0.897	0.912	0.915	0.889	0.897	0.890	0.902	0.816	0.816
sensorless	0.959	0.966	0.953	0.953	0.963	0.959	0.963	0.957	0.958	0.948	0.952	0.965	0.796	0.796
tae	0.361	0.315	0.321	0.419	0.299	0.323	0.418	0.321	0.383	0.299	0.471	0.438	0.321	0.321
wine	0.982	0.982	0.982	0.982	0.962	0.979	0.979	0.962	0.982	0.982	1.000	0.982	0.982	0.982
yeast	0.468	0.454	0.455	0.457	0.458	0.452	0.430	0.450	0.460	0.444	0.445	0.456	0.457	0.457
Average macro-F1	0.753	0.742	0.749	0.748	0.748	0.746	0.739	0.747	0.745	0.742	0.747	0.744	0.577	0.577
Average rank	3.6	4.9	5.1	5.4	5.7	5.9	5.9	5.9	6.3	6.8	7.2	7.6	11.7	11.7

Table 4.4: Comparison of EvoDAG’s performance using different techniques for parent and negative selection. The values represent the macro-F1 value obtained over the training sets. Columns are ordered based on the macro-F1 average rank. The symbol * represents that the heuristic for parent selection was applied to all functions with arity greater than one. The best performance in each problem is indicated in boldface. Colors indicate the performance of the selection techniques combination, the bluer, the better, in the opposite, the more gray, the worse.

	prs-rnd	acc-rnd	prs-fit	prs-fit*	sim-fit	acc-fit	prs-rnd*	rnd-rnd	fit-fit	sim-rnd	sim-rnd*	sim-fit*	acc-rnd*	acc-fit*
ad	0.944	0.946	0.942	0.941	0.943	0.932	0.928	0.940	0.942	0.937	0.933	0.940	0.530	0.524
adult	0.803	0.800	0.801	0.802	0.802	0.801	0.803	0.801	0.802	0.802	0.800	0.802	0.688	0.688
agaricus-lepiota	0.703	0.704	0.703	0.703	0.702	0.702	0.706	0.707	0.697	0.704	0.699	0.700	0.035	0.035
aps-failure	0.805	0.808	0.806	0.805	0.806	0.815	0.828	0.843	0.820	0.799	0.818	0.814	0.711	0.711
banknote	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	0.858	0.858
bank	0.761	0.759	0.759	0.759	0.760	0.757	0.759	0.760	0.754	0.759	0.758	0.758	0.697	0.697
biodeg	0.906	0.915	0.912	0.908	0.913	0.902	0.904	0.902	0.904	0.910	0.906	0.905	0.645	0.645
car	0.869	0.878	0.884	0.866	0.825	0.912	0.857	0.851	0.874	0.839	0.864	0.872	0.283	0.283
census-income	0.511	0.510	0.509	0.765	0.766	0.763	0.509	0.512	0.501	0.765	0.509	0.509	0.418	0.418
cmc	0.620	0.612	0.611	0.597	0.628	0.597	0.598	0.610	0.604	0.631	0.623	0.629	0.475	0.475
dota2	0.602	0.601	0.601	0.602	0.601	0.601	0.602	0.602	0.602	0.602	0.602	0.601	0.479	0.477
drug-consumption	0.432	0.454	0.430	0.461	0.432	0.408	0.452	0.422	0.413	0.431	0.477	0.432	0.399	0.399
fertility	0.822	0.822	0.822	0.822	0.822	0.734	0.822	0.892	0.822	0.951	0.892	0.951	0.634	0.634
IndianLiverPatient	0.695	0.677	0.690	0.708	0.683	0.690	0.685	0.664	0.727	0.671	0.674	0.696	0.593	0.593
iris	0.979	0.979	0.969	0.979	0.959	0.979	0.979	0.969	0.969	0.959	0.979	0.959	0.959	0.959
krkopt	0.160	0.191	0.190	0.161	0.140	0.193	0.162	0.205	0.147	0.138	0.143	0.143	0.117	0.117
letter-recognition	0.650	0.655	0.658	0.650	0.650	0.655	0.650	0.660	0.650	0.650	0.650	0.650	0.650	0.650
magic04	0.857	0.856	0.857	0.850	0.852	0.857	0.852	0.854	0.842	0.854	0.848	0.849	0.658	0.658
ml-prove	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	0.697	0.697
musk1	0.982	0.972	0.969	0.975	0.976	0.960	0.982	0.972	0.979	0.976	0.969	0.969	0.360	0.360
musk2	0.969	0.966	0.970	0.966	0.970	0.928	0.969	0.966	0.973	0.974	0.967	0.967	0.459	0.459
optdigits	0.982	0.979	0.983	0.983	0.973	0.975	0.982	0.981	0.979	0.972	0.972	0.974	0.742	0.742
page-blocks	0.776	0.800	0.786	0.790	0.767	0.796	0.798	0.765	0.748	0.744	0.738	0.738	0.702	0.702
parkinsons	0.871	0.886	0.907	0.862	0.886	0.979	0.851	0.829	0.840	0.859	0.901	0.909	0.763	0.763
pendigits	0.983	0.984	0.983	0.983	0.979	0.986	0.983	0.982	0.983	0.979	0.973	0.973	0.869	0.869
segmentation	0.931	0.946	0.956	0.941	0.937	0.947	0.941	0.931	0.942	0.932	0.926	0.917	0.818	0.818
sensorless	0.965	0.959	0.963	0.963	0.959	0.966	0.957	0.958	0.965	0.953	0.958	0.958	0.798	0.798
tae	0.773	0.696	0.736	0.733	0.761	0.812	0.728	0.637	0.829	0.720	0.762	0.784	0.490	0.490
wine	0.984	0.992	1.000	0.992	0.992	1.000	0.992	0.992	0.984	0.992	0.992	0.992	0.984	0.984
yeast	0.580	0.594	0.570	0.575	0.593	0.571	0.562	0.577	0.569	0.590	0.574	0.569	0.539	0.540
Average macro-F1	0.797	0.798	0.799	0.805	0.803	0.807	0.795	0.793	0.795	0.803	0.797	0.799	0.602	0.602
Average rank	4.6	4.8	5.0	5.4	5.7	5.8	5.8	6.2	6.5	6.5	7.0	7.0	12.6	12.6

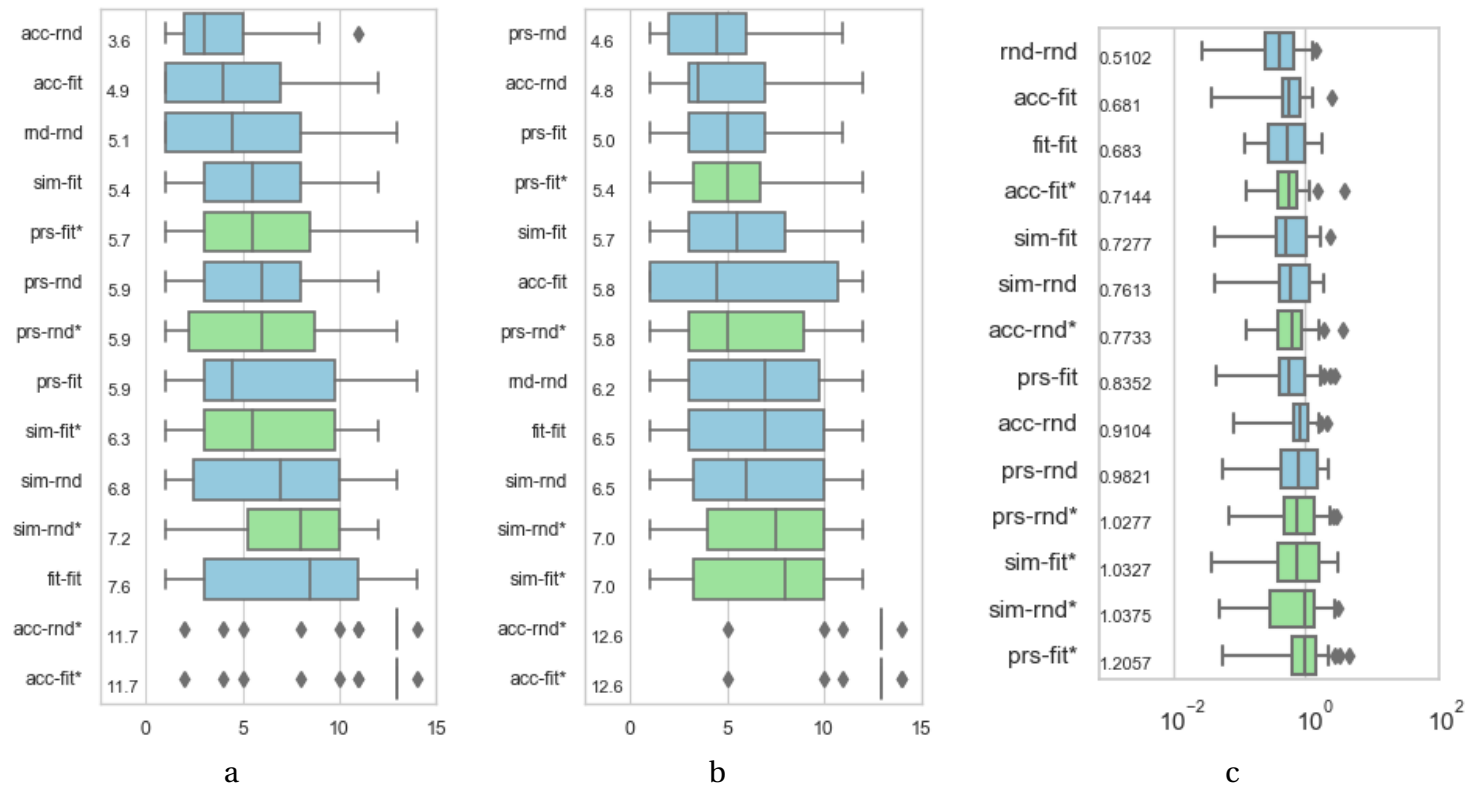


Figure 4.1: Selection schemes comparison. In a) and b), boxplots present the macro-F1 ranks measured over the test (a) and train (b) datasets. c) Boxplots present the time, in seconds, required by EvoDAG using the different selection techniques combinations in the training phase. The time is divided by the number of train samples in the datasets. In all figures, green boxplots represent where the selection heuristics (sim, prs, or acc) are applied to all functions. The average rank, or time per sample, sorts classifiers, and it appears on the left. Source: Own elaboration.

In Figure 4.2, an analysis in two dimensions is performed. The x-axis represents the performance of EvoDAG's and the selection schemes combinations using the macro-F1 average rank. The y-axis represents the average time per sample (in seconds) that EvoDAG spends on the training phase. We analyze the results using the technique Pareto frontier, which is commonly used when we tried to reach two objectives. In this image, the closest the technique to the origin, the better it is in terms of performance and time. We can see that the classifiers that use the selection heuristics in all functions with more than one argument, the ones with the symbol *, are further to the origin. It is because they spend much time in the training phase, and their performance, based on macro-F1 average rank, is poor. The classifiers that are part of the Pareto frontier are acc-rnd, acc-fit and rnd-rnd. All of them are better than the classical selection scheme based on fitness (fit-fit), in performance, macro-F1 average rank, and time.

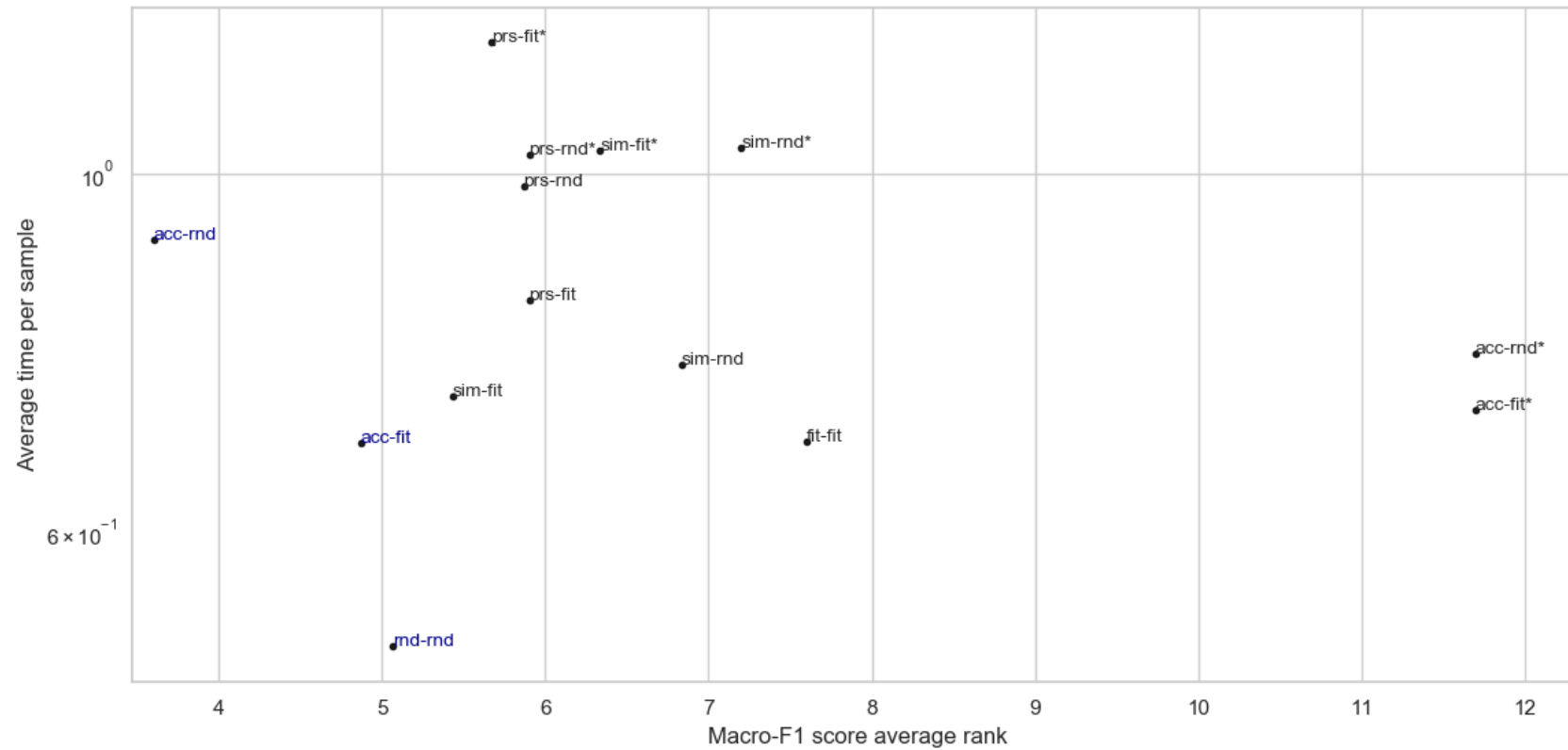


Figure 4.2: Comparison of the different selection schemes combinations based on macro-F1 average rank and the time, in seconds, required by EvoDAG's training phase. The time is presented in seconds, and it is the average time per sample. Source: Own elaboration.

The statistical test used was Wilcoxon signed-rank test [102], and the p -values were adjusted with the Holm-Bonferroni method [44] to consider multiple comparisons. Macro-F1 values were used for the statistical test. There was found that the best combination of selection schemes, acc-rnd, is statistically better than prs-rnd, prs-rnd*, sim-rnd, fit-fit, acc-rnd*, and acc-fit*, with a confidence of 95%. It indicates that the combination of our proposed heuristic based on accuracy for parent selection and random negative selection (acc-rnd) performs statistically better than the classical selection schemes based on fitness. In addition, Figure 4.3 shows the results of Wilcoxon signed-rank test [102] by pairs of techniques. It confirms that acc-rnd is statistically better than fit-fit with a confidence of 95%.

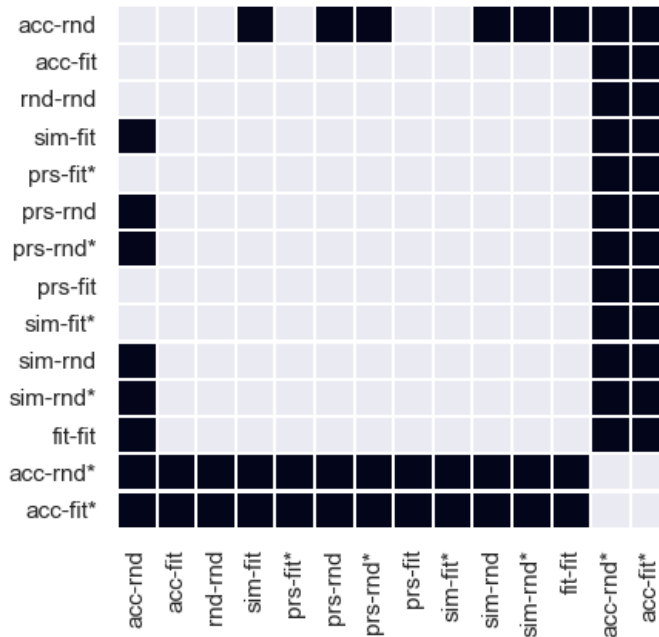


Figure 4.3: Statistical comparison (Wilcoxon signed-rank test) of the different selection schemes combinations based on macro-F1. Black cells represent that the pair of schemes are statistically different with a 95% confidence. Source: Own elaboration.

An analysis of the behavior of the selection schemes combinations taking into account the datasets' properties, as classes' entropy, number of classes and number of variables, is presented in Tables 4.5, 4.6 and 4.7, respectively. The tables show the macro-F1 results, but the dataset' property orders the rows. If the results of combinations depended on the properties of the dataset, we could see patterns in the columns.

For example, if one combination were better when the value of classes' entropy is smaller, we could see that the column corresponding to that combination started with green, and then it became red. That is not the case of any of the combinations in the Tables 4.5, 4.6 and 4.7. For that reason, we can affirm that the combinations of selection schemes do not depend on datasets' properties.

Table 4.5: Comparison of EvoDAG's performance using different techniques for parent and negative selection, ordered by classes' entropy of the datasets. The values represent the macro-F1 value obtained over the test sets. Rows and columns are ordered by classes' entropy, and macro-F1 average rank, respectively. The best performance in each problem is indicated in boldface. Colors indicate the performance of the selection techniques combination, the bluer, the better, in the opposite, the more gray, the worse.

	Classes' entropy	acc-rnd	acc-fit	rnd-rnd	sim-fit	prs-fit	prs-rnd	sim-rnd	fit-fit
aps-failure	0.12	0.828	0.840	0.864	0.839	0.836	0.830	0.825	0.847
page-blocks	0.27	0.825	0.791	0.761	0.772	0.793	0.804	0.773	0.773
census-income	0.34	0.767	0.510	0.770	0.767	0.511	0.767	0.765	0.753
fertility	0.43	0.453	0.453	0.453	0.453	0.442	0.453	0.453	0.442
drug-consumption	0.44	0.227	0.204	0.197	0.224	0.209	0.209	0.223	0.196
bank	0.52	0.762	0.760	0.765	0.763	0.762	0.762	0.762	0.756
ad	0.58	0.940	0.934	0.932	0.933	0.932	0.933	0.934	0.934
car	0.6	0.866	0.914	0.860	0.829	0.868	0.848	0.830	0.843
musk2	0.61	0.943	0.906	0.940	0.942	0.942	0.945	0.948	0.951
yeast	0.76	0.468	0.454	0.455	0.457	0.430	0.452	0.444	0.456
parkinsons	0.79	0.747	0.752	0.730	0.734	0.653	0.718	0.718	0.667
adult	0.8	0.793	0.792	0.792	0.791	0.793	0.791	0.791	0.791
agaricus-lepiota	0.81	0.684	0.684	0.682	0.684	0.685	0.682	0.682	0.677
krkopt	0.84	0.188	0.183	0.198	0.143	0.185	0.160	0.142	0.147
IndianLiverPatient	0.85	0.661	0.711	0.694	0.649	0.654	0.639	0.642	0.642
biodeg	0.91	0.841	0.835	0.851	0.842	0.849	0.823	0.838	0.829
magic04	0.93	0.848	0.849	0.846	0.847	0.849	0.848	0.846	0.834
ml-prove	0.98	1.000	1.000	1.000	1.000	1.000	1.000	1.000	0.999
cmc	0.98	0.536	0.548	0.526	0.537	0.547	0.549	0.534	0.531
musk1	0.99	0.883	0.856	0.867	0.877	0.860	0.883	0.891	0.862
banknote	0.99	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
wine	0.99	0.982	0.982	0.982	0.982	0.979	0.979	0.982	0.982
tae	0.99	0.361	0.315	0.321	0.419	0.418	0.323	0.299	0.438
sensorless	1.0	0.959	0.966	0.953	0.953	0.963	0.959	0.948	0.965
segmentation	1.0	0.910	0.904	0.909	0.896	0.912	0.897	0.897	0.902
dota2	1.0	0.595	0.594	0.594	0.595	0.593	0.592	0.594	0.595
optdigits	1.0	0.953	0.949	0.955	0.946	0.955	0.947	0.946	0.945
letter-recognition	1.0	0.653	0.652	0.658	0.646	0.652	0.646	0.646	0.646
pendigits	1.0	0.945	0.948	0.940	0.933	0.940	0.950	0.933	0.937
iris	1.0	0.980	0.980	0.980	0.980	0.960	0.980	0.980	0.980
Average macro-F1		0.753	0.742	0.749	0.748	0.739	0.746	0.742	0.744
Average rank		2.8	3.6	3.7	4.0	4.1	4.2	4.9	5.1

In Section 1.4, we mentioned that EvoDAG uses an ensemble of 30 models to improve its performance. Figure 4.4 presents an analysis of the best combination of selection schemes, EvoDAG acc-rnd, by the number of models in the ensemble. It can be observed that results converge around the 30th model.

Table 4.6: Comparison of EvoDAG’s performance using different techniques for parent and negative selection, ordered by number of classes of the datasets. The values represent the macro-F1 value obtained over the test sets. Rows and columns are ordered by number of classes, and macro-F1 average rank, respectively. The best performance in each problem is indicated in boldface. Colors indicate the performance of the selection techniques combination, the bluer, the better, in the opposite, the more gray, the worse.

	Number of classes	acc-rnd	acc-fit	rnd-rnd	sim-fit	prs-fit	prs-rnd	sim-rnd	fit-fit
ad	2	0.940	0.934	0.932	0.933	0.932	0.933	0.934	0.934
adult	2	0.793	0.792	0.792	0.791	0.793	0.791	0.791	0.791
aps-failure	2	0.828	0.840	0.864	0.839	0.836	0.830	0.825	0.847
banknote	2	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
bank	2	0.762	0.760	0.765	0.763	0.762	0.762	0.762	0.756
biodeg	2	0.841	0.835	0.851	0.842	0.849	0.823	0.838	0.829
parkinsons	2	0.747	0.752	0.730	0.734	0.653	0.718	0.718	0.667
census-income	2	0.767	0.510	0.770	0.767	0.511	0.767	0.765	0.753
musk2	2	0.943	0.906	0.940	0.942	0.942	0.945	0.948	0.951
dota2	2	0.595	0.594	0.594	0.595	0.593	0.592	0.594	0.595
musk1	2	0.883	0.856	0.867	0.877	0.860	0.883	0.891	0.862
fertility	2	0.453	0.453	0.453	0.453	0.442	0.453	0.453	0.442
IndianLiverPatient	2	0.661	0.711	0.694	0.649	0.654	0.639	0.642	0.642
ml-prove	2	1.000	1.000	1.000	1.000	1.000	1.000	1.000	0.999
magic04	2	0.848	0.849	0.846	0.847	0.849	0.848	0.846	0.834
tae	3	0.361	0.315	0.321	0.419	0.418	0.323	0.299	0.438
iris	3	0.980	0.980	0.980	0.980	0.960	0.980	0.980	0.980
wine	3	0.982	0.982	0.982	0.982	0.979	0.979	0.982	0.982
cmc	3	0.536	0.548	0.526	0.537	0.547	0.549	0.534	0.531
car	4	0.866	0.914	0.860	0.829	0.868	0.848	0.830	0.843
page-blocks	5	0.825	0.791	0.761	0.772	0.793	0.804	0.773	0.773
drug-consumption	7	0.227	0.204	0.197	0.224	0.209	0.209	0.223	0.196
segmentation	7	0.910	0.904	0.909	0.896	0.912	0.897	0.897	0.902
agaricus-lepiota	7	0.684	0.684	0.682	0.684	0.685	0.682	0.682	0.677
optdigits	10	0.953	0.949	0.955	0.946	0.955	0.947	0.946	0.945
pendigits	10	0.945	0.948	0.940	0.933	0.940	0.950	0.933	0.937
yeast	10	0.468	0.454	0.455	0.457	0.430	0.452	0.444	0.456
sensorless	11	0.959	0.966	0.953	0.953	0.963	0.959	0.948	0.965
krkopt	18	0.188	0.183	0.198	0.143	0.185	0.160	0.142	0.147
letter-recognition	26	0.653	0.652	0.658	0.646	0.652	0.646	0.646	0.646
Average macro-F1		0.753	0.742	0.749	0.748	0.739	0.746	0.742	0.744
Average rank		2.8	3.6	3.7	4.0	4.1	4.2	4.9	5.1

Table 4.7: Comparison of EvoDAG’s performance using different techniques for parent and negative selection, ordered by number of variables of the datasets. The values represent the macro-F1 value obtained over the test sets. Rows and columns are ordered by number of variables, and macro-F1 average rank, respectively. The best performance in each problem is indicated in boldface. Colors indicate the performance of the selection techniques combination, the bluer, the better, in the opposite, the more gray, the worse.

	Number of variables	acc-rnd	acc-fit	rnd-rnd	sim-fit	prs-fit	prs-rnd	sim-rnd	fit-fit
iris	4	0.980	0.980	0.980	0.980	0.960	0.980	0.980	0.980
banknote	4	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
tae	5	0.361	0.315	0.321	0.419	0.418	0.323	0.299	0.438
krkopt	6	0.188	0.183	0.198	0.143	0.185	0.160	0.142	0.147
car	6	0.866	0.914	0.860	0.829	0.868	0.848	0.830	0.843
fertility	9	0.453	0.453	0.453	0.453	0.442	0.453	0.453	0.442
cmc	9	0.536	0.548	0.526	0.537	0.547	0.549	0.534	0.531
yeast	9	0.468	0.454	0.455	0.457	0.430	0.452	0.444	0.456
IndianLiverPatient	10	0.661	0.711	0.694	0.649	0.654	0.639	0.642	0.642
magic04	10	0.848	0.849	0.846	0.847	0.849	0.848	0.846	0.834
page-blocks	10	0.825	0.791	0.761	0.772	0.793	0.804	0.773	0.773
wine	13	0.982	0.982	0.982	0.982	0.979	0.979	0.982	0.982
adult	14	0.793	0.792	0.792	0.791	0.793	0.791	0.791	0.791
bank	16	0.762	0.760	0.765	0.763	0.762	0.762	0.762	0.756
letter-recognition	16	0.653	0.652	0.658	0.646	0.652	0.646	0.646	0.646
pendigits	16	0.945	0.948	0.940	0.933	0.940	0.950	0.933	0.937
segmentation	19	0.910	0.904	0.909	0.896	0.912	0.897	0.897	0.902
agaricus-lepiota	22	0.684	0.684	0.682	0.684	0.685	0.682	0.682	0.677
parkinsons	22	0.747	0.752	0.730	0.734	0.653	0.718	0.718	0.667
drug-consumption	30	0.227	0.204	0.197	0.224	0.209	0.209	0.223	0.196
census-income	41	0.767	0.510	0.770	0.767	0.511	0.767	0.765	0.753
biodeg	41	0.841	0.835	0.851	0.842	0.849	0.823	0.838	0.829
sensorless	48	0.959	0.966	0.953	0.953	0.963	0.959	0.948	0.965
ml-prove	56	1.000	1.000	1.000	1.000	1.000	1.000	1.000	0.999
optdigits	64	0.953	0.949	0.955	0.946	0.955	0.947	0.946	0.945
dota2	116	0.595	0.594	0.594	0.595	0.593	0.592	0.594	0.595
musk2	166	0.943	0.906	0.940	0.942	0.942	0.945	0.948	0.951
musk1	166	0.883	0.856	0.867	0.877	0.860	0.883	0.891	0.862
aps-failure	170	0.828	0.840	0.864	0.839	0.836	0.830	0.825	0.847
ad	1557	0.940	0.934	0.932	0.933	0.932	0.933	0.934	0.934
Average macro-F1		0.753	0.742	0.749	0.748	0.739	0.746	0.742	0.744
Average rank		2.8	3.6	3.7	4.0	4.1	4.2	4.9	5.1

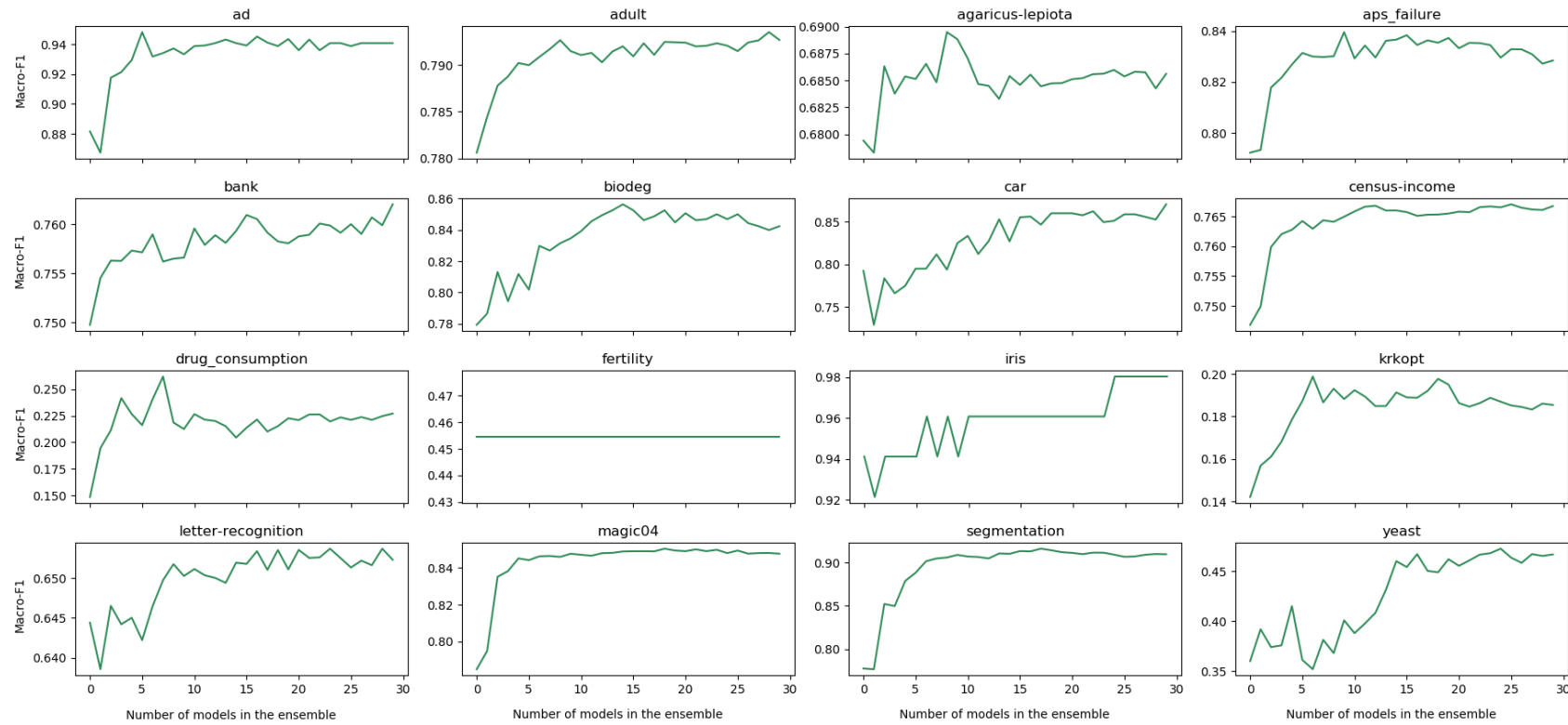


Figure 4.4: Analysis of EvoDAG acc-rnd according to the number of models in the ensemble. Lines represent the ensemble macro-F1 result using the number of models represented in axis-x. Source: Own elaboration.

Figure 4.5 shows the results of EvoDAG with several selection schemes combinations changing the number of arguments of function Σ . At the ending of the selection scheme's name appears the number of arguments that function Σ receives, by default it is 60. For this analysis only the following combinations were used: acc-rnd, rnd-rnd and fit-fit. In general, the use of the combination of parent selection using our heuristic based on accuracy and random negative selection presents the best performance. Followed by random selection for parent and negative selection. When only 10 arguments are used for the function Σ is a special case. fit-fit-10, rnd-rnd-10, and acc-fit-10 appear in the last positions, also, only in this case, fit-fit-10 is better than rnd-rnd-10 and acc-rnd-10, but statistical differences were not found. Our interpretation is: the selection heuristic based on accuracy needs a high number of arguments in the function Σ for working better. Based on our experiments, 60 is optimum. Additionally, there is a tendency that acc-rnd is better than fit-fit despite the number of arguments in function Σ are different.

Discussion: First, we demonstrate that our selection heuristics based on accuracy (acc), cosine similarity (sim), and Pearson's correlation coefficient (prs) performed better when they are applied to the functions Σ , Naive Bayes (NB and MN), and Nearest Centroid (NC) than when they are used for all the functions with more than one argument (Σ , Π , max, min, hypot, NB, MN, and NC). It is because the proposed heuristics were designed based on the properties of the functions Σ , Naive Bayes, and Nearest Centroid. Based on the results, the best parent selection heuristic is the one based on accuracy. We assume that it is because we are solving classifications problems, and accuracy is the most straightforward technique for comparing individuals' semantics in multi-class problems. Besides, the combination of selection schemes rnd-rnd, random selection of parents and negative random selection is better than fit-fit, parent and negative selection based on fitness. We assume that the reason is because rnd-rnd promotes population diversity, also, when EvoDAG optimizes the parameters of functions with ordinary least squares (see Section 1.4), it improves the semantics of individuals using the target semantics.

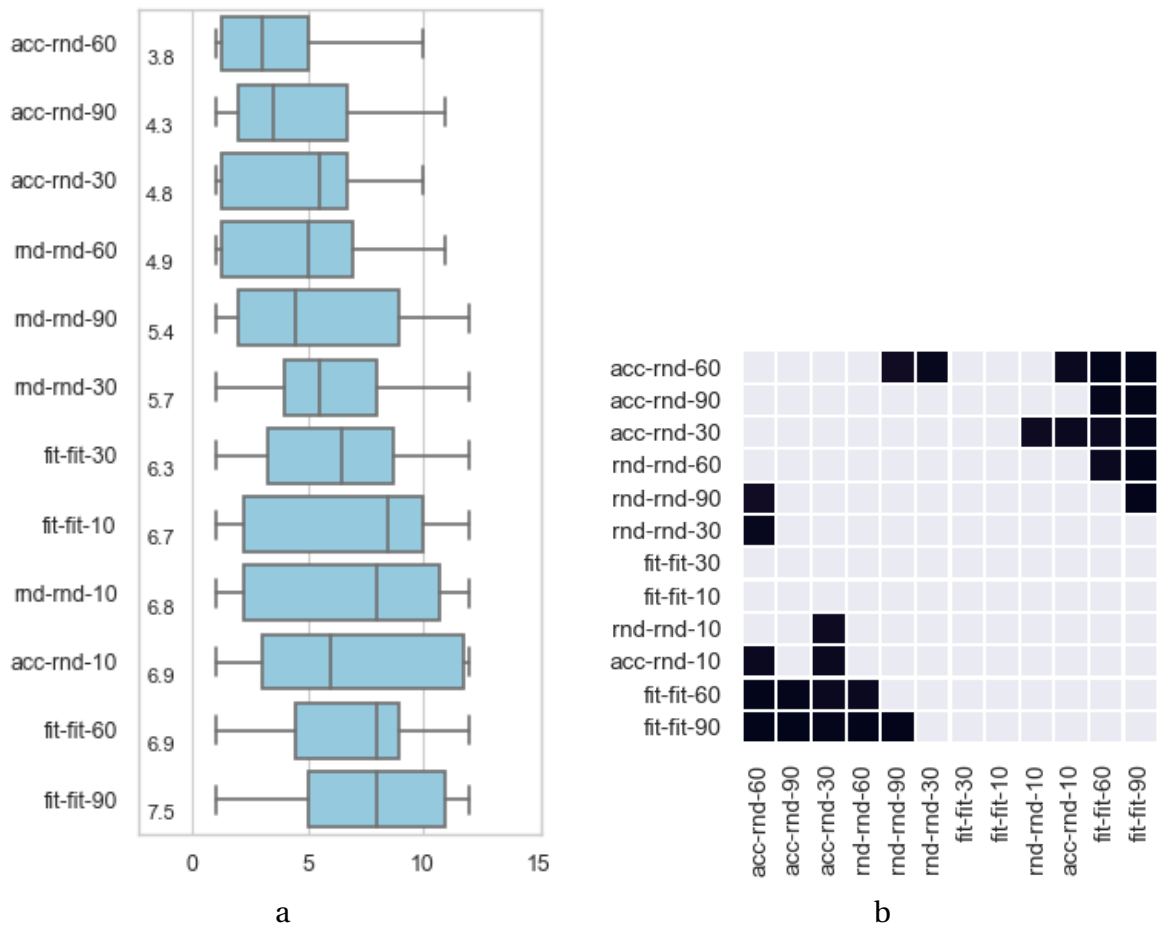


Figure 4.5: Analysis of selection schemes combinations changing the number of arguments in the function Σ . a) Boxplots present the macro-F1 ranks measured over the test datasets. The average rank sorts classifiers, and it appears on the left. b) Statistical comparison (Wilcoxon signed-rank test) of the different selection schemes combinations based on macro-F1. Black cells represent that the pair of schemes are statistically different with a 95% confidence. Source: Own elaboration.

4.5 Comparison of the Proposed Selection Heuristics and State-of-the-Art Selection Schemes

As we mentioned in Chapter 2, some new selection techniques for Genetic Programming have been proposed. For us, because of the relationship with our selection heuristics, the most important ones for parent selection are Angle-Driven-Selection [15] and Novelty Search [62]. For that reason, we decide to implement them in EvoDAG and compare them against our proposal. In this section, we describe in detail those techniques and how they were implemented in EvoDAG. Moreover, the results are shown.

Angle-Driven Selection

Angle-Driven Selection

The objective of Angle-Driven Selection is to maximize the angle of parents in the error space.

$$\gamma_r = \arccos\left(\frac{(\vec{t} - \vec{P}_1) \cdot (\vec{t} - \vec{P}_2)}{\|\vec{t} - \vec{P}_1\| \|\vec{t} - \vec{P}_2\|}\right)$$

Chen *et al.* said in [15], “The angle-awareness is expected to make geometric operators more effective and help the evolutionary process to converge to the target semantics much more accurately and faster”. They started with the idea that individuals’ semantics can be represented as vectors in the semantics space, then, the angle between two individuals \vec{p}_1 and \vec{p}_2 can be calculated as $\gamma = \arccos\left(\frac{\vec{p}_1 \cdot \vec{p}_2}{\|\vec{p}_1\| \|\vec{p}_2\|}\right)$, where $\|\vec{p}\| = \sum_j \sqrt{p_j^2}$. In addition, they calculate the angle between individuals’ semantics in the error space (described in Section 2.3) to include the target semantics t , this is, the angle between relative individuals. The angle between the relative semantics of two individuals \vec{p}_1 and \vec{p}_2 is defined as:

$$\gamma_r = \arccos\left(\frac{(\vec{t} - \vec{p}_1) \cdot (\vec{t} - \vec{p}_2)}{\|\vec{t} - \vec{p}_1\| \|\vec{t} - \vec{p}_2\|}\right) \quad (4.1)$$

Angle-Driven Selection (ADS) was proposed to select parents for geometric crossover. ADS selects the first parent \vec{p}_1 according to the tournament selection based on fitness. An iterative procedure is performed to select the second parent \vec{p}_2 according to its angle distance from \vec{p}_1 . The goal is to find parents with an angle distance near to 180 degrees.

Originally, ADS was designed for geometric crossover, and in the case of EvoDAG, several functions that need more than one argument are applied, as: Σ , Π , max, min, hypot, NB, MN, and NC. For that reason, we needed to modify the implementation but keeping the main idea of ADS. First, a parent is selected using a tournament based on fitness, as the original proposal of ADS, then, the next arguments, or parents, are selected using a tournament based on the relative angle between a candidate and the first parent. A simple example is showed in Figure 4.6. In this case, the function Σ is selected for creating the offspring and it needs three arguments. The first parent is selected from the population \mathcal{P} using a tournament based on fitness. Then, the next arguments are independently selected. For each one, a tournament is performed, which is described as follows. First, two individuals are randomly selected from the population \mathcal{P} , and the relative angle (Equation 4.1) between their semantics and the first argument semantics is calculated. The individual with the maximum angle between its semantics and the first parent semantics is selected as argument. Algorithm 10 describes the selection of arguments based on ADS in EvoDAG. It is important to mention that, as we affirmed in Section 1.4, the individuals' semantics in EvoDAG when solving classification problems is an array of semantics vectors, then, the relative angle between individuals' semantics is calculated as the sum of angles among pairs of vectors. This process is described in the algorithm.

Algorithm 10: Selection of arguments based on Angle-Driven Selection (ads) in EvoDAG

Input: k , a population of individuals \mathcal{P}

Input: ts , the tournament size

Input: t , the array of target semantics

Output: The list of individuals that will be passed as arguments

$args \leftarrow$ an empty list of individuals;

Select the first parent F using tournament selection based on fitness;

for i **from** 2 **to** k **do**

$A \leftarrow$ an individual randomly selected from the population \mathcal{P} ;

$semantics_A \leftarrow$ the array of individual A 's semantics vectors;

$angle_A \leftarrow 0$;

for each element in $semantics_F$ **and** $semantics_A$ **do**

$$\quad \quad \quad \left| \quad \quad \quad angle_A \leftarrow angle_A + \arccos \left(\frac{(\vec{t}_i - \vec{semantics}_{Fi}) \cdot (\vec{t}_i - \vec{semantics}_{Ai})}{\|\vec{t}_i - \vec{semantics}_{Fi}\| \|\vec{t}_i - \vec{semantics}_{Ai}\|} \right); \right.$$

end

for j **from** 2 **to** ts **do**

$B \leftarrow$ an individual randomly selected from the population \mathcal{P} ;

$semantics_B \leftarrow$ the array of individual B 's semantics vectors;

$angle_B \leftarrow 0$;

for each element in $semantics_F$ **and** $semantics_B$ **do**

$$\quad \quad \quad \left| \quad \quad \quad angle_B \leftarrow angle_B + \arccos \left(\frac{(\vec{t}_i - \vec{semantics}_{Fi}) \cdot (\vec{t}_i - \vec{semantics}_{Bi})}{\|\vec{t}_i - \vec{semantics}_{Fi}\| \|\vec{t}_i - \vec{semantics}_{Bi}\|} \right); \right.$$

end

if $angle_B > angle_A$ **then**

$A \leftarrow B$;

$angle_A \leftarrow angle_B$;

end

end

 Add the individual A to $args$;

end

Return $args$;

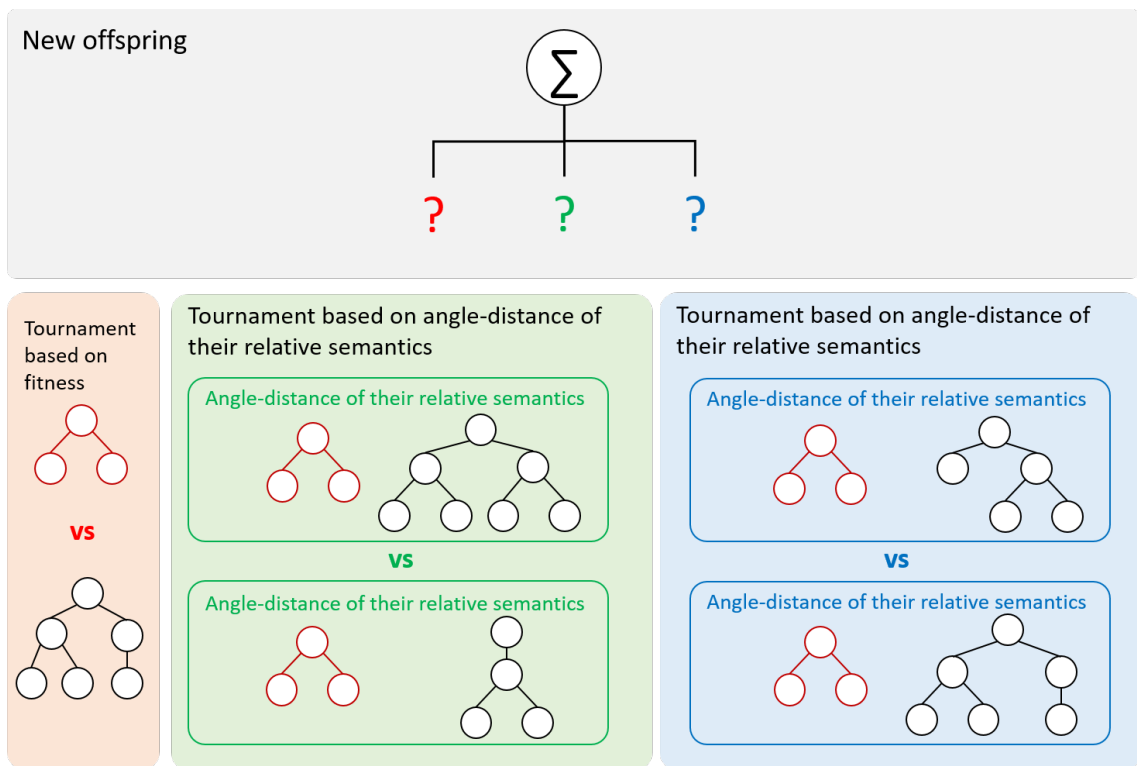


Figure 4.6: Example of parent selection based on ADS (in EvoDAG). Source: Own elaboration.

Novelty Search

Novelty Search

The aim of Novelty Search, in the case of parent selecting, is to choose as parents individuals that are novel in the population. Each argument is independently selected using tournament selection based on novelty.

$$\log(\phi(x)) = \sum_j \log\left(\frac{1}{P_j(x_j) + \epsilon}\right)$$

Novelty Search [62] proposes to replace the fitness function by individuals' novelty with the idea of promoting population diversity. The novelty of the individual is computed as the average of the distances between it and its k -nearest neighbors in the semantic space. Rather than viewing open-ended evolution as an adaptive competition, it can be viewed simply as a passive drift through the lattice of novelty. We use the equations proposed by Naredo *et al.* in [72] for evolving classifiers in Genetic Pro-

gramming using Novelty Search. They used as individual's novelty $\phi(x)$ the inverse of the probability of observing that individual in the population, this is, $\phi(x) = \frac{1}{P(x)}$. As we mentioned in Section 1.4, an individual can be represented as its labels vectors, this is, a vector whose entries are the labels that the individual's classifier assigns to each training sample. Based on those labels vectors, the novelty of individuals is calculated. For explaining the calculus, we use the example showed in Figure 4.7. Columns and rows represent individuals and training samples, respectively. In this example there are only 3 classes: A, B, and C. For each training sample, the probability of each classes is calculated, we can do it by counting. In case of the training sample 1, the probability of class A, $P(A)$, is equal to 0.4 because there are only 2 A's in 5 individuals. Using those probabilities, we can calculate the novelty of individuals ϕ . For calculating the novelty of an individual x , we use the equation $\phi(x) = \frac{1}{P(x)}$. In addition, we can calculate $P(x)$ as the multiplication of the probabilities of individual's values over all the training samples. This is, $\phi(x) = \frac{1}{\prod_j P_j(x_j)}$, where j goes for each training sample. In Figure 4.7 this calculation is showed. The multiplication of probabilities can be result in numeric problems, for solving that, it is changed for the sum of logarithms, this is, $\log(\phi(x)) = \sum_j \log\left(\frac{1}{P_j(x_j)}\right)$. For avoiding numerical errors caused by divisions by zero, an small value ϵ is added to the probabilities, as it can be seen as follows:

$$\log(\phi(x)) = \sum_j \log\left(\frac{1}{P_j(x_j) + \epsilon}\right) \quad (4.2)$$

Specifically, in EvoDAG, each one of the k arguments (or parents) are independently selected from the population \mathcal{P} using a tournament selection based on novelty search. Figure 4.8 shows a simple example of parent selection in EvoDAG. In this case, an offspring is created with the function Σ . Three parents need to be chosen from the population \mathcal{P} for becoming arguments. For each one of the arguments, two individuals and randomly selected from the population and the most novel one is chosen as parent. In this case, the number of individuals in the tournament ts is two. Algorithm 11 describes the selection of arguments in EvoDAG based on novelty search.

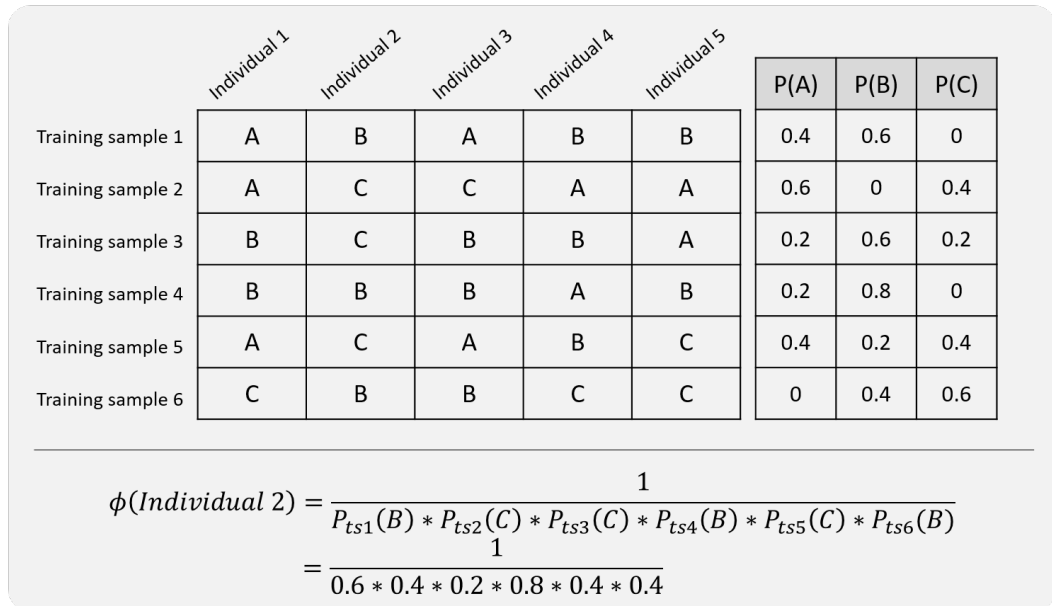


Figure 4.7: Example of individual's novelty calculation. Source: Own elaboration.

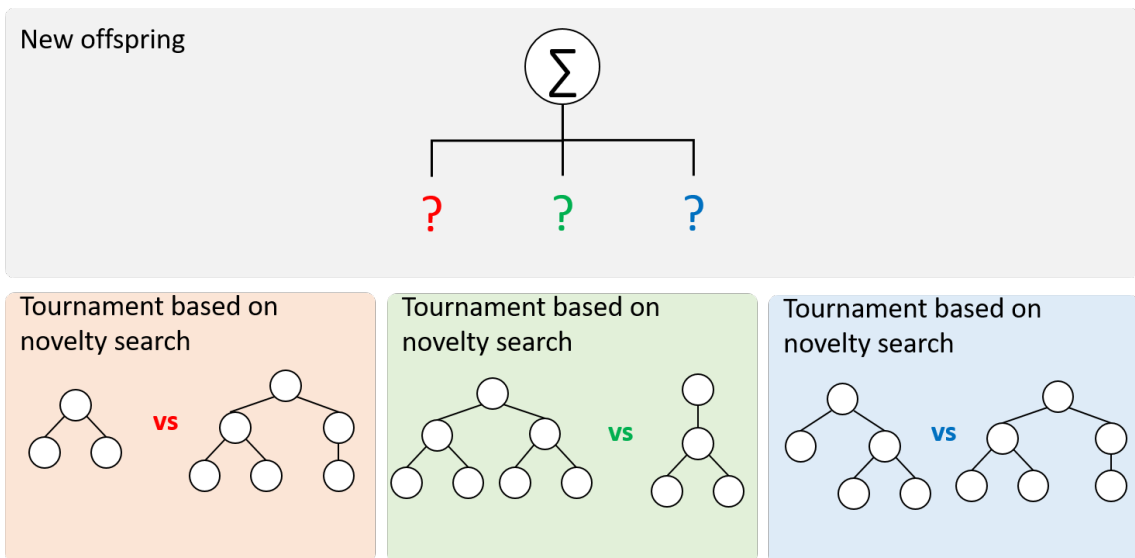


Figure 4.8: Parent selection based on novelty search (in EvoDAG). Source: Own elaboration.

Algorithm 11: Selection of arguments based on Novelty Search (nvs) in EvoDAG

Input: k , the number of arguments**Input:** ts , the tournament size**Output:** The list of individuals that will be passed as arguments $args \leftarrow$ an empty list of individuals;**for** i **from** 1 **to** k **do** $A \leftarrow$ an individual randomly selected from the population \mathcal{P} ; $novelty_A \leftarrow \sum_j \log\left(\frac{1}{P_j(A_j)+\epsilon}\right)$; **for** j **from** 2 **to** ts **do** $B \leftarrow$ an individual randomly selected from the population \mathcal{P} ; $novelty_B \leftarrow \sum_j \log\left(\frac{1}{P_j(B_j)+\epsilon}\right)$; **if** $novelty_B > novelty_A$ **then** $A \leftarrow B$; $novelty_A \leftarrow novelty_B$; **end** **end** Add the individual A to $args$;**end****Return** $args$;

Results of the Comparison of our Proposed Selection Heuristics against ADS and NVS

As it can be observed in the explanation of ads, the authors proposed first select the individuals of the tournament based on fitness, and then apply other tournament based on their relative angle in the error space. For that reason, we decided to add another parameter to our heuristics and to ads, this is, the way that individuals of the tournament are selected, it can be randomly (rnd) or based on fitness (fit). The combinations of selection techniques' notation is as follows, the symbol "-" follows the parent selection technique, then comes the abbreviation of the negative selection scheme, and, for our heuristics and ads, at the ending, after the symbols "-", it comes the abbreviation of the scheme for selecting the individuals that participate in the tournaments. For example, acc-rnd-fit means that tournament selection based on accuracy is used for parent selection, the negative selection is performed randomly, and the individuals in the tournament based on accuracy are previously selected using a tournament based

on fitness.

Figure 4.9 presents the results of the comparison, based on macro-F1, of our proposed heuristics (acc, prs, and sim) against state-of-the-art selection techniques: Angle-Driven Selection (ads) and Novelty Search (nvs). In Figure 4.1.a, it can be observed that the performance of our heuristics is generally better than ads and nvs. As the behavior of our heuristics, we can see that ads works better when it is applied only to the functions Σ , Naive Bayes, and Nearest Centroid than when it is applied to all functions with more than one argument. Besides, ads is better when it uses the combination of selection schemes ads-rnd-rnd than the original proposal ads-fit-fit. The Wilcoxon statistical test (see 4.9.b) presents that acc-rnd-rnd is statistically better than ads and nvs.

Figure 4.10 presents a comparison of different selection schemes, including our proposed heuristics and state-of-the-art selection schemes based on two criteria: macro-F1 average rank and the time that EvoDAG spends on the training phase. The closest the technique to the origin, the better it is in terms of performance and time. It can be observed that Angle-Driven Selection (ads) and Novelty Search (nvs) are further to the origin. Based on the experiments, it indicates that our proposed heuristics performed better, for classification tasks than ads and nvs based on macro-F1 and time.

Discussion: We consider that, in these experiments, our heuristics are better than Angle-Driven Selection (ads) because all they were implemented in EvoDAG, and ads was designed for working with geometric crossover, and our heuristics were inspired in the properties of the function Σ and the classifiers Naive Bayes and Nearest Centroid that are part of EvoDAG. In the case of Novelty Search (nvs), we assume that the problem with nvs was that it searches for individuals that are different from the rest of individuals in the population, and when the functions' arguments are selected, individuals are different from the majority of individuals in the population but maybe among them are similar.

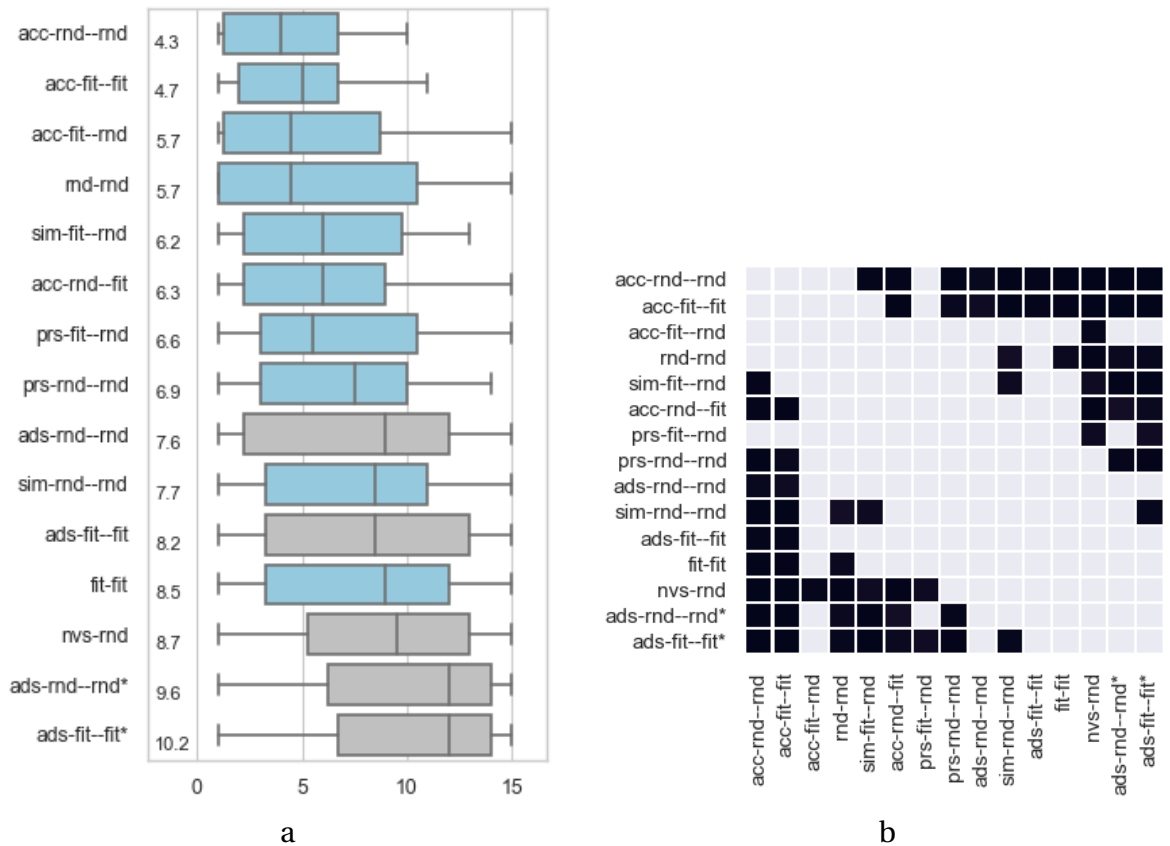


Figure 4.9: Proposed heuristics against state-of-the-art selection schemes based on macro-F1. a) Boxplots presents the ranks, those are measured using macro-F1 over the test dataset. Gray boxplots represent the selection techniques from the state-of-the-art, Novelty Search (nvs) and Angle-Driven Selection (ads). The average rank, or time per sample, sorts classifiers, and it appears on the left. b) Statistical comparison (Wilcoxon signed-rank test) of the different selection schemes combinations against state-of-the-art selection techniques based on macro-F1. Black cells represent that the pair of schemes are statistically different with a 95% confidence. Source: Own elaboration.

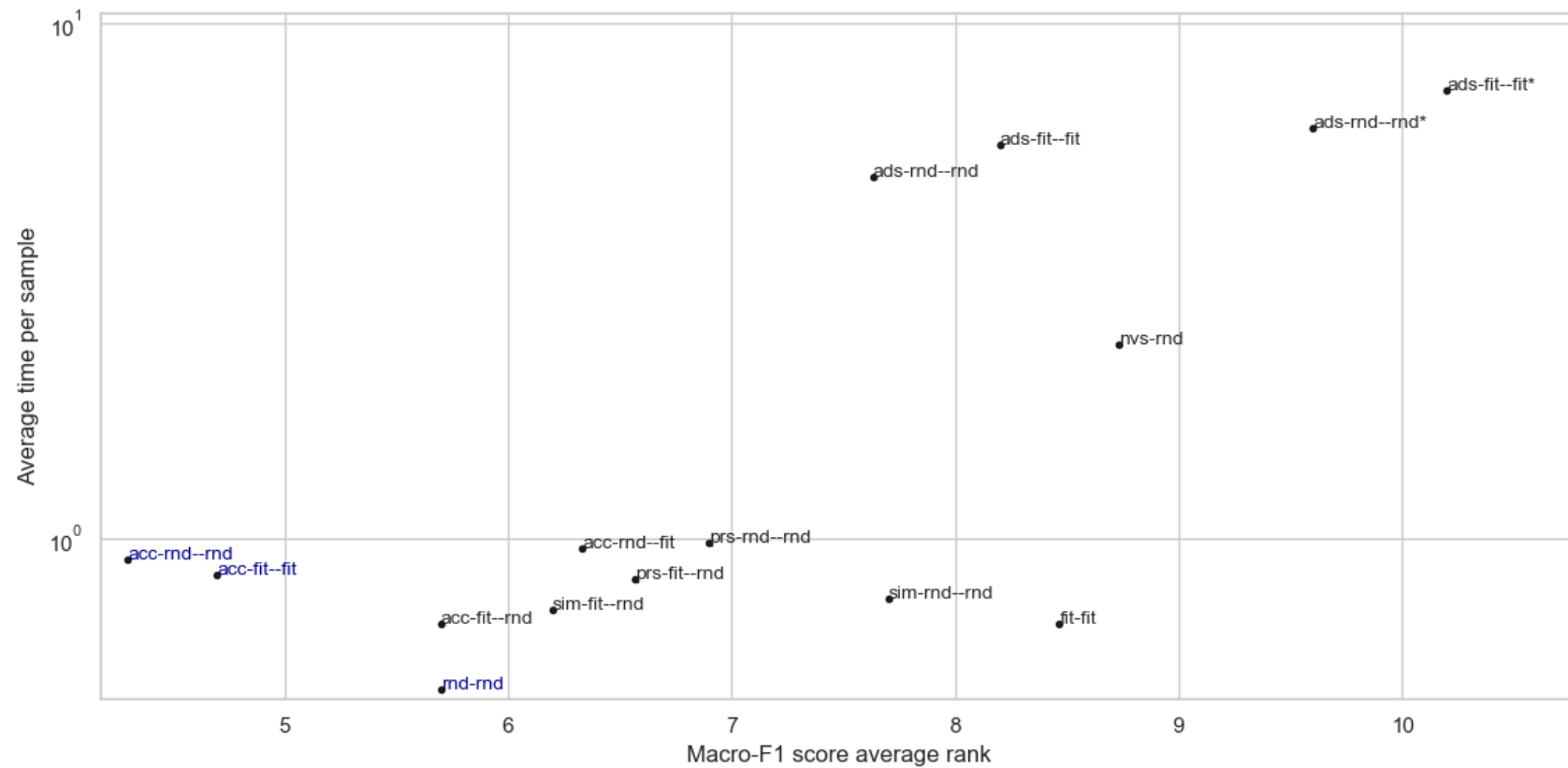


Figure 4.10: Comparison of the different selection schemes combinations and state-of-the-art selection schemes based on macro-F1 average rank and the time, in seconds, required by EvoDAG's training phase. The time is presented in seconds, and it is the average time per sample. Source: Own elaboration.

4.6 Comparison of EvoDAG and State-of-the-Art Classifiers

After analyzing the performance of the different selection schemes, it is the moment to compare EvoDAG against state-of-the-art classifiers. We chose EvoDAG only with the combination of the selection schemes: acc-rnd, rnd-rnd, fit-fit, ads-rnd, nvs-rnd. acc-rnd, because it is the heuristic that gives better results, rnd-rnd they are the simplest schemes and they proportionate good results, fit-fit because it uses the traditional selection schemes based on fitness, and finally, ads-rnd and nvs-rnd because they are the selection schemes of the state of the art. We decided to compare EvoDAG against sixteen classifiers of the scikit-learn python library [80], all of them using their default parameters. Specifically, these classifiers are Perceptron, MLPClassifier, BernoulliNB, GaussianNB, KNeighborsClassifier, NearestCentroid, LogisticRegression, LinearSVC, SVC, SGDClassifier, PassiveAggressiveClassifier, DecisionTreeClassifier, ExtraTreesClassifier, RandomForestClassifier, AdaBoostClassifier and GradientBoostingClassifier. It is also included in the comparison two auto-machine learning libraries: autosklearn [23] and TPOT [76]. It is important to mention that TPOT (see Section 2.4) is a Genetic Programming tool for automatically constructing and optimizing machine learning pipelines using 14 preprocessors, 5 feature selectors, and 11 classifiers; all these techniques implemented in scikit-learn.

Figure 4.11 shows the comparison of classifiers based on macro-F1 ranks. The best classifier, based on the results of these experiments, is TPOT, followed by EvoDAG acc-rnd, autosklearn, and EvoDAG rnd-rnd. It can be seen that the use of our proposed selection heuristic based on accuracy and negative random selection improves the performance of EvoDAG and positioned it into second place. EvoDAG performs better than the scikit-learn classifiers, and it is competitive with auto-machine learning libraries.

For validating the results, we use the statistical Wilcoxon signed-rank test [102],

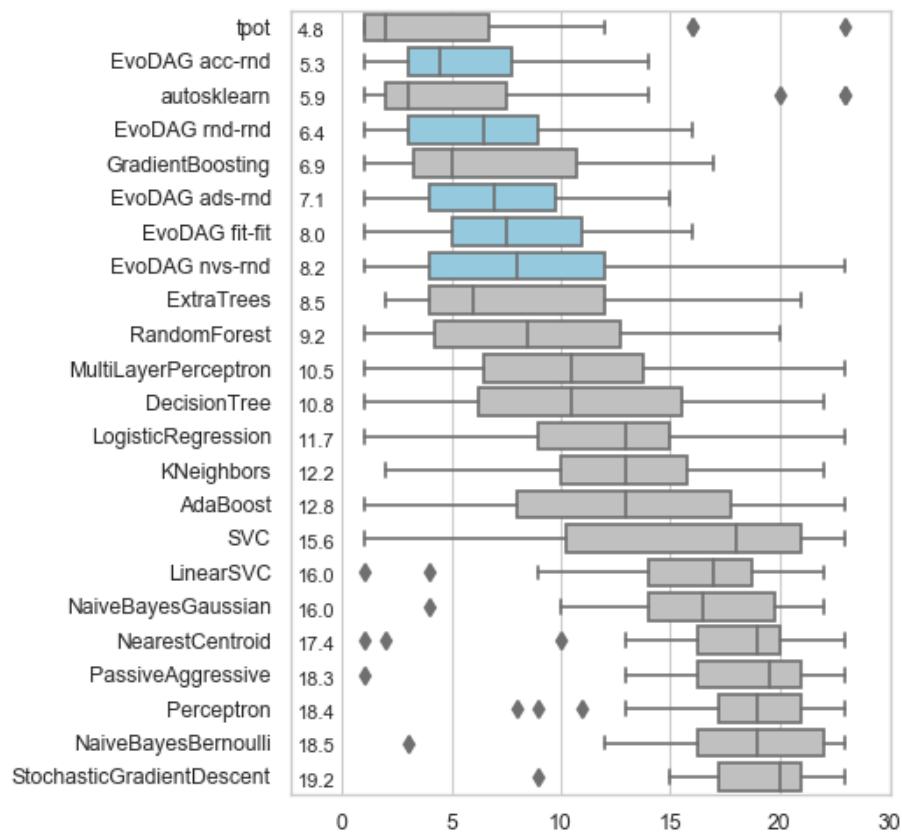


Figure 4.11: Comparison of EvoDAG against state-of-the-art classifiers based on macro-F1 rank. The average rank sorts classifiers, and the ranks values are on the left. The blue boxplots represent EvoDAG systems. Source: Own elaboration.

and the p -values were adjusted with the Holm-Bonferroni method [44] to consider multiple comparisons. Macro-F1 values were used for the statistical test. TPOT was found statistically better than Logistic Regression, KNeighborsClassifier, NearestCentroid, AdaBoostClassifier, SVC, Linear SVC, GaussianNB, BernoulliNB, PassiveAggressiveClassifier, Perceptron, SGDClassifier. Nevertheless, there were not found statistical differences between TPOT and EvoDAG. Figure 4.12 shows the results of Wilcoxon signed-rank test [102] by pairs of classifiers. It can be seen that the results of TPOT are statistically different from the ones obtained with EvoDAG fit-fit, the classical selection schemes, EvoDAG ads-rnd, and EvoDAG nvs-rnd, the ones of the state-of-the-art. However, no statistical differences were found between TPOT and EvoDAG acc-rnd or EvoDAG rnd-rnd. It confirms that our proposed selection schemes statistically improve the performance of EvoDAG.

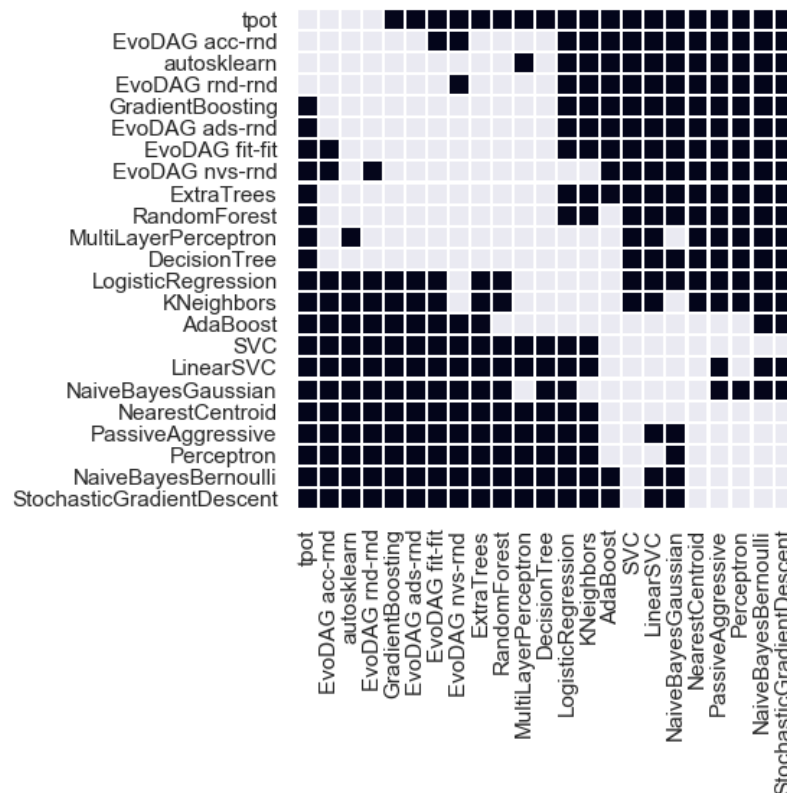


Figure 4.12: Statistical comparison (Wilcoxon test) of the different classifiers based on macro-F1. Black cells represent that the pair of schemes are statistically different with a 95% confidence. Source: Own elaboration.

The classifiers' comparison based on the time that they spend learning the model

is presented in Figure 4.13. It can be seen that scikit-learn classifiers are the faster; most of them spend from 0.007 to 0.009 seconds per sample. EvoDAG, with the different selection schemes, spends more time than scikit-learn classifiers in the learning phase. It spends, on average, from 0.5 to 5 seconds per sample. However, EvoDAG is considerably faster than the auto-machine learning libraries, autosklearn and TPOT, that consume on average 11.5 and 57.68 seconds, respectively, in the learning phase.

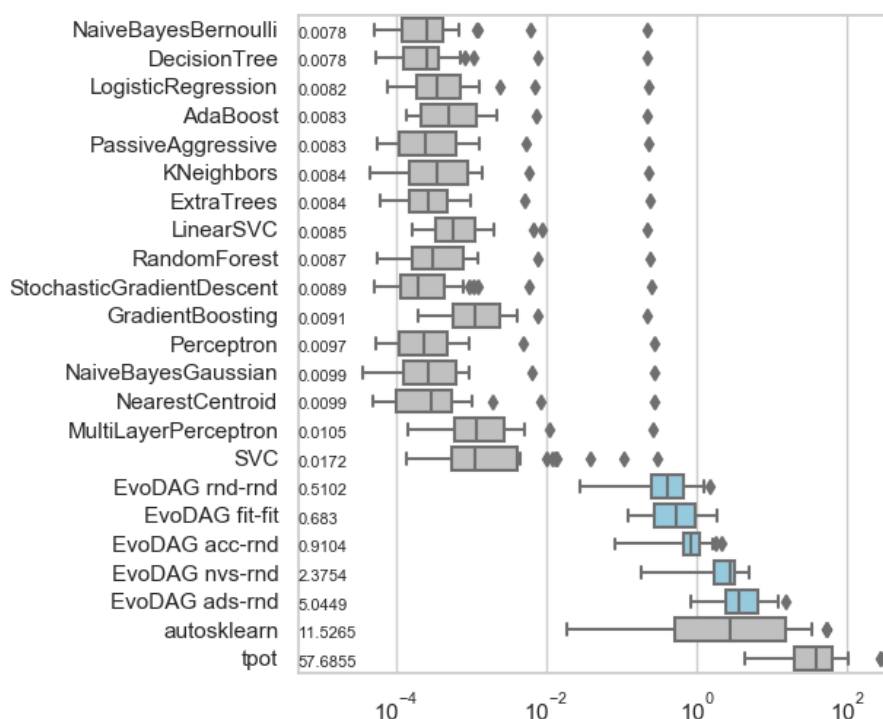


Figure 4.13: Comparison of EvoDAG against state-of-the-art classifiers based on the time required by the classifiers' training phase. The time is presented in seconds, and it is the average time per sample. The average time sorts classifiers, and those values are on the left. The blue boxplots represent EvoDAG systems. Source: Own elaboration.

Once more time, an analysis in two dimensions for comparing the different classifiers based on performance (macro-F1 average rank) and time (seconds per sample) was performed (see Figure 4.14). Remembering, the closest is the classifier to the origin, the better it is in terms of performance and time. We can observe that the classifiers in the Pareto frontier are TPOT, EvoDAG acc-rnd, EvoDAG rnd-rnd, GB, ET, and DT. The interpretation of this is as follows. If you want a good performance and you do not care about the time-consuming in the training phase, choose TPOT. If you want to

have results very fast with good performance, but not the best performance, use Gradient Boosting Classifier. On the other hand, if you wish to a considerable performance at a reasonable time, use EvoDAG acc-rnd.

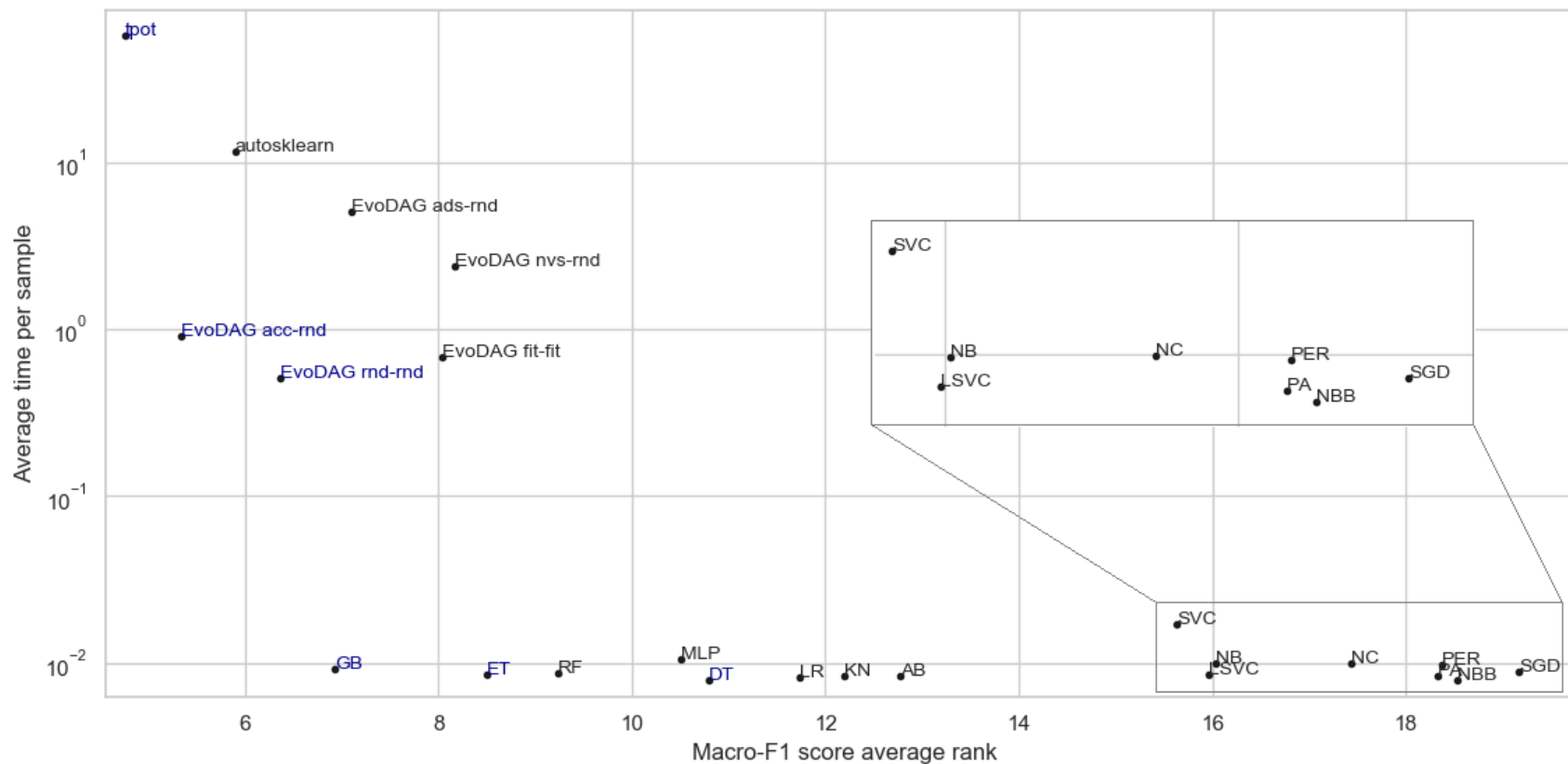


Figure 4.14: Comparison of EvoDAG with state-of-the-art classifiers based on macro-F1 and the time required by the classifiers' training phase. The time is presented in seconds, and it is the average time per sample. The classifiers are: EvoDAG acc-rnd, EvoDAG rnd-rnd, EvoDAG fit-fit, tpot, autosklearn, Perceptron (PER), MLPClassifier (MLP), BernoulliNB (NBB), GaussianNB (NB), KNeighborsClassifier (KN), NearestCentroid (NC), LogisticRegression (LR), LinearSVC (LSVC), SVC, SGDClassifier (SDG), PassiveAggressiveClassifier (PA), DecisionTreeClassifier (DT), ExtraTreesClassifier (ET), RandomForestClassifier (RF), AdaBoostClassifier (AB) and GradientBoostingClassifier (GB). Source: Own elaboration.

4.7 Visualization Map for Classifiers and Datasets

In this section, we use our visualization methodology called Liking Product Landscape (LPL) [87]. It was proposed as a visualization tool within the Sensory Analysis area. Its original goal is to understand the grades that consumers assign to products, with the idea of making comparisons among different products or for improving one based on the perception of consumers. However, in this section, we use this same methodology for analyzing the results of different techniques to solve classification problems. It can be helpful to visualize if some techniques produce similar results and how they perform in different datasets.

Figures 4.15 and 4.16 show two visualization maps for comparing the performance of different classifiers, in terms of macro-F1 ranks, based on the datasets. Figure 4.15 presents the results of the different combinations of selection schemes and Figure 4.16 the results of the comparison of EvoDAG against several state-of-the-art classifiers. The circles in the maps correspond to the classifiers, and in all maps, the position is the same. In the center of the figures, a map with the names of classifiers, or selection schemes combination, can be observed. Colors represent the performance; in this case, in terms of macro-F1 average rank, the bluer, the highest the classifier performance, in the opposite, the redder, the worse performance. Color outside points represents the tendency of the area. In the visualization map, techniques that have similar results are plotting together, whereas techniques that have different results are plotted apart. Lines in the center map represent the classifiers' distribution. They are useful for identifying groups. For example, in Figure 4.16, it can be detected 3 main groups of classifiers. One image for each dataset is showed intending to analyze the performance of classifiers by datasets. In this sense, images with the same pattern represent that the datasets are similar in terms of the results of classifiers.

The performance of the different combinations of selection schemes for parent and negative selection based on macro-F1 rank is shown on Figure 4.15. Based on the colors of the center map, the best combination of selection schemes is acc-rnd-rnd, followed by acc-fit-fit. Based on the position of selection techniques' combinations, the performance of acc-rnd-rnd is similar to acc-fit-fit, sim-fit-rnd, and prs-fit-rnd. On the bottom left, we can see all the versions of ads, in particular, the worst combinations are ads-rnd-rnd* and ads-fit-fit. Also, we can see that heuristics (acc, sim, prs) results are similar, while, fit-fit and rnd-rnd appear apart. The dataset that was the easiest to solve is banknote, because all versions of EvoDAG got good results (all are in blue). It is followed by ml-prove, fertility, iris, and wine. For the pattern and colors of images, we can see that ads and nvs got good results on the datasets: ad, banknote, ml-prove, and musk1. All of those datasets are binary classification problems. The combination of the classical tournaments based on fitness, fit-fit, is marked with a star, we can see that it got good results on the datasets aps-failure, banknote, dota2, iris, musk2, sensorless, tae, and wine, all of those datasets have a small number of classes, except sensorless that has 11 classes.

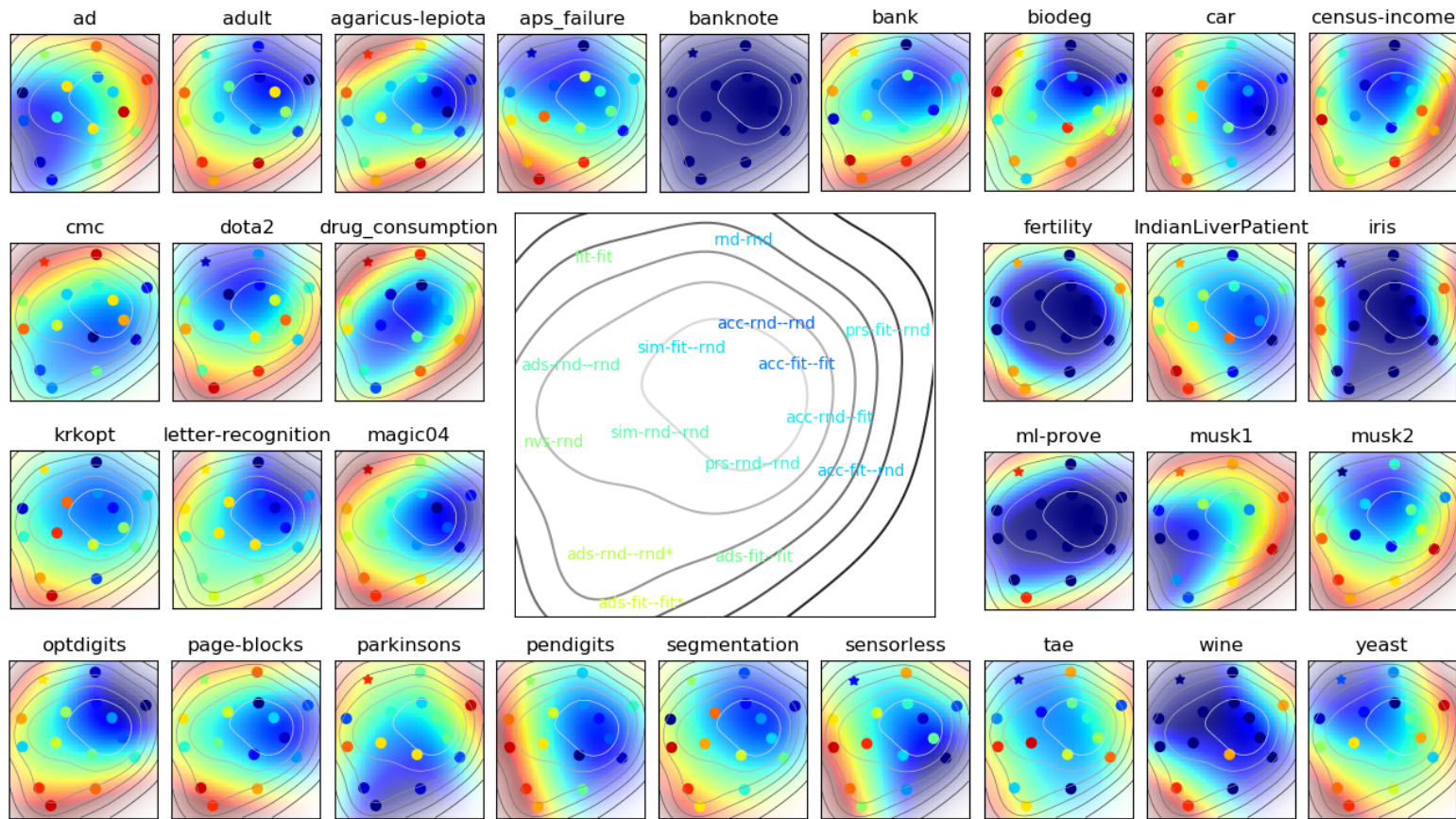


Figure 4.15: Comparison of different selection schemes for parent and negative selection based on macro-F1 rank using LPL as a visualization map. Source: Own elaboration.

Figure 4.16 shows the comparison of EvoDAG against state-of-the-art classifiers based on macro-F1 rank using the visualization map. Three main groups can be detected: (1) the different versions of EvoDAG; (2) the classifiers based on decision trees: RandomForestClassifier, ExtraTreesClassifier, DecisionTreeClassifier, and GradientBoostingClassifier; and (3) the scikit-learn classifiers. For the colors in the images, it can be seen that the classifiers in the bottom are the ones with the best performance, groups 1, 2, and the auto-machine learning tools: TPOT and autosklearn. Specifically, group 2 got better results in the datasets ad, krkopt, letter-recognition, and fertility. TPOT, the tool with the highest macro-F1 average rank, generally obtained excellent results in most of the datasets. However, for his color red in the dataset's images, we can detect that it could not solve the following datasets satisfactorily: fertility and IndianLiverPatient.

Most datasets (ad, adult, car, optdigits, magic04, musk2, page-blocks, parkinsons, pendigits, segmentation, sensorless, tae, wine, and, yeast) have the same color pattern. It means that the classifiers obtained approximately the same results on all those datasets. However, some datasets differ from the rest. For example, banknote, fertility, and iris, obtained good results using the classifiers SVC and Nearest Centroid.

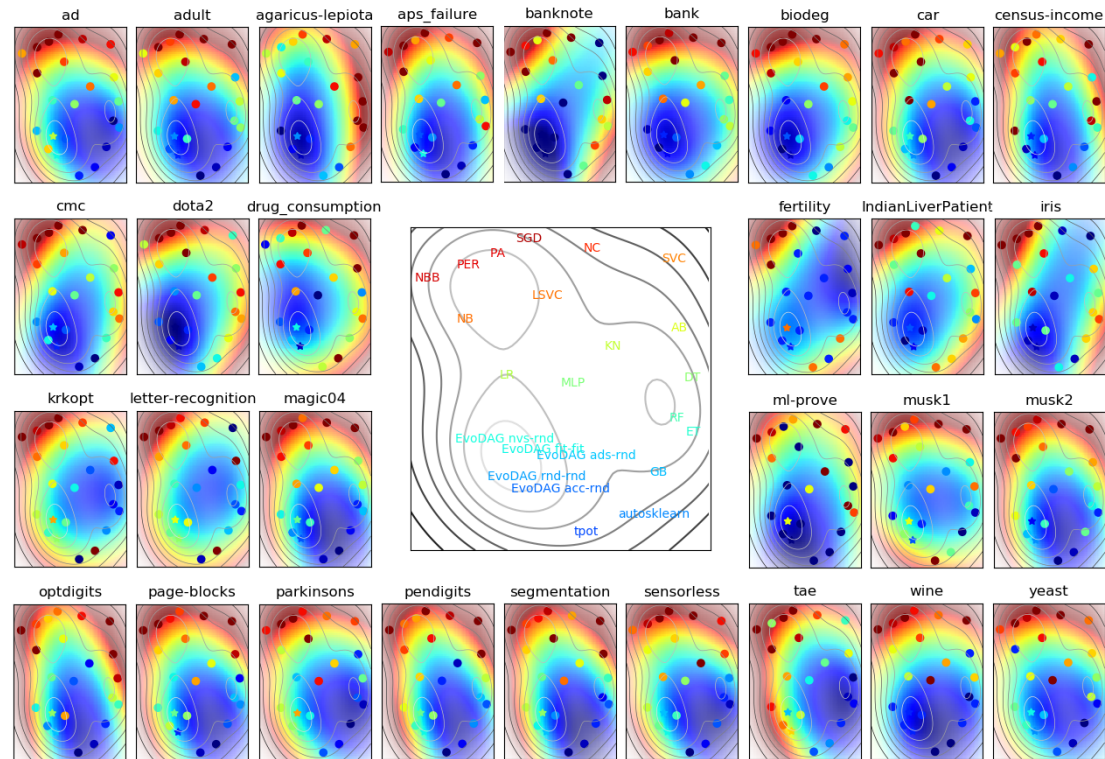


Figure 4.16: Comparison of EvoDAG vs state-of-the-art classifiers based on macro-F1 rank using LPL as a visualization map. The classifiers are: EvoDAG acc-rnd, EvoDAG rnd-rnd, EvoDAG fit-fit, tpot, autosklearn, Perceptron (PER), MLPClassifier (MLP), BernoulliNB (NBB), GaussianNB (NB), KNeighborsClassifier (KN), NearestCentroid (NC), LogisticRegression (LR), LinearSVC (LSVC), SVC, SGDClassifier (SDG), PassiveAggressiveClassifier (PA), DecisionTreeClassifier (DT), ExtraTreesClassifier (ET), RandomForestClassifier (RF), AdaboostClassifier (AB) and GradientBoostingClassifier (GB). Source: Own elaboration.

4.8 Summary

In this chapter, we presented the results of the different selection schemes for parents and negative selection in EvoDAG, including our proposed selection heuristics. For **parent selection** we tested: tournament selection based on fitness (*fit*), random selection (*rnd*), tournament selection based on the absolute of the cosine similarity (*sim*), and, tournament selection based on the absolute of the Pearson's correlation coefficient (*pearson*). Being our proposed heuristics *sim*, *prs*, and *acc*. For **negative selection**, we tested: negative tournament selection based on fitness (*fit*), and random selection (*rnd*). Notation: the technique for parent selection is followed by the symbol "-", and then comes the abbreviation of the negative selection scheme. Based on macro-F1 ranks, *acc-rnd* got the best results, followed by *acc-fit* and *rnd-rnd*. The techniques *acc-rnd* and *fit-fit* were found statistically different, which means that the use of our heuristic for parent selection based on accuracy and negative selection was statistically better than the use of the classical selection schemes based on fitness. On the other hand, *rnd-rnd* got better results than *fit-fit*, but they were not statistically different. Besides, it was found that the application of the heuristics to all functions with more than one argument (Σ , Π , *max*, *min*, *hypot*, *NB*, *MN*, and *NC*) got generally worse results than applying them only to the functions that they were designed for (Σ , Naive Bayes and Nearest Centroid). The proposed selection heuristics and classical selection schemes were compared with the state-of-the-art selection schemes Angle-Driven Selection (*ads*) [15] and Novelty Search (*nvs*) [62]. Both techniques were implemented in EvoDAG. Based on the experiments, our proposed heuristics and random selection got better results than *ads* and *nvs*. Finally, we compared EvoDAG, with the use of the proposed heuristics, against 16 state-of-the-art classifiers from scikit-learn plus two auto-machine learning libraries: TPOT and autosklearn. The technique that got the best results was TPOT, followed by EvoDAG *acc-rnd*, autosklearn, and EvoDAG *rnd-rnd*. However, there were not found statistical differences between TPOT and EvoDAG *acc-rnd* or EvoDAG *acc-rnd*. On the other hand, TPOT and EvoDAG *fit-fit* were found statistically different. Then, we can interpret that the use of our proposed heuristic based on accuracy for parent selection plus random negative selection statistically im-

proves EvoDAG.



Conclusions

Conclusions

This dissertation has proposed three selection heuristics for parent selection in GP that use functions' properties and individuals' semantics. They are described as follows. First, *tournament selection based on cosine similarity* (sim) aims to promote the selection of parents whose semantics' vectors ideally have rectangle angles. *Tournament selection based on Pearson's correlation coefficient* (prs) aims to promote the selection of parents whose semantics' vectors are uncorrelated. Finally, *tournament selection based on the accuracy* (acc) tries to select parents whose predictions are different, and this is measured with the accuracy score. All of those heuristics were inspired in the properties of the function Σ , and the classifiers Naive Bayes and Nearest Centroid (more details in Chapter 3). To the best of our knowledge, this is the first time in Genetic Programming that functions' properties are taking into account to design methodologies for parent selection.

We performed a comparison of our proposed heuristics against the classical parent selection technique, tournament selection based on fitness, and also with random parent selection. In the case of the last one, it is the most straightforward, and we wanted to analyze its behavior. For negative selection, we tested the use of negative tournament selection based on fitness and random selection. Besides, we also tasted two state-of-the-art selection schemes, Novelty Search and Angle-Driven selection (more details in Section 4.6). We use EvoDAG, a GP system tailored to solve supervised learning problems, to implement and test the different selection techniques (more details in Section 1.4).

The performance of EvoDAG with the different selection schemes was analyzed on thirty classification problems taken from the UCI repository. The datasets were heterogeneous in terms of the number of samples, variables, and some of them are balanced, and others imbalanced. The results showed that the use of our heuristics for

parent selection (sim, prs, and acc) outperformed EvoDAG using the classical tournament selection based on fitness. The heuristic that obtained the best performance was accuracy combined with random negative selection. Besides, it was statistically better than tournament selection based on fitness. It is interesting to note that random selection was competitive, achieving good places in the rank when several combinations for parent and negative selection were studied. Also, our heuristics presented better performance than selection techniques from the state-of-the-art, Novelty Search and Angle-Driven Selection (more details in Section 4.6).

Besides, EvoDAG, with the use of the proposed heuristics, was compared against 18 state-of-the-art classifiers, 16 of them from the scikit-learn python library, and two auto-machine learning algorithms. We confirmed in this experiment that the use of EvoDAG acc-rnd, using accuracy for parent selection and the random negative selection, statistically outperformed EvoDAG fit-fit, using classical selection techniques based on fitness. EvoDAG acc-rnd got the 2nd place in the comparison, and EvoDAG fit-fit appeared on the 7-th position. The results showed that EvoDAG acc-rnd outperformed the scikit-learn classifiers and was competitive against auto-machine learning algorithms. Based on the average rank (measured with macro-F1), the best system was TPOT, which is an auto-machine learning algorithm, the second was EvoDAG acc-rnd, the third position was autosklearn, and the fourth position was EvoDAG rnd-rnd, using random selection for parent and negative selection. Interestingly, the performance of EvoDAG acc-rnd was statistically equivalent to the two auto-machine learning algorithms considered in this comparison. The time required in the training phase of the classifiers was also included in the comparison. The auto-machine learning algorithms were the slowest ones, and the scikit-learn classifiers the fastest. Nonetheless, the difference in time was considerable; TPOT used, on average, more than 57 seconds per sample, autosklearn 11, and EvoDAG less than 5 seconds per instance (more details in Section 4.6).

Discussion

In Chapter 2, we showed a lot of recent documents that have used semantics for improving the performance of GP. Most of them designed crossover and mutation operators. Focus on selection; most of the documents are related to the comparison of individuals' semantics to select parents whose semantics are different. However, to the best of our knowledge, this is the first time that functions' properties are used for guiding the parent selection. For example, the state-of-the-art selection schemes, Angle-Driven Selection [15] and the semantic tournament selection for GP based on statistical analysis of error vectors [18], are applied without take care of functions. In fact, based on our results, we can observed that the proposed selection heuristics work better when they were applied only to the functions that they were designed for (Σ , NB, MN, and NC) than using them in all functions with more than one argument (Σ , Π , max, min, hypot, NB, MN, and NC), this indicates that the functions' properties are an important factor for the selection of arguments in GP.

Recently, it was observed that the use of angles between individuals' semantics in the semantic space is important for selecting individuals. The last year, 2019, two selection schemes that use angle for guiding the selection of individuals were proposed, Angle-Driven Selection (ADS) [15] and Nested Alignment Genetic Programming (NAGP) [98]. Our proposal, specifically, tournament selection based on cosine similarity (sim), is related with those techniques. The comparison is presented in Table 4.8. Besides implementing the proposed selection heuristics, we implemented Angle-Driven Selection (ads) (see Section 4.6) in EvoDAG, the results were that our proposed heuristics outperformed ads. Also, the proposal of ADS establishes that the individuals in the tournaments are first selected using a tournament selection based on fitness. However, we proved that it was better when the individuals in the tournament were randomly selected from the population (see Figure 4.9).

Based on our results, we realized that random selection in the two stages, parent

Table 4.8: Selection techniques that recently use the angle between individuals' semantics

	Angle-Driven Selection (ads)	Selection heuristics (sim, prs, and acc)	Nested Alignment Genetic Programming (NAGP)
GP system	GSGP (see Section 2.2).	EvoDAG (see Section 1.4).	NAGP (see Section 2.3).
Objective	ADS was designed for improving the crossover operator of Geometric Semantic Genetic Programming. The idea is to maximize the angle γ_r between the relative semantics of individuals.	Our selection heuristics aim to select parents based on functions' properties and individuals' semantics, they are tailored to the function Σ and the classifiers Naive Bayes and Nearest Centroid. The idea is to select individuals whose semantics' vectors have rectangle angles (sim), or are uncorrelated (prs), or have different behaviors (acc).	In the system NAGP, the objective is to find a pair of individuals I_1 and I_2 whose angle between $\vec{I}_1 - \vec{t}$ and $\vec{I}_2 - \vec{t}$ is 0. The idea behind this is to accomplish the property of Optimally Aligned Individuals, introduced in [85]. It says that two individuals I_1 and I_2 are optimally aligned if it exists a constant k such that $\vec{I}_1 - \vec{t} = k(\vec{I}_2 - \vec{t})$. This is because finding them, it is possible to calculate the individual whose semantics' vector is equal to the target vector in the semantics space.
Procedure	Given the parents' semantics represented with the vectors \vec{P}_1 and \vec{P}_2 and the target semantics t , ADS aims to maximize the value of $\gamma_r = \arccos\left(\frac{(\vec{t} - \vec{P}_1) \cdot (\vec{t} - \vec{P}_2)}{\ \vec{t} - \vec{P}_1\ \ \vec{t} - \vec{P}_2\ }\right)$ More details in Section 4.6.	Given the parents' semantics represented with the vectors \vec{P}_1 and \vec{P}_2 and the target semantics t , our heuristics tries to minimize the following values: $sim(\vec{P}_1, \vec{P}_2) = \frac{ \vec{P}_1 \cdot \vec{P}_2 }{\ \vec{P}_1\ \ \vec{P}_2\ }$ $prs(\vec{P}_1, \vec{P}_2) = \frac{ (\vec{P}_1 - \vec{P}_1) \cdot (\vec{P}_2 - \vec{P}_2) }{\ \vec{P}_1 - \vec{P}_1\ \ \vec{P}_2 - \vec{P}_2\ }$ $acc(\vec{P}_1, \vec{P}_2) = \frac{1}{n} \sum_i \delta(P_{1i} == P_{2i})$ More details in Sections 3.2.3 and 3.2.4.	The objective is that $\arccos\left(\frac{\vec{I}_1 - \vec{t}}{\ \vec{I}_1 - \vec{t}\ } \cdot \frac{\vec{I}_2 - \vec{t}}{\ \vec{I}_2 - \vec{t}\ }\right) = 0$ More details in Section 2.3
Application	Multivariate symbolic regression	30 classification problems	UCI regression problems
Results	In general, the system ADGSGP has a faster convergence rate than traditionally GP and GSGP, a good interpret ability, and requires less computational effort.	Based on the experiments (see Chapter 4), the combination of selection techniques acc-rnd has significantly better results than selection based on fitness (fit-fit), and the selection of parents based on Novelty Search or Angle-Driven selection.	NAGP outperforms GSGP and GSGP-LS on four complex real-life applications. Its models are not only more effective but also significantly smaller.

and negative selection, outperformed the classical implementation of EvoDAG, where individuals are selected based on fitness. Random selection (rnd-rnd) achieved good places in all the comparisons (see Figures 4.1, 4.9, and 4.11). It was surprising because, in this case, the evolution process is not guided by the fitness of individuals; instead of that, it is random. However, we assume that random selection works well because it promotes population diversity, also, in EvoDAG after selecting the function and its arguments, or parents, a set of parameters θ are estimated with ordinary least squares (OLS) using the target and the individual' semantics. The aim is minimizing the difference between the individual's semantics and the target semantics (see Section 1.4). Another technique that is quite similar to random selection is Novelty Search (NS) [62], which was used for evolving classifiers in [72]. The proposal of NS is abandoning objectives in evolutionary computation, the idea is to replace the fitness function by the novelty of individuals. Table 4.9 shows the comparison between random selection and novelty search. Based on our results (see Section 4.6), random selection is better than novelty search. We assume that the problem of NS is that it searches for individuals that are different from the rest of individuals in the population, and when the functions' arguments are selected, individuals are different from the majority of individuals in the population but maybe among them are similar.

Table 4.9: Comparison of Novelty Search and Random Selection

	Novelty Search (nvs)	Random selection (rnd)
Objective	It tries to select individuals that are different from the rest of individuals in the population.	It randomly selects individuals from the population.
Procedure	Novelty Search selects an individual T maximizing: $\log(\phi(T)) = \sum_j \log\left(\frac{1}{P_j(T_j) + \epsilon}\right)$ where T represents the semantics of the individual, and P a probability function. More details in Section 4.6.	It is effortless, it only selects individuals randomly from the population.
GP system	Originally, it was implemented in M4GP [72] (see Section 2.4), but we also implemented it in EvoDAG.	EvoDAG (see Section 1.4).
Application	11 classification problems	30 classification problems
Results	In terms of performance, results show that all NS variants achieve competitive results relative to the standard approach in GP. Moreover, NS variants got smaller program trees.	Based on the experiments (see Chapter 4), the combination of random selection for parent and negative selection (rnd-rnd) was better than the use of novelty search for parent selection and negative random selection (nvs-rnd).

In 2014, Ingalalli *et al.* affirmed that GP was never regarded as a good method

to perform multi-class classification [46]. Nevertheless recently, Genetic Programming classifiers whose results are competitive with state-of-the-art classifiers have been proposed: EvoDAG [36] in 2016, TPOT [76] in 2016, and M4GP [60] in 2019. Table 4.10 shows a comparison of those GP classifiers. It can be seen that TPOT, M4GP, and EvoDAG, have different approaches, and we want to stand out that EvoDAG evolves full classifiers combining elements from the function and terminal sets, as classical GP methodologies. On the other hand, M4GP uses GP in the first phase of the process for transforming the data space, and, TPOT combines classifiers, preprocessors, and feature selectors from scikit-learn.

Table 4.10: Comparison of recent GP classifiers

	M4GP	TPOT	EvoDAG acc-rnd
Description	The main idea is to transform the original space into another one using functions evolved with GP, then, a centroid is calculated for each class, and the vectors are assigned to the class that corresponds to the nearest centroid using the Mahalanobis distance. In that case, GP is used in the first phase of the classifier, the pre-processing step (see Section 2.4).	It is an auto-machine learning approach that automatically constructs and optimizes machine learning pipelines using GP. It searches the best combination of 14 preprocessors, five feature selectors, and 11 classifiers, all these techniques implemented in scikit-learn [80].	It constructs a full classifier combining elements from a function set and terminal set (see Section 1.4).
Results	M4GP produces better results than classical classifiers as LR, RF, MLP, KNN and RF but it is not better than TPOT.	TPOT is a powerful technique for optimizing machine learning pipelines, however it takes a lot of time.	EvoDAG outperformed 18 scikit-learn classifiers, and its results are statistically similar to TPOT.

Conclusions

Based on our results and observations, we provide the following conclusions:

- Functions' properties are useful for designing parent selection techniques in GP. In this dissertation we used the properties of function Σ and the classifiers Naive Bayes and Nearest Centroid. To the best of our knowledge, this is the first time that functions' properties were used for guiding parent selection in GP.
- The proposed heuristics for parent selection; tournament selection based on cosine similarity (sim), tournament selection based on Pearson's correlation coefficient (prs), and, tournament selection based on the accuracy (acc); improved the performance of EvoDAG. In fact, the selection heuristic based on accuracy

obtained statistically better results than parent selection based on fitness.

- When random selection was used for parent and negative selection, it outperformed the classical selection schemes based on fitness. Besides, it is more straightforward. We assumed that it is because random selection promotes population diversity. Also, when EvoDAG optimizes the functions' parameters θ (see Section 1.4), it uses the target semantics for improving the offspring.
- The use of the selection heuristic based on accuracy for parent selection, and random negative selection, made EvoDAG competitive with state-of-the-art classifiers and auto-machine learning techniques.

Future Work

As future work, it could be interesting to:

- Implement and test the selection heuristics on other GP implementations, as Geometric Semantic Genetic Programming (GSGP) [67], Angle-Driven Geometric Semantic Genetic Programming (ADGSGP) [15], or, M4GP [60].
- Propose new heuristics for regression problems and for other functions.

Bibliography

- [1] ALPAYDIN, E. *Introduction to machine learning*. MIT press, 2009.
- [2] BARLOW, G. J., OH, C., AND GRANT, E. Incremental evolution of autonomous controllers for unmanned aerial vehicles using multi-objective genetic programming. In *IEEE Conference on Cybernetics and Intelligent Systems, 2004*. (2004), vol. 2, IEEE, pp. 689–694.
- [3] BEADLE, L., AND JOHNSON, C. G. Semantically driven crossover in genetic programming. In *2008 IEEE Congress on Evolutionary Computation (IEEE World Congress on Computational Intelligence)* (6 2008), IEEE, pp. 111–116.
- [4] BEADLE, L., AND JOHNSON, C. G. Semantically driven mutation in genetic programming. In *2009 IEEE Congress on Evolutionary Computation* (2009), IEEE, pp. 1336–1342.
- [5] BERGSTRA, J., AND BENGIO, Y. Random Search for Hyper-Parameter Optimization. *Journal of Machine Learning Research* 13, Feb (2012), 281–305.
- [6] BREIMAN, L. Bagging predictors. *Machine Learning* 24, 2 (8 1996), 123–140.
- [7] BRERETON, R. G. Orthogonality, uncorrelatedness, and linear independence of vectors. *Journal of Chemometrics* 30, 10 (10 2016), 564–566.
- [8] BRYANT, R. E. Graph-based algorithms for boolean function manipulation. *Computers, IEEE Transactions on* 100, 8 (1986), 677–691.
- [9] CASTELLI, M., MANZONI, L., MARIOT, L., AND SALETTA, M. Extending local search in geometric semantic genetic programming. In *EPIA Conference on Artificial Intelligence* (2019), Springer, pp. 775–787.
- [10] CASTELLI, M., SILVA, S., AND VANNESCHI, L. A C++ framework for geometric semantic genetic programming. *Genetic Programming and Evolvable Machines* 16, 1 (3 2015), 73–81.

- [11] CASTELLI, M., TRUJILLO, L., VANNESCHI, L., AND POPOVIČ, A. Prediction of energy performance of residential buildings: A genetic programming approach. *Energy and Buildings* 102 (6 2015), 67–74.
- [12] CASTELLI, M., TRUJILLO, L., VANNESCHI, L., SILVA, S., Z-FLORES, E., AND LEGRAND, P. Geometric Semantic Genetic Programming with Local Search. In *Proceedings of the 2015 on Genetic and Evolutionary Computation Conference - GECCO '15* (New York, New York, USA, 2015), ACM Press, pp. 999–1006.
- [13] CASTELLI, M., VANNESCHI, L., SILVA, S., AND RUBERTO, S. How to exploit alignment in the error space: Two different gp models. In *Genetic Programming Theory and Practice XII*. Springer, 2015, pp. 133–148.
- [14] CHEN, Q., XUE, B., MEI, Y., AND ZHANG, M. Geometric semantic crossover with an angle-aware mating scheme in genetic programming for symbolic regression. In *European Conference on Genetic Programming* (2017), Springer, pp. 229–245.
- [15] CHEN, Q., XUE, B., AND ZHANG, M. Improving Generalization of Genetic Programming for Symbolic Regression With Angle-Driven Geometric Semantic Operators. *IEEE Transactions on Evolutionary Computation* 23, 3 (6 2019), 488–502.
- [16] CHEN, Q., ZHANG, M., AND XUE, B. New geometric semantic operators in genetic programming: perpendicular crossover and random segment mutation. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion* (2017), ACM, pp. 223–224.
- [17] CHU, T. H., NGUYEN, Q. U., AND O'NEILL, M. Tournament selection based on statistical test in genetic programming. In *International Conference on Parallel Problem Solving from Nature* (2016), Springer, pp. 303–312.
- [18] CHU, T. H., NGUYEN, Q. U., AND O'NEILL, M. Semantic tournament selection for genetic programming based on statistical analysis of error vectors. *Information Sciences* 436-437 (4 2018), 352–366.
- [19] COMISKY, W., YU, J., AND KOZA, J. Automatic synthesis of a wire antenna using genetic programming. In *Late Breaking Papers at the 2000 Genetic and Evolution-*

- ary Computation Conference, Las Vegas, Nevada* (2000), Citeseer, pp. 179–186.
- [20] DUA, D., AND GRAFF, C. UCI Machine Learning Repository, 2017.
- [21] EIBEN, A. E., AND SMITH, J. *Introduction to Evolutionary Computing*. Springer, 2003.
- [22] FANG, Y., AND LI, J. A Review of Tournament Selection in Genetic Programming. In *International Symposium on Intelligence Computation and Applications ISICA 2010*. Springer, Berlin, Heidelberg, 10 2010, pp. 181–192.
- [23] FEURER, M., KLEIN, A., EGGENSBERGER, K., SPRINGENBERG, J., BLUM, M., AND HUTTER, F. *Efficient and Robust Automated Machine Learning*, 2015.
- [24] FOGEL, L. J., OWENS, A. J., AND WALSH, M. J. Artificial intelligence through a simulation of evolution. In *Evolutionary Computation: The Fossil Record*. Wiley-IEEE Press, 1 1998, pp. 230–248.
- [25] FRIEDMAN, J., HASTIE, T., AND TIBSHIRANI, R. *The elements of statistical learning*, vol. 1. Springer series in statistics New York, 2001.
- [26] FRIEDMAN, J. H., AND HALL, P. On bagging and nonlinear estimation. *Journal of Statistical Planning and Inference* 137, 3 (3 2007), 669–683.
- [27] GALVAN-LOPEZ, E., CODY-KENNY, B., TRUJILLO, L., AND KATTAN, A. Using semantics in the selection mechanism in Genetic Programming: A simple method for promoting semantic diversity. In *2013 IEEE Congress on Evolutionary Computation* (6 2013), IEEE, pp. 2972–2979.
- [28] GALVÁN-LÓPEZ, E., MEZURA-MONTES, E., ELHARA, O. A., AND SCHOENAUER, M. On the use of semantics in multi-objective genetic programming. In *International Conference on Parallel Problem Solving from Nature* (2016), Springer, pp. 353–363.
- [29] GRAFF, M., FLORES, J. J., AND ORTIZ, J. Genetic Programming: Semantic point mutation operator based on the partial derivative error. In *2014 IEEE Interna-*

- tional Autumn Meeting on Power, Electronics and Computing (ROPEC)* (11 2014), IEEE, pp. 1–6.
- [30] GRAFF, M., GRAFF-GUERRERO, A., AND CERDA-JACOBO, J. Semantic crossover based on the partial derivative error. In *European Conference on Genetic Programming* (2014), Springer, pp. 37–47.
- [31] GRAFF, M., MIRANDA-JIMÉNEZ, S., TELLEZ, E. S., AND MOCTEZUMA, D. Evomsa: A multilingual evolutionary approach for sentiment analysis. *arXiv preprint arXiv:1812.02307* (2018).
- [32] GRAFF, M., MIRANDA-JIMÉNEZ, S., TELLEZ, E. S., AND MOCTEZUMA, D. Evomsa: A multilingual evolutionary approach for sentiment analysis. *arXiv preprint arXiv:1812.02307* (2018).
- [33] GRAFF, M., MIRANDA-JIMÉNEZ, S., TELLEZ, E. S., AND MOCTEZUMA, D. Ingeotec at semeval-2018 task 1: Evomsa and μ tc for sentiment analysis. In *Proceedings of The 12th International Workshop on Semantic Evaluation* (2018), pp. 146–150.
- [34] GRAFF, M., TELLEZ, E. S., ESCALANTE, H. J., AND MIRANDA-JIMÉNEZ, S. Semantic genetic programming for sentiment analysis. In *NEO 2015*. Springer, 2017, pp. 43–65.
- [35] GRAFF, M., TELLEZ, E. S., ESCALANTE, H. J., AND ORTIZ-BEJAR, J. Memetic Genetic Programming based on orthogonal projections in the phenotype space. In *2015 IEEE International Autumn Meeting on Power, Electronics and Computing (ROPEC)* (11 2015), IEEE, pp. 1–6.
- [36] GRAFF, M., TELLEZ, E. S., MIRANDA-JIMENEZ, S., AND ESCALANTE, H. J. EvoDAG: A semantic Genetic Programming Python library. In *2016 IEEE International Autumn Meeting on Power, Electronics and Computing (ROPEC)* (11 2016), IEEE, pp. 1–6.
- [37] GRAFF, M., TELLEZ, E. S., VILLASEÑOR, E., AND MIRANDA-JIMÉNEZ, S. Semantic Genetic Programming Operators Based on Projections in the Phenotype Space.

In *Research in Computing Science* (2015), pp. 73–85.

- [38] HARA, A., KUSHIDA, J.-I., AND TAKAHAMA, T. Deterministic Geometric Semantic Genetic Programming with Optimal Mate Selection. In *2016 IEEE International Conference on Systems, Man, and Cybernetics (SMC)* (10 2016), IEEE, pp. 003387–003392.
- [39] HARA, A., KUSHIDA, J.-I., TANEMURA, R., AND TAKAHAMA, T. Deterministic crossover based on target semantics in geometric semantic genetic programming. In *2016 5th IIAI International Congress on Advanced Applied Informatics (IIAI-AAI)* (2016), IEEE, pp. 197–202.
- [40] HARA, A., UENO, Y., AND TAKAHAMA, T. New crossover operator based on semantic distance between subtrees in Genetic Programming. In *2012 IEEE International Conference on Systems, Man, and Cybernetics (SMC)* (10 2012), IEEE, pp. 721–726.
- [41] HARUYAMA, S., AND ZHAO, Q. Designing smaller decision trees using multiple objective optimization based GPs. In *IEEE International Conference on Systems, Man and Cybernetics* (2002), vol. vol.6, IEEE, p. 5.
- [42] HOLLAND, J. H. Genetic Algorithms and the Optimal Allocation of Trials. *SIAM Journal on Computing* 2, 2 (6 1973), 88–105.
- [43] HOLLAND, J. H. *Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence*. MIT Press, 1992.
- [44] HOLM, S. A Simple Sequentially Rejective Multiple Test Procedure, 1979.
- [45] IBA, H., AND SASAKI, T. Using genetic programming to predict financial data. In *Proceedings of the 1999 Congress on Evolutionary Computation, CEC 1999* (1999), vol. 1, IEEE Computer Society, pp. 244–251.
- [46] INGALALLI, V., SILVA, S., CASTELLI, M., AND VANNESCHI, L. A multi-dimensional genetic programming approach for multi-class classification problems. In *European Conference on Genetic Programming* (2014), Springer, pp. 48–60.

- [47] JABEEN, H., AND BAIG, A. R. Two-stage learning for multi-class classification using genetic programming. *Neurocomputing* 116 (9 2013), 311–316.
- [48] KOZA, J. R. *Genetic programming: on the programming of computers by means of natural selection*. MIT Press, 1992.
- [49] KOZA, J. R., BENNETT, F. H., ANDRE, D., KEANE, M. A., AND DUNLAP, F. Automated synthesis of analog electrical circuits by means of genetic programming. *IEEE Transactions on Evolutionary Computation* 1, 2 (1997), 109–128.
- [50] KOZA, J. R., KEANE, M. A., STREETER, M. J., MYDLOWEC, W., YU, J., AND LANZA, G. *Genetic programming IV: Routine human-competitive machine intelligence*, vol. 5. Springer Science & Business Media, 2006.
- [51] KRAWIEC, K. Medial crossovers for genetic programming. In *European Conference on Genetic Programming* (2012), Springer, pp. 61–72.
- [52] KRAWIEC, K. Semantic Genetic Programming. In *Behavioral Program Synthesis with Genetic Programming*. Springer, Cham, 2016, pp. 55–66.
- [53] KRAWIEC, K., AND LICHOCKI, P. Approximating geometric crossover in semantic space. In *Proceedings of the 11th Annual conference on Genetic and evolutionary computation - GECCO '09* (New York, New York, USA, 2009), ACM Press, p. 987.
- [54] KRAWIEC, K., AND PAWLAK, T. Locally geometric semantic crossover. In *Proceedings of the fourteenth international conference on Genetic and evolutionary computation conference companion - GECCO Companion '12* (New York, New York, USA, 2012), ACM Press, p. 1487.
- [55] KRAWIEC, K., AND PAWLAK, T. Quantitative analysis of locally geometric semantic crossover. In *International Conference on Parallel Problem Solving from Nature* (2012), Springer, pp. 397–406.
- [56] KRAWIEC, K., AND PAWLAK, T. Approximating geometric crossover by semantic backpropagation. In *Proceedings of the 15th annual conference on Genetic and evolutionary computation* (2013), ACM, pp. 941–948.

- [57] KRAWIEC, K., AND PAWLAK, T. Approximating geometric crossover by semantic backpropagation. In *Proceeding of the fifteenth annual conference on Genetic and evolutionary computation conference - GECCO '13* (New York, New York, USA, 2013), ACM Press, p. 941.
- [58] KRAWIEC, K., AND PAWLAK, T. Locally geometric semantic crossover: a study on the roles of semantics and homology in recombination operators. *Genetic Programming and Evolvable Machines* 14, 1 (3 2013), 31–63.
- [59] KUO, C.-S., HONG, T.-P., AND CHEN, C.-L. Applying genetic programming technique in classification trees. *Soft Computing* 11, 12 (8 2007), 1165–1172.
- [60] LA CAVA, W., SILVA, S., DANAI, K., SPECTOR, L., VANNESCHI, L., AND MOORE, J. H. Multidimensional genetic programming for multiclass classification. *Swarm and Evolutionary Computation* 44 (2 2019), 260–272.
- [61] LEE, Y.-S., AND TONG, L.-I. Forecasting energy consumption using a grey model improved by incorporating genetic programming. *Energy conversion and Management* 52, 1 (2011), 147–152.
- [62] LEHMAN, J., AND STANLEY, K. O. Abandoning Objectives: Evolution Through the Search for Novelty Alone. *Evolutionary Computation* 19, 2 (6 2011), 189–223.
- [63] LOHN, J. D., HORNBY, G. S., AND LINDEN, D. S. An Evolved Antenna for Deployment on Nasa's Space Technology 5 Mission. In *Genetic Programming Theory and Practice II*. Springer-Verlag, 3 2006, pp. 301–315.
- [64] LOVEARD, T., AND CIESIELSKI, V. Representing classification problems in genetic programming. In *Proceedings of the 2001 Congress on Evolutionary Computation (IEEE Cat. No.01TH8546)* (2001), vol. 2, IEEE, pp. 1070–1077.
- [65] MOCTEZUMA, D., GRAFF, M., MIRANDA-JIMÉNEZ, S., TELLEZ, E. S., CORONADO, A., SÁNCHEZ, C. N., AND ORTIZ-BEJAR, J. A genetic programming approach to sentiment analysis for twitter: Tass'17. In *Proceedings of TASS 2017: Workshop on Sentiment Analysis at SEPLN co-located with 33rd SEPLN Conference (SEPLN 2017)* (2017), vol. 1896.

- [66] MONTES, E. M. *Alternative techniques to handle constraints in evolutionary optimization*. PhD thesis, CINVESTAV-IPN, México City, 2004.
- [67] MORAGLIO, A., KRAWIEC, K., AND JOHNSON, C. G. Geometric semantic genetic programming. In *International Conference on Parallel Problem Solving from Nature* (2012), Springer, pp. 21–31.
- [68] MORAGLIO, A., AND POLI, R. Topological interpretation of crossover. In *Genetic and Evolutionary Computation Conference* (2004), Springer, pp. 1377–1388.
- [69] MUNI, D. P., PAL, N. R., AND DAS, J. A Novel Approach to Design Classifiers Using Genetic Programming. *IEEE Transactions on Evolutionary Computation* 8, 2 (4 2004), 183–196.
- [70] MUNOZ, L., SILVA, S., AND TRUJILLO, L. M3gp–multiclass classification with gp. In *European Conference on Genetic Programming* (2015), Springer, pp. 78–91.
- [71] NAREDO, E., DUARTE VILLASEÑOR, M. A., GARCÍA ORTEGA, M. D. J., VÁZQUEZ LÓPEZ, C. E., TRUJILLO, L., SIORDIA, O. S., NAREDO, E., DUARTE-VILLASEÑOR, M. A., GARCÍA-ORTEGA, M. D. J., VÁZQUEZ-LÓPEZ, C. E., TRUJILLO, L., AND SIORDIA, O. S. Novelty Search for the Synthesis of Current Followers. *Computación y Sistemas* 20, 4 (12 2016), 609–621.
- [72] NAREDO, E., TRUJILLO, L., LEGRAND, P., SILVA, S., AND MUÑOZ, L. Evolving genetic programming classifiers with novelty search. *Information Sciences* 369 (11 2016), 347–367.
- [73] NGUYEN, Q. U., NGUYEN, X. H., O’NEILL, M., AND AGAPITOS, A. An investigation of fitness sharing with semantic and syntactic distance metrics. In *European Conference on Genetic Programming* (2012), Springer, pp. 109–120.
- [74] NGUYEN, Q. U., PHAM, T. A., NGUYEN, X. H., AND MCDERMOTT, J. Subtree semantic geometric crossover for genetic programming. *Genetic Programming and Evolvable Machines* 17, 1 (3 2016), 25–53.
- [75] OH, C. K., AND BARLOW, G. J. Autonomous controller design for unmanned aerial vehicles using multi-objective genetic programming. In *Proceedings of*

- the 2004 Congress on Evolutionary Computation (IEEE Cat. No.04TH8753)* (2004), IEEE, pp. 1538–1545.
- [76] OLSON, R. S., URBANOWICZ, R. J., ANDREWS, P. C., LAVENDER, N. A., MOORE, J. H., ET AL. Automating biomedical data science through tree-based pipeline optimization. In *European Conference on the Applications of Evolutionary Computation* (2016), Springer, pp. 123–137.
- [77] ORTIZ-BEJAR, J., SALGADO, V., GRAFF, M., MOCTEZUMA, D., MIRANDA-JIMÉNEZ, S., AND TELLEZ, E. S. Ingeotec at ibereval 2018 task haha: μ tc and evomsa to detect and score humor in texts. In *Proceedings of the Third Workshop on Evaluation of Human Language Technologies for Iberian Languages (IberEval 2018) co-located with 34th Conference of the Spanish Society for Natural Language Processing (SEPLN 2018)* (2018).
- [78] PAWLAK, T. P., WIELOCH, B., AND KRAWIEC, K. Review and comparative analysis of geometric semantic crossovers. *Genetic Programming and Evolvable Machines* 16, 3 (9 2015), 351–386.
- [79] PAWLAK, T. P., WIELOCH, B., AND KRAWIEC, K. Semantic Backpropagation for Designing Search Operators in Genetic Programming. *IEEE Transactions on Evolutionary Computation* 19, 3 (6 2015), 326–340.
- [80] PEDREGOSA, F., VAROQUAUX, G., GRAMFORT, A., MICHEL, V., THIRION, B., GRISEL, O., BLONDEL, M., PRETTENHOFER, P., WEISS, R., DUBOURG, V., VANDERPLAS, J., PASSOS, A., COURNAPEAU, D., BRUCHER, M., PERROT, M., AND DUCHESNAY, E. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research* 12, Oct (2011), 2825–2830.
- [81] POLI, R., AND LANGDON, W. B. Schema theory for genetic programming with one-point crossover and point mutation. *Evolutionary Computation* 6, 3 (1998), 231–252.
- [82] POLI, R., LANGDON, W. B., AND MCPHEE, N. F. N. *A field guide to genetic programming*. Published via lulu.com and freely available at www.gp-field-

guide.org.uk, 2008.

- [83] QUANG UY, N., HOAI, N. X., AND O'NEILL, M. Semantics based mutation in genetic programming: the case for real-valued symbolic regression. In *15th international conference on soft computing* (2009).
- [84] RECHENBERG, I. Evolution strategie: Optimierung Technisch Systeme nach Prinzipien des Biologischen Evolution. In *Frommann-Hozlboog, Stuttgart* (Stuttgart, Germany, 1973).
- [85] RUBERTO, S., VANNESCHI, L., CASTELLI, M., AND SILVA, S. Esagp - a semantic gp framework based on alignment in the error space. In *European Conference on Genetic Programming* (2014), Springer, pp. 150–161.
- [86] RUBINSTEIN, B. Evolving quantum circuits using genetic programming. In *Proceedings of the 2001 Congress on Evolutionary Computation* (2001), vol. 1, IEEE, pp. 144–151.
- [87] SÁNCHEZ, C. N., DOMÍNGUEZ-SOBERANES, J., ESCALONA-BUENDÍA, H. B., GRAFF, M., GUTIÉRREZ, S., AND SÁNCHEZ, G. Liking product landscape: Going deeper into understanding consumers' hedonic evaluations. *Foods* 8, 10 (2019).
- [88] SANTINI, M., AND TETTAMANZI, A. Genetic programming for financial time series prediction. In *European conference on genetic programming* (2001), Springer, pp. 361–370.
- [89] SCHWEFEL, H.-P. P. *Evolution and optimum seeking: the sixth generation*. John Wiley & Sons, Inc., 1993.
- [90] SMART, W., AND ZHANG, M. Continuously evolving programs in genetic programming using gradient descent. In *Proceedings of THE 7th Asia-Pacific Conference on Complex Systems* (2004).
- [91] STORN, R., AND PRICE, K. Differential Evolution - A Simple and Efficient Heuristic for Global Optimization over Continuous Spaces. *Journal of Global Optimization* 11, 4 (1997), 341–359.

- [92] SUÁREZ, R. R., GRAFF, M., AND FLORES, J. J. Semantic crossover operator for gp based on the second partial derivative of the error function. *Research in Computing Science* 94 (2015), 87–96.
- [93] SUGANUMA, M. A Genetic Programming Approach to Designing Convolutional. *Gecco* (2017), 5369–5373.
- [94] SULLIVAN, K. M., AND LUKE, S. Evolving kernels for support vector machine classification. In *Proceedings of the 9th annual conference on Genetic and evolutionary computation - GECCO '07* (New York, New York, USA, 2007), ACM Press, p. 1702.
- [95] SZUBERT, M., KODALI, A., GANGULY, S., DAS, K., AND BONGARD, J. C. Semantic forward propagation for symbolic regression. In *International Conference on Parallel Problem Solving from Nature* (2016), Springer, pp. 364–374.
- [96] UY, N. Q., HOAI, N. X., O’NEILL, M., MCKAY, R. I., AND GALVÁN-LÓPEZ, E. Semantically-based crossover in genetic programming: application to real-valued symbolic regression. *Genetic Programming and Evolvable Machines* 12, 2 (6 2011), 91–119.
- [97] VANNESCHI, L. An Introduction to Geometric Semantic Genetic Programming. In *Neo 2015*, Oliver Schütze, Leonardo Trujillo, Pierrick Legrand, and Yazmin Maldonado, Eds. Springer, Cham, 2017, pp. 3–42.
- [98] VANNESCHI, L., CASTELLI, M., SCOTT, K., AND TRUJILLO, L. Alignment-based genetic programming for real life applications. *Swarm and evolutionary computation* 44 (2019), 840–851.
- [99] VANNESCHI, L., CASTELLI, M., AND SILVA, S. A survey of semantic methods in genetic programming. *Genetic Programming and Evolvable Machines* 15, 2 (6 2014), 195–214.
- [100] VANNESCHI, L., SILVA, S., CASTELLI, M., AND MANZONI, L. Geometric semantic genetic programming for real life applications. In *Genetic programming theory and practice xi*. Springer, 2014, pp. 191–209.

- [101] VIRGOLIN, M., ALDERLIESTEN, T., AND BOSMAN, P. A. Linear scaling with and within semantic backpropagation-based genetic programming for symbolic regression. In *Proceedings of the Genetic and Evolutionary Computation Conference* (2019), ACM, pp. 1084–1092.
- [102] WILCOXON, F. Individual Comparisons by Ranking Methods. *Biometrics Bulletin* 1, 6 (12 1945), 80–83.