NOVA IMS Information Management School

MAA

Mestrado em Métodos Analíticos Avançados
Master Program in Advanced Analytics

Geometric Semantic Inspired Mutation for M3GP

Ana Sofia Brás Pinto

Dissertation presented as partial requirement for obtaining the Master's degree in Advanced Analytics

NOVA Information Management School
Instituto Superior de Estatística e Gestão de Informação
Universidade Nova de Lisboa

Universidade NOVA de Lisboa
NOVA Information Management School

Geometric Semantic Inspired Mutation for M3GP

Ana Sofia Brás Pinto

Supervisor: Doctor Leonardo Vanneschi

Dissertation presented as a partial requirement for obtaining the
Master's degree in Advanced Analytics
November 2018

Abstract

One of the most challenging Machine Learning tasks is multiclass classification. Genetic Programming (GP) is not able to achieve a very good performance when applied to classification problems with number of classes bigger than two. However, Multidimensional Multiclass Genetic Programming (M2GP) and Multidimensional Multiclass Genetic Programming with Multidimensional Populations (M3GP), two wrapper-based GP classifiers, have shown to be competitive with state-of-the-art classifiers.

The main focus of this work is a new version of M3GP, called Geometric Semantic Inspired M3GP (GSI-M3GP), inspired in geometric semantic operators. GSI-M3GP works in the same way as M3GP, but uses only three operators to create new individuals: add branch, remove branch and a new mutation operator called geometric semantic inspired mutation (gsi-mutation).

In order to test GSI-M3GP and compare it to M3GP, an implementation in Java was developed. Nine different versions of GSI-M3GP were created and tested on eight benchmark problems. For most of the versions of GSI-M3GP, the new algorithm is competitive with M3GP on all the problems. Additionally, it was tested if adding a crossover operator would improve the results, which it did not. A few other alterations were made to the original M3GP algorithm to test the possibility of using the Euclidean distance, instead of the Mahalanobis distance, without harming the quality of the solutions. These alterations do not always maintain the quality of the solutions.

Keywords: Machine Learning, Multiclass Classification, Genetic Programming, Geometric Semantic Genetic Programming,...

Resumo

Uma das tarefas mais desafiantes de Aprendizagem Automática é classificação em mais de duas classes. Genetic Programming (GP) não consegue obter um bom desempenho nestes problemas. No entanto, Multidimensional Multiclass Genetic Programming (M2GP) e Multidimensional Multiclass Genetic Programming with Multidimensional Populations (M3GP), dois algoritmos de classificação que utilizam GP como método *wrapper*, mostraram ser competitivos com classificadores do estado-de-arte.

O foco deste trabalho é a criação de uma nova versão de M3GP, chamada Geometric Semantic Inspired M3GP (GSI-M3GP), inspirada em operadores da geometria semântica. GSI-M3GP funciona da mesma forma que M3GP, mas utiliza apenas três operadores para criar novos indivídulos: adicionar dimensão, remover dimensão e um novo operador de mutação, de nome geometric semantic inspired mutation (gsi-mutation).

Para testar GSI-M3GP e compará-lo com M3GP, foi criada uma implementação em Java. Foram testadas nove versões diferentes de GSI-M3GP em oito problemas de benchmark. GSI-M3GP é competitivo com M3GP em todos os problemas considerados. Foi ainda testado se adicionar um operador de crossover melhoraria os resultados, mas tal não se verificou. Outras alterações foram feitas a M3GP de forma a testar a possibilidade de utilizar a distância Euclideana em vez da distância de Mahalanobis, sem que a qualidade das soluções fosse afetada. Estas alterações nem sempre mantêm a qualidade das soluções.

Palavras-chave: Aprendizagem Automática, Classificação em mais de duas classes, Genetic Programming, Geometric Semantic Genetic Programming,...

Acknowledgements

I would like to express my gratitude towards my supervisor, professor Leonardo Vanneschi. For the enthusiasm he shows while teaching and the way he passes his enthusiasm to the students. For his availability and for giving me the opportunity to work on the amazing idea he had and the operator he created. I would also like to thank professor Sara Silva and her student, João Batista, for helping me with some details of M3GP while I was implementing my own code. I would like to thank professor Jorge Mendes for providing useful book references in statistics.

I would also like to thank Susana and Carina, the amazing women with whom I shared my path during the masters of Advanced Analytics. Susana, without you the first year would not have been the same. I learned so much from you, you are a true inspiration and friend. Carina, thank you for your support during this second year, you made it much easier and fun to handle. Thank you for believing in me the same way I believe in you.

Finally, I would like to express how grateful I am for the people that have always been and will always be here for me: my family and friends. Special acknowledgements go to my two favourite physicists, my brother, my dad and my mum. Thank you to Bruno for being the friend everyone would love to have. No words that can express how much I am thankful for you Pepas - thank you for always being my home. Thank you to my brother, for the eternal love and caring. *Obrigada mãe e pai, por fazerem tudo ao vosso alcance por mim.*

‘The important thing is not to stop questioning. Curiosity has its own reason for existing.’

Albert Einstein

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Genetic Programming (GP) is able to produce good and competitive results for regression problems and binary classification problems. However, in multiclass classification problems, GP is not able to achieve state-of-the-art performance[4]. In 2014, a new multiclass classification algorithm, able to achieve state-of-the-art performance, was proposed[8]. The algorithm is called Multidimensional Multiclass Genetic Programming (M2GP) and its name was chosen based on the fact that, at each iteration, a *population* of *multidimensional solutions* is created. In 2015, a new version of M2GP was introduced: Multidimensional Multiclass Classification Genetic Programming with Multidimensional Populations (M3GP)[15]. While in M2GP the number of dimensions of each solution is fixed, in M3GP the number of dimensions of each solution might differ from *solution* to *solution*. In both algorithms, GP is used as a feature extraction method, then a clustering procedure is applied and, finally, the Mahalanobis distance is used to measure the quality of the feature transformation[15].

Moraglio *et al.* proposed, in 2012, semantic aware genetic operators[14], *i.e.* genetic operators that are able to manipulate the syntax of the solutions in such a way that their effect on semantics is "known". These geometric semantic genetic operators are able to outperform the traditional genetic programming operators, although they might produce more complex and bigger solutions.

The motivation for this thesis comes from the powerful properties of geometric semantic genetic operators and the interesting approach used by M2GP and M3GP to solve multiclass classification problems. The main ambition was to create an operator to M3GP, inspired in the geometric semantic operators, able to change the classification of specific instances without changing the classification of the others. This operator is called *geometric semantic inspired mutation* and it is integrated in a new multiclass classification algorithm - Geometric Semantic Inspired M3GP (in short, GSI-M3GP).

The document is organized as follows. **Chapter 2** introduces the reader to Machine Learning, to Evolutionary Algorithms and to two multiclass classification algorithms which are the base of this thesis' work: the previously referred M2GP and M3GP. **Chapter 3** comprises the original results of M3GP using the MATLAB original implementation[15, 17] and the results obtained with a different implementation, a Java implementation, developed in the context of this thesis. In **Chapter 4**, the Euclidean and Mahalanobis distance measures are discussed and the results of general changes to M3GP are presented. **Chapter 5** presents the new genetic operator and hence, the new multiclass classification algorithm - Geometric Semantic Inspired M3GP. Different versions of the new algorithm are described and the results are also presented. **Chapter 6** introduces a slight variation of the classification algorithm presented in **Chapter 5**, by adding a crossover operator to the set of genetic operators. **Chapter 7** briefly highlights the main implementation issues of M3GP. Finally, **Chapter 8** closes this document with a summary of contributions, what could have been done differently and what can still be improved in the future.

Chapter 2

Background Theory

2.1 Machine Learning

Machine Learning is the area of **Artificial Intelligence** which studies the construction of computer programs/algorithms that learn/improve by means of experience in an automatic way[13]. The learning process of a machine learning algorithm is done using data and, after that, some algorithms are used to make predictions on new data.

The data used by the algorithms in the learning process can be represented in the following format:

$\mathbf{x_1}$	$\mathbf{x_2}$	$\mathbf{x_3}$	...	$\mathbf{x_p}$
x_{11}	x_{12}	x_{13}	...	x_{1p}
x_{21}	x_{22}	x_{23}	...	x_{2p}
...	...	...	...	...
x_{m1}	x_{m2}	x_{m3}	...	x_{mp}

$\mathbf{t}$	t_1	t_2	...	t_m

i.e. as a $m \times p$ matrix, where each line represents an *instance* and each column represents a variable, and additionally, a m-dimensional vector might also be given. Variables $\mathbf{x_1}$ to $\mathbf{x_p}$ are

called *features* and variable **t** is the *target variable*. When a *target variable* exists, then the objective of the algorithm is to, based on the p-dimensional instances, predict the values in **t**.

Machine Learning tasks can be classified according to whether a target variable exists (*i.e.* if data is labelled) and, in that case, also according to its type. If all data is labelled, then we are in the presence of a **Supervised Learning** problem. If data is not labelled, we have, depending on the purpose, an **Unsupervised Learning** problem or a **Reinforcement Learning** problem. If a target variable exists but not all data is labelled, then we are in the presence of a **Semi-Supervised** problem.

2.1.1 Supervised Learning

Supervised Learning problems can be further divided into **Classification** and **Regression** problems. In a **Classification** problem, an instance is classified as belonging to a certain group/class and this class can be represented by an integer. When the target variable is continuous, then the problem is a **Regression** problem.

2.2 Evolutionary Algorithms

Evolutionary Algorithms (EAs) are a family of heuristic population-based optimization algorithms inspired in biological evolution. These algorithms are heuristic since their solutions are created in a way that they are sufficiently good, but the global optimal solution is not necessarily found.

EAs are inspired by biological evolution since they use Charles Darwin's concepts of *reproduction*, *likelihood of survival*, *variation*, *inheritance* and *competition* [5, 18]. As such, instead of creating only one solution that, hopefully, improves from iteration to iteration, a family of solutions (a *population*) is created so that the previously referred biological concepts can be applied. Also, instead of iteration, the word *generation* is used.

Genetic Programming is one of the existing types of **EAs**.

2.2.1 Genetic Programming

Genetic Programming (GP)[11] was presented by John Koza in 1992 and, as an evolutionary algorithm, it *evolves* a population of computer programs[16]. An initial population of programs (*individuals*, in **GP** terms) is created and, at each *generation*, *genetic operators* are applied to the best individuals of the previous population, creating new individuals which might be better than the previous ones. More thoroughly, the basic steps of the **GP** algorithm are described bellow:

1. Randomly *generate* an *initial population* of *individuals*, P;
2. *Evaluate* the individuals' *fitness*;
3. Repeat for a predefined number of *generations* or until another termination condition is met:
   (a) Create an empty population P';
   (b) Repeat until the number of individuals in P' is equal to the number of individuals in P:
      i. *Select* one or two *individuals* from P, with probability of selection based on *fitness*;
      ii. Apply one of the *genetic operators* (with specified probabilities) to the selected individuals to create new individuals;
      iii. Add the previously created individuals to the new population P';
   (c) *Evaluate* the individuals in P';
   (d) Set $P = P'$.
4. Return the individual in P with best *fitness*.

The concepts of *fitness*, *selection* and *genetic operators* are now going to be explained. It will also be explained what a *program/individual* really is, its characteristics, how it is represented and the methods used to *generate* it.

Individuals Representation

In **GP**, *programs* or *individuals* are solutions for the considered problem, and they can be represented in various different ways. One of the most commonly used is the *tree structure*, as shown in figure 2.1.

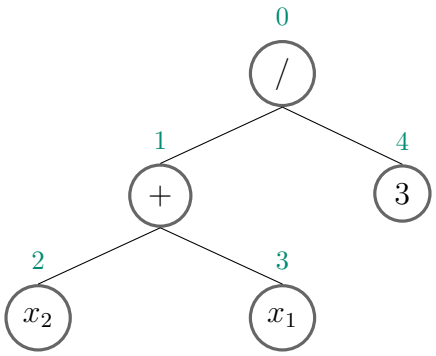


Figure 2.1: GP program/individual.

The *individual* represented in figure 2.1 is the function $f(x_1, x_2) = (x_2 + x_1) \div 3$, where x_1 and x_2 are features of the dataset. The individual might also be represented as

$$(/\ (\ +\ (\ x_2\ x_1)\ 3)$$

i.e. in the called *prefix notation*[16] (always from left to right, as the numbers in figure 2.1 imply). The inspiration for the *prefix notation* comes from Lisp programming language[11]. This notation is useful since it makes it easy to access subtrees of the tree representing the individual.

To construct an individual, two sets are considered: the *function set* and the *terminal set*. The *function set* is the set of *functions* or arithmetic operators, $\mathcal{F} = \{f_1, ..., f_l\}$ (with $l \in \mathbb{Z}$),

and these are present in the trees as *internal nodes*. The *terminal set* consists of features and constant values, $\mathcal{T} = \{t_1, ..., t_j\}$ (with $j \in \mathbb{Z}$), which appear as *leaf nodes* of the trees.

One can consider that the tree in figure 2.1 was created considering, for example, the function set $\mathcal{F} = \{+, -, \times, \div\}$ and the terminal set $\mathcal{T} = \{0, \pm 1, \pm 2, \pm 3\} \cup \{x_1, x_2, x_3\}$.

Each node of the tree has a *depth* value associated to it, which corresponds to the number of edges from the *root node* to it. The *root node* corresponds to the first node to write in the prefix notation (or to the node in the "highest" position when drawing the tree).

The tree itself has a *depth* value associated to it, and it corresponds to the depth of the deepest node in the tree. As an example, the tree represented in figure 2.1 has a depth of 2.

Methods to generate GP individuals,[11]

The individuals in the *initial population* are created using one of the following three known methods (and specifying the *maximum depth*):

- **Full method:**

  Trees created using **Full** always have depth value equal to the maximum depth. While the maximum depth is not reached, nodes are taken at random from the function set. Nodes at depth equal to the maximum depth are randomly chosen from the terminal set. This method produces very regular trees, as the one in figure 2.2.

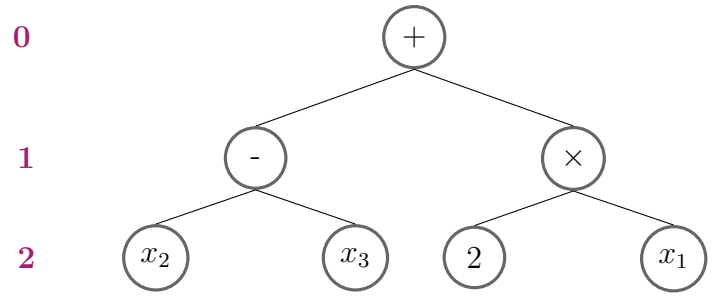


Figure 2.2: Example of a GP individual created using method *Full*, with maximum depth equal to two.

- **Grow method:**

  Grow produces trees which have depth smaller or equal to the predefined maximum. Each node with depth smaller than the predefined maximum is randomly chosen from $\mathcal{T} \cup \mathcal{F}$, so it can be a function or a terminal. If the predefined maximum depth is reached, then nodes with that depth are taken from $\mathcal{T}$. Thus, this method produces very irregular trees, as the ones in figure 2.3, which have depth of two, one and zero, respectively.

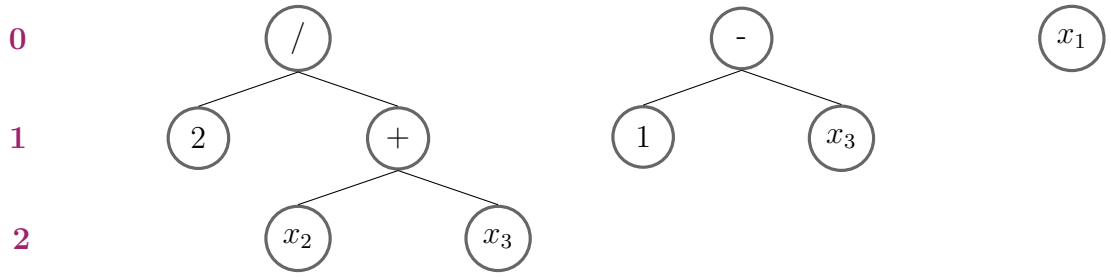


Figure 2.3: Examples of GP individuals created using method *Grow*, with maximum depth equal to two. From left to right, the trees have, respectively, depth of 2, 1 and 0.

- **Ramped half-and-half method:**

  When using **Ramped half-and-half** initialization method, the number of individuals is divided by the maximum depth value, so that an equal percentage of individuals is created with each value of depth between 1 and the maximum depth. Then, for each value of depth, half of the individuals are created using **Full** and the other half is created using **Grow**. This way, the initial population is composed of individuals of different sizes and shapes.

  As an example, if a population is composed of 100 individuals and the maximum depth is 5, then 20% of the individuals will have depth 1, 20% will have depth 2, and so on.

Fitness evaluation of the individuals

GP is an optimization algorithm, so the solutions it produces need to be evaluated. **Fitness** measures how good a solution is, *i.e.* how close it is to the global optimum solution. The name *fitness* is used because of **GP**'s Darwinism inspiration.

A *fitness function* is defined and used to evaluate individuals, so that they can be compared between each other. Examples of *fitness functions* are the sum of absolute errors (used for regression problems) and accuracy (used in classification problems). If considering the sum of absolute errors then the purpose of the optimization problem is to minimize it. When using accuracy, one is optimizing by maximizing accuracy.

Selection

Selection is another biological evolution inspiration. In nature, individuals compete and the best individuals have a higher probability to be chosen for mating. In **GP**, *selection methods* are probabilistic methods used to select individuals. One of the most used is the *tournament selection method* [11, 16].

Tournament selection method: Each time an individual needs to be selected, a *tournament* is performed. A prefixed number of individuals is randomly chosen from the population, with repetition allowed. Then, the individual in the tournament which has the best value of fitness is selected.

Genetic operators

Genetic operators are used to create a new population of individuals given a previous existing population. The *crossover* operator mimics sexual reproduction and *mutation* mimics the fact that each individual, besides having parents' characteristics, also has characteristics of its own that cannot be found in any of the parents.

- **Standard GP Crossover** is a binary operator, needing two individuals to be performed. Two individuals are selected using a *selection* method. These individuals are called *parent individuals*. Crossover starts by first selecting a *crossover point* in each of the parents. These crossover points are randomly and independently selected and correspond to one of the nodes in each parents' tree. Two *offspring individuals* (the resulting individuals

from applying the crossover operator) are created by exchanging the subtrees with root at the crossover points. An example can be found in figure 2.4.

- **Standard GP Mutation** can be seen as a unary operator, as it only needs one *parent individual* to be performed. But it can also be interpreted as a binary operator since, besides needing a parent individual, it also needs a random tree in order to be performed. The *parent individual* is selected using a *selection method*. A node from the parent is chosen at random and with discrete uniform distribution, to be the *mutation point*. The subtree with root at the mutation point, *i.e.* the subtree below the mutation point, is replaced by a new tree - a tree which is randomly generated using *Full* or *Grow*. Figure 2.5 is an example of applying the mutation operator.

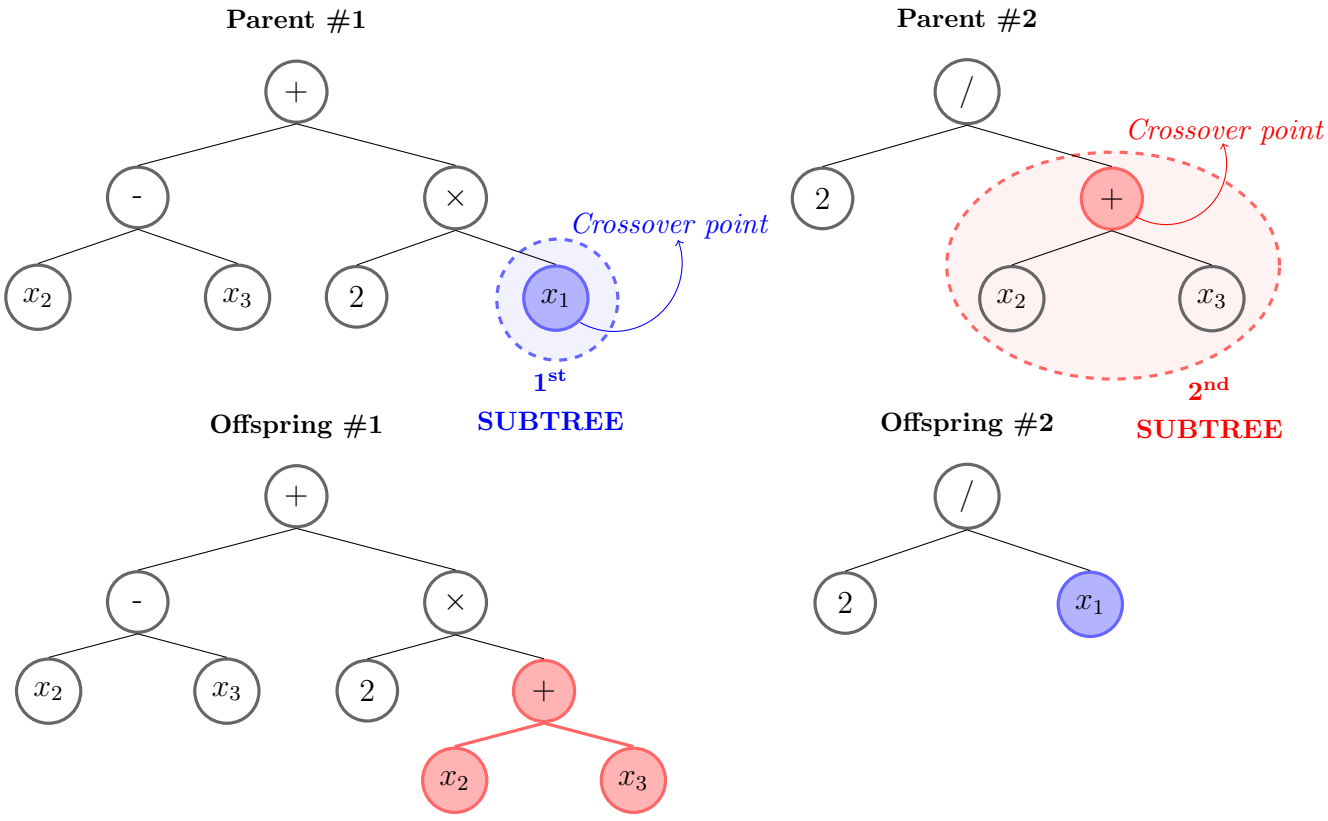


Figure 2.4: Example of individuals generated by the *crossover* operator.

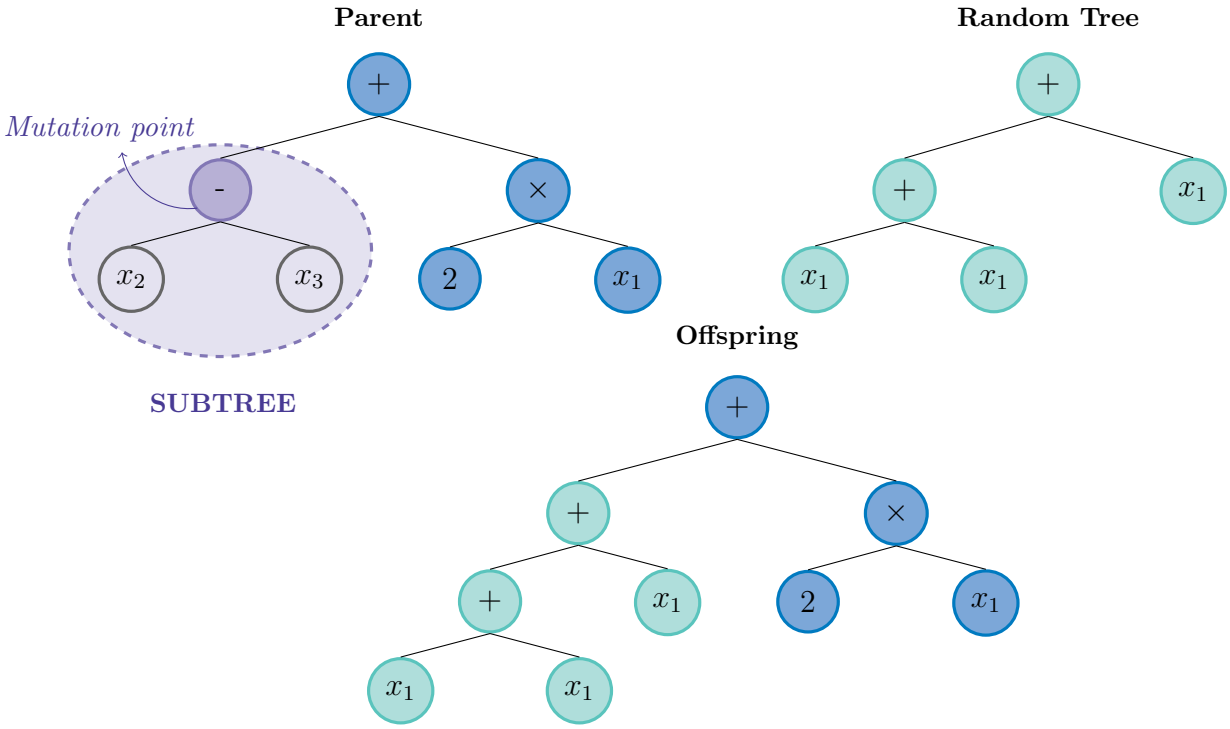


Figure 2.5: Example of an individual generated by the *mutation* operator. The random tree was generated using *grow*.

2.2.2 Geometric Semantic Genetic Programming (GSGP)

The *semantics* of a GP individual can be defined as the m-dimensional point of outputs of the individual on the input data[14, 18]. Mathematically, the *semantics* of an individual P is given by

$$S_P = (P(\vec{y_1}), ..., P(\vec{y_m})),$$

where $\vec{y_i}$ is the i^{th} instance, with $i \in \{1, ..., m\}$, of the given dataset.

Considering a population of individuals, each of the individuals' semantics can be represented as a point in a m-dimensional space. The target variable can also be represented in this space. While one knows the semantics of an individual by knowing its corresponding tree, one does not necessarily knows which tree corresponds to the point representing the target (see figure 2.6). If that was the case, then the considered optimization problem would be solved[18].

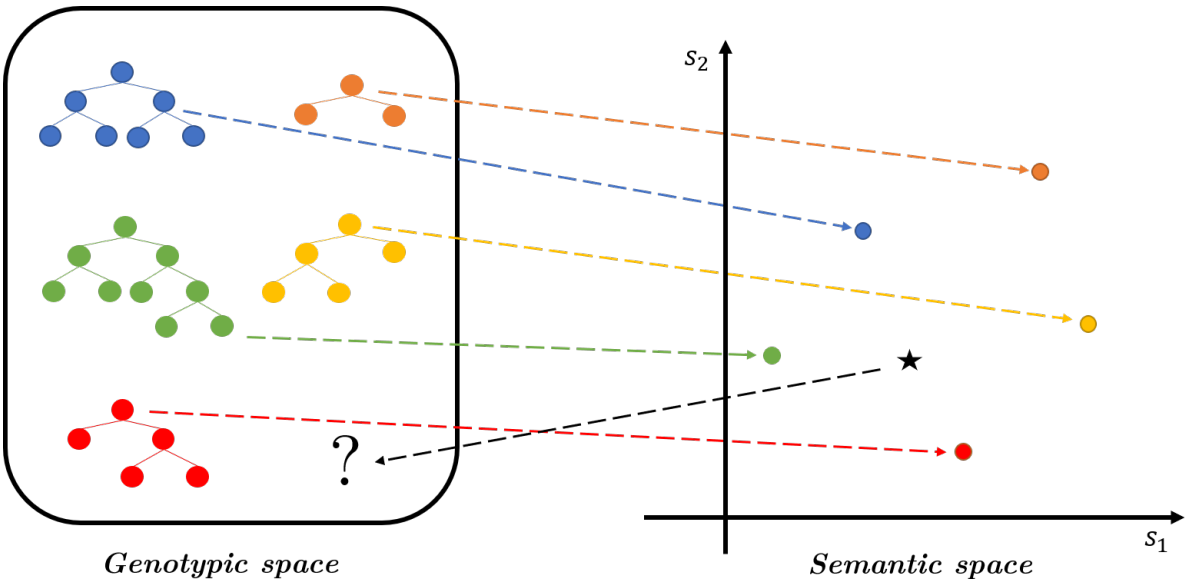


Figure 2.6: Representation of the genotypic space (where individuals are represented by their trees) *versus* the semantic space (where the individuals are represented by their semantics). The semantic space is 2-dimensional since one is considering a toy example of a dataset with only 2 instances. The star corresponds to the target.

In 2012, Moraglio *et al.* presented two operators that act on the semantics of the individuals, transforming the fitness landscape, making it unimodal. When a unimodal fitness landscape is considered, no local optima exists, only the global optimum. This means that there is always chance of improvement. These operators are called **Geometric Semantic Crossover (GSC)** and **Geometric Semantic Mutation (GSM)** [14].

While individuals created using standard **GP** genetic operators have semantics anywhere in the semantic space (and not necessarily close to the semantics of the parents), one knows where the semantics of the individuals created using the geometric semantic operators are, relatively to their parents' semantics. If **GSM** is applied to an individual, the semantics of the offspring is around the parent semantics (see figure 2.7). When **GSC** is applied, the semantics of the offspring is between the parents' semantics (see figure 2.8).

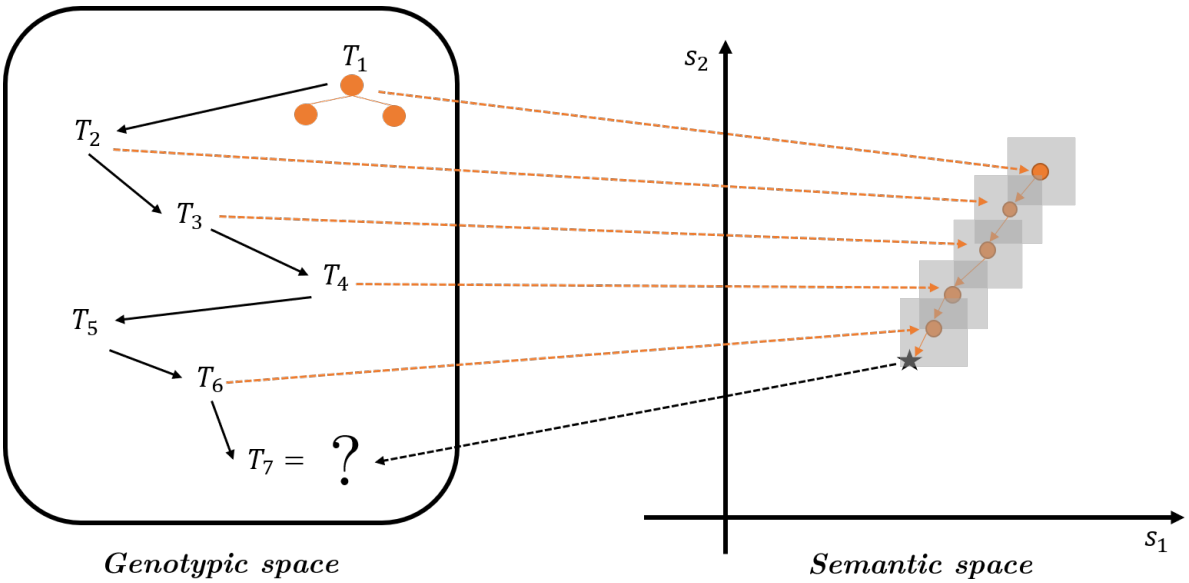


Figure 2.7: Toy example of the application of the geometric semantic mutation. T_1, T_2,...,T_6 is a sequence of trees to which the geometric semantic mutation is applied, where T_{i+1} is generated by applying the operator to T_i. By generating a tree using the mutation operator, the semantics of the offspring is inside a grey square (since the semantic space is 2-dimensional, because one is considering a dataset with 2 instances) with center in the semantics of the parents. There is always a chance of improvement, as expressed here. The star represents que target.

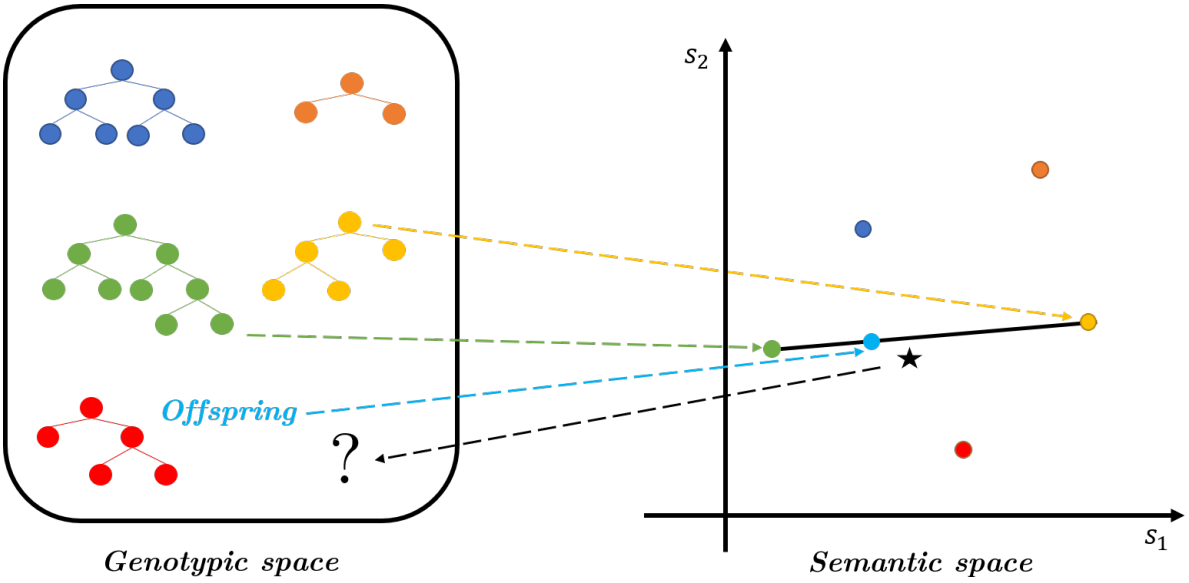


Figure 2.8: Example of applying the geometric semantic crossover operator to two parent trees. The offspring has semantics falling in the line segment joining the semantics of the parents, so the offspring might be closer to the target - the offspring is not worse than the worst of the parents. The semantic space is bi-dimensional since a 2 instance dataset is being considered, for simplicity. The star represents the target.

The big disadvantage of **GSGP** is that the individuals created by **GSC** and **GSM** are always bigger than their parents and so, from generation to generation, the individuals grow very quickly[6], making **GSGP** impossible to use in some cases. In 2013, a new implementation was proposed[19], for which not all the individuals had to be stored, only their semantics, making **GSGP** able to be used.

Geometic semantic operators

- **Geometric Semantic Crossover**

  Let $P_1, P_2 : \mathbb{R}^p \longrightarrow \mathbb{R}$ be two parent individuals, chosen using a selection method (described in the previous subsection). The *offspring* individual, $O_{XO} : \mathbb{R}^p \longrightarrow \mathbb{R}$, is the following:

  $$O_{XO} = P_1 \times R + P_2 \times (1 - R),$$

  where R is a randomly generated tree with output values in [0,1].

- **Geometric Semantic Mutation**

  Let P be the parent individual, $P : \mathbb{R}^p \longrightarrow \mathbb{R}$, chosen using a selection method. The *offspring* individual, $O_M : \mathbb{R}^p \longrightarrow \mathbb{R}$, is the following:

  $$O_M = P + ms \times (R_1 - R_2),$$

  where R_1 and R_2 are randomly generated trees, and ms is a predefined real number called *mutation step*.

2.3 Multiclass classification problems and algorithms

Let us consider a dataset $\mathcal{D}$ with m instances and p features. This can be represented by a $m \times p$ matrix. Let us also consider a vector $\vec{t} \in \mathbb{R}^m$ (the *target* variable values), and the set of classes $\mathcal{C} = \{1, 2, ..., c\}$.

Each one of the m p-dimensional instances of $\mathcal{D}$ belongs to one of the classes in $\mathcal{C}$. The i^{th} instance belongs to the class stored in the i^{th} entry of $\vec{t}$.

Let us consider a partition of dataset $\mathcal{D}$ in two datasets: the *training set* and the *test set*. The *training set* ($\mathcal{D}|_T$, with n instances) is the set of instances used to create solutions/models, and the *test set* ($\mathcal{D}|_U$, with m-n instances) is the set used to check the models' *generalization ability*. The models are created in such a way that the accuracy of predictions is maximized (or, equivalently, another metric might be used). However, the model needs to generalize well on unseen data. That is, the model also needs to be able to predict accurately the class to which an unseen instance belongs. If that happens, the model has *generalization ability*. Otherwise, it *overfits* training data, *i.e.* it just "mimics" training data.

Given a new instance $\vec{y} \in \mathbb{R}^p$, to which class does $\vec{y}$ belong to? A classification algorithm solves this problem by, based on data in $\mathcal{D}$, finding a function $f : \mathbb{R}^p \longrightarrow \mathcal{C}$, *i.e.* a function that assigns a class to a given instance, and so by applying f to $\vec{y}$ one gets $f(\vec{y}) = a \in \mathcal{C}$.

2.3.1 Multidimensional multiclass Genetic Programming (M2GP)

In 2014, Ingalalli *et al.* presented a new multiclass classification algorithm called **Multidimensional Multiclass Genetic Programming**[8] (from now on **M2GP**).

In **M2GP**, **GP** is used as a *wrapper method*. That is, **GP** is used as a feature extraction method, transforming the original set of p variables in a new set of d variables. The quality of this transformation is then evaluated by a predefined measure[15].

Thus, two steps are considered:

1. Creation of a d-dimensional tree: $g : \mathbb{R}^p \longrightarrow \mathbb{R}^d$, $d \geq 1$;
2. Evaluation of the tree defined by function g: $h : \mathbb{R}^d \longrightarrow \mathcal{C}$;

GP typically transforms a set of p variables in only one variable. This is not considered informative enough in muticlass classification[8]. Therefore, instead of having a typical **GP** tree, **M2GP** creates trees with d *dimensions* (also called *branches* in **M2GP** terms), where $d \geq 1$.

Figure 2.9 presents possible multidimensional trees. Variables x_1 and x_2 are 2 of the p features in the dataset. The blue squared nodes are the root nodes of each one of the trees. In multidimensional trees, the root node exists just to define the number of *dimensions* of the tree.

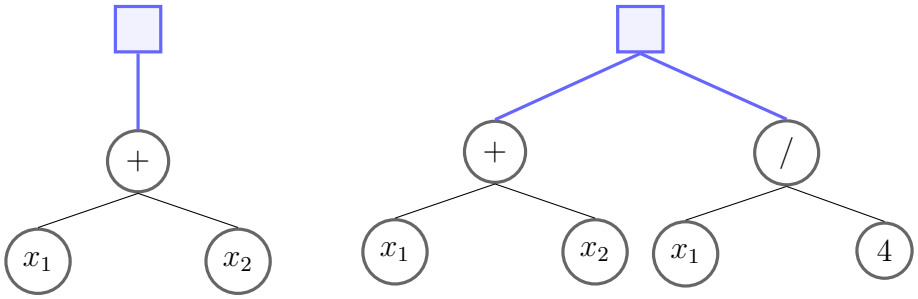


Figure 2.9: Example of a *one*-dimensional tree and a *two*-dimensional tree.

It is important to notice that the number of *dimensions*, d, is a parameter of the algorithm and so, it does not depend on the number of classes/features/*etc.*

Applying function g to each one of the n training instances, one gets n d-dimensional points that can be represented in a d-dimensional space. Let us refer to these points as the *mapped training instances*. The *mapped training instances* are then clustered according to the classes in $\mathcal{C}$, and classified using function h.

The fact that **GP** is only used as a wrapper method, implies that an **M2GP** solution does not consist only of the tree, but also of other structures that are used in the evaluation of the individual[8].

Applying g to the $m-n$ test instances in $\mathcal{D}$, one gets a $(m-n) \times d$ matrix called, from now on, *mapped test instances*. The previously mentioned evaluation structures are used to evaluate these outputs and assign a class to each one of the test instances.

M2GP algorithm - Training phase

Let:

- $\mathcal{D}|_T$ refer to the set of $n \times p$ training instances in $\mathcal{D}$;
- $\vec{t}\,|_T$ refer to the n dimensional vector of training target values;
- d be a predefined number representing the number of *dimensions* of the trees;
- $\mathcal{C} = \{1, ..., c\}$ be the set of classes in which instances can be grouped;

be the inputs of the training phase of **M2GP** algorithm.

Then, the *pseudo-code* of each generation's training phase of the algorithm[8] is organized below:

1. *C*reate an empty population P.
2. *Repeat* until there are *pop* elements in P:
   (a) *Generate* a d-dimensional tree, P_i, $i \in \{1, ..., pop\}$, using a *genetic operator*;
   (b) *Evaluate* P_i on $\mathcal{D}|_T$ getting the matrix MT, the matrix of mapped training instances of P_i;
   (c) Add the j^{th} line in MT to Z^k if $\vec{t}\,|_T^j = k$, $\forall j \in \{1, ..., n\}$, where $\vec{t}\,|_T^j$ is the j^{th} entry of $\vec{t}\,|_T$;
   (d) *For* $k \in \mathcal{C}$:
      i. *Create* $C_k = covariance(Z^k)$;

ii. *Create* $M_k = centroid(Z^k)$;

iii. *Calculate* $D_j^k = \sqrt{(MT_j - M_k)^T C_k^{-1}(MT_j - M_k)}$, $\forall j \in \{1, ..., n\}$, where MT_j is the j^{th} line of MT;

(e) *For* $j \in \{1, ..., n\}$

i. $Pred_j = k$, where $k \in \mathcal{C}$ such that $D_j^k = min\{D_j^1, ..., D_j^c\}$

ii. $Matched_j = \begin{cases} 1 & \vec{t}\,|_T^j = Pred_j \\ 0 & \vec{t}\,|_T^j \neq Pred_j \end{cases}$

(f) *Evaluate* the *training fitness* of I_i: $f_T(I_i) = \frac{1}{n}\sum_{j=1}^{n} Matched_j$.

(g) Add individual $I_i = (P_i, C_1, ..., C_c, M_1, ..., M_c)$ to the population.

Initial population

The trees in the initial population of **M2GP** are created in the same way as for **GP** - using *Full*, *Grow* or *Ramped half-and-half methods*. The only difference here is that the trees created for **M2GP** are d-dimensional and the root node is not a "real" node, but the node defining the number of dimensions. The results presented in [8] were ran using Ramped half-and-half with 75% Full and 25% Grow.

Genetic operators

The genetic operators are also the same as the ones used in **GP**: mutation and crossover. The only restriction in **M2GP** is that the root node cannot be chosen as mutation/crossover point.

Calculation of the mapped instances

As previously referred, by applying the tree to an instance of $D|_T$, one gets a d-dimensional point. Considering all the instances in $D|_T$, the mapped training instances are considered as points in the d-dimensional, and grouped in clusters.

Clustering

The clustering phase consists of grouping the mapped training instances according to the class to which the training instances belong. That is, the mapped training instances are divided in c groups (and c matrices are created, one corresponding to each class). According to the *pseudo-code* in page 15 and 16, these matrices are called Z^k, with $k \in \{1, ..., c\}$.

Covariance matrices, centroids and Mahalanobis distance

For each one of the Z^k matrices, a covariance matrix, $C_k = covariance(Z^k)$, and a centroid, $M_k = centroid(Z^k)$, are created. The entry (a, b) of C_k stores the covariance between column a and column b of Z^k. Hence, each Z^k matrix is $d \times d$. The size of $M_k = centroid(Z^k)$ is d, and entry a of M_k (with $a \leq d$) stores the average of Z^k's a^{th} column.

M2GP is a classification algorithm. As such, it classifies instances into classes, *i.e.* it makes predictions. In **M2GP** the predicted values are found with the help of the covariance matrices and the centroids. The distances between the mapped training instances and all the centroids are considered. Each instance is assigned with the class for which the distance from the mapped training instance to the classe's centroid is minimized. Thus, each instance is assigned with the class represented by that centroid.

The question is, *why are the covariance matrices needed?* The covariance matrices are needed because the Mahalanobis distance measure is the distance measure considered. In [8] two distance measures are compared: the Euclidean and the Mahalanobis distance measures. According to the authors, "the distance measure indeed plays a significant role in the performance of M2GP" and, by using the Mahalanobis distance, **M2GP** is able to reach significantly better results.

The Mahalanobis distance between an observation $x = (x_1, ..., x_d)$ and a set of samples characterized by a mean point $\mu = (\mu_1, ..., \mu_d)$ and a covariance matrix $S_{d \times d}$ is given by:

$$d_M(x) = \sqrt{(x-\mu)^T S^{-1}(x-\mu)}$$

It is important to notice that, by using the Mahalanobis distance, one takes into consideration the correlations between variables. That is, the Mahalanobis distance assumes that variables are correlated and gives less importance to the variables with higher variances and to groups of variables with high correlations. When using the Euclidean distance, one assumes that variables are uncorrelated and have equal variance[10]. This topic will be addressed again in chapter 4.

Finally, looking at the Mahalanobis distance formula, one can see that it requires the calculation of the inverse of S. As such, it is only possible to calculate the Mahalanobis distance if the covariance matrix is nonsingular.

Fitness function

The fitness function is the accuracy of the classification: the fraction or, equivalently, the percentage of correctly classified instances (as described in step 2.(f) of the algorithm in page 16).

$$\text{Accuracy} = \frac{\text{Number of correctly classified instances}}{\text{Total number of instances}}$$

What is an individual?

Unlike **GP**, an **M2GP** individual consists not only of the function tree, but also, the covariance matrices and the centroids, as described in step 2.(g) of the training phase algorithm in page 16).

Testing phase

Let:

- $\mathcal{D}|_U$ refer to the set of $(m-n) \times p$ test instances in $\mathcal{D}$;
- $\vec{t}\,|_U$ refer to the $m-n$ dimensional vector of test target values;
- d a predefined number representing the number of *dimensions* of the trees;
- $\mathcal{C} = \{1, ..., c\}$ be the set of classes in which instances can be grouped;
- P a population of *pop* individuals - where each individual is composed by its tree, P_i, and the corresponding covariance matrices (C_1,...,C_c) and centroids (M_1,...,M_c).

be the inputs of the testing phase of **M2GP** algorithm.

Then, the *pseudo-code* of each generation's testing phase of the algorithm[8] is organized below:

1. *For* each individual, $I_i \in P$ (with $i \in \{1, ..., pop\}$):
   (a) *Evaluate* P_i on $\mathcal{D}|_U$ getting the matrix MU, the matrix of mapped test instances of P_i;
   (b) *For* $k \in \mathcal{C}$:
      i. *Calculate* $D_j^k = \sqrt{(MU_j - M_k)^T C_k^{-1} (MU_j - M_k)}$, $\forall j \in \{1, ..., m-n\}$, where MU_j is the j^{th} line of MU;
   (c) *For* $j \in \{1, ..., m-n\}$
      i. $Pred_j = h$, where $h \in \mathcal{C}$ such that $D_j^h = min\{D_j^1, ..., D_j^c\}$
      ii. $Matched_j = \begin{cases} 1 & \vec{t}\,|_U^j = Pred_j \\ 0 & \vec{t}\,|_U^j \neq Pred_j \end{cases}$
   (d) *Evaluate* the *test fitness* of I_i: $f_U(I_i) = \frac{1}{m-n} \sum_{j=1}^{m-n} Matched_j$.

The most important thing to refer is that, when classifying test data, the covariance matrices and the centroids are not recalculated.

Trees' number of branches, d

The main drawback of **M2GP** is that the number of dimensions is not automatically chosen, it is a parameter. And as a parameter, some questions can be posed, such as: *what is its best value? Does this value change along the generations? Does it change from problem to problem?*

The authors introduce, in [8], an automatic procedure to choose the number of dimensions. The procedure consists of creating initial populations with increasing values of d, while the accuracy of those populations increases. The procedure stops when the accuracy decreases. The chosen value of d is the last value for which the accuracy of the initial population increased. As such, d depends on the considered dataset.

2.3.2 M2GP with Multidimensional Populations (M3GP)

In 2015, Delgado *et al.* presented an improved version of **M2GP**, called **Multidimensional Multiclass Genetic Programming with Multidimensional Populations** or **M2GP with Multidimensional Populations** (in short, **M3GP**)[15].

The main drawback of **M2GP** pointed out by Delgado *et al.* is the fact the the number of dimensions is a parameter and its value is set before running the algorithm. **M3GP** does not have the number of dimensions as a parameter fixed before running the algorithm. **M3GP** *evolves* it with the algorithm.

Initial population

The initial population is composed by *one*-dimensional individuals. These individuals are created using *Full* method[11] with depth of 6.

Genetic operators

The *genetic operators* used to create individuals in **M3GP** are different from the ones considered in **M2GP**. *Mutation* and *crossover* can be chosen to create a new individual with equal probability.

If *Mutation* is chosen, one of the following operators is randomly chosen, with equal probability:

- **Standard subtree mutation**: a mutation point is randomly selected (excluding the root node) and the subtree below the mutation point is replaced by a randomly created tree;
- **Add new dimension**: a randomly created new branch is added to the tree, *i.e.* a *one*-dimensional tree is added as a branch of the parent's tree.
- **Remove existing dimension**: a branch of the tree is randomly selected to be removed from the tree;

If *Crossover* is chosen, one of the following two operators is applied, both having the same probability of being chosen:

- **Standard subtree crossover**: two crossover points are randomly and independently selected (excluding the root node) from two parent trees and the subtrees below the points are interchanged;
- **Swap dimensions**: A randomly chosen dimension from a parent tree is replaced by a randomly chosen dimension from another parent's tree;

Hence, *Mutation* is responsible by increasing and decreasing the number of dimensions of the individual.

Pruning procedure

As just referred, *Mutation* is responsible for adding and removing dimensions to the individuals. However, as expressed in [15], an excessive number of dimensions might decrease the accuracy and, on the other side, the remove dimensions' operator removes a random branch, which might also decrease accuracy. Because of this fact, the authors say that the remove dimensions' operator is "blind to fitness".

The *pruning procedure* was created to counteract this problem and it is applied, at each generation, to the tree corresponding to the individual with higher training accuracy in the population. Let T represent that tree. The *pseudo-code* for the *pruning procedure* is:

1. Let d be the number of dimensions of T;
2. Set $i := 1$;
3. *Repeat* while $i \leq d$:
   (a) Let T' be the tree found by removing the i^{th} dimension of T;
   (b) *If* Training Accuracy(T')>Training Accuracy(T) *Then*:
      i. Set $T := T'$;
   (c) *Else*:
      i. Increment the value of i;
4. Return T.

The pruned tree replaces the previous best tree in the population.

Chapter 3

M3GP algorithm original results

As referred in section 2.3.2, **M3GP** was first presented in 2015 by Delgado *et al.*[15], and a modified version of GPLAB 3 (an open source GP toolbox for MATLAB, available in http://gplab.sourceforge.net)[17] was used to get all the results.

The results presented in this document were obtained using a new implementation developed in the context of this thesis - a Java implementation.

3.1 Experimental setup and results

M3GP was originally run on 8 problems, i.e. on 8 datasets with varying number of classes, attributes and samples (see table 3.1). These are the problems which are going to be considered for this experimental analysis (and also in the following chapters).

Dataset	HRT	IM-3	WAV	SEG	IM-10	YST	VOW	M-L
# classes	2	3	3	7	10	10	11	15
# attributes	13	6	40	19	6	8	13	90
# samples	270	322	5000	2310	6798	1484	990	360

Table 3.1: Description of the datasets used for the experimental analysis.

Datasets HRT, SEG, YST, VOW and M-L can be found at the KEEL dataset repository[2], while IM-3 and IM-10 are the satellite datasets used in [1] and WAV can be found in [3].

Each of these datasets is of the following format:

$\mathbf{x_1}$	$\mathbf{x_2}$	$\mathbf{x_3}$	...	$\mathbf{x_p}$	$\mathbf{t}$
x_{11}	x_{12}	x_{13}	...	x_{1p}	t_1
x_{21}	x_{22}	x_{23}	...	x_{2p}	t_2
...	...	...	...	...	...
x_{m1}	x_{m2}	x_{m3}	...	x_{mp}	t_m

where p is the number of attributes and m is the number of samples, and so each of the lines in the dataset is a sample. The last column of each dataset is the target variable, the variable with the values to predict.

The following table (table 3.2) presents the running parameters of the original M3GP[15], which we are leaving unchanged in the first subsection, to have a fair comparison between the two implementations.

Runs	30
Population size	500 individuals
Generations	100 generations
Initialization	6-depth Full Initialization[11]
Operator probabilities	Crossover $p_c = 0.5$, Mutation $p_\mu = 0.5$
Function set	$+$, -, $\times$, $\div$ protected as in [11]
Terminal set	Ephemeral random constants [0,1]
Bloat control	17-depth limit[11]
Selection	Lexicographic tournament[12] of size 5
Elitism	Keep best individual

Table 3.2: Running parameters of M3GP.

For each run of the algorithm on a specific dataset, the dataset is randomly split into training and test sets: 70% of samples for training and the remaining 30% for test (more specifically, the samples were shuffled and then the partition was made).

The comparisons between algorithms/versions are presented in terms of training accuracy, test accuracy and number of nodes. Additionally, the number of dimensions is presented. Notice that the number of dimensions is only informative, as a bigger number of dimensions does not imply that the individual has more nodes.

Two results are said to be significantly different (throughout all the document), when their difference is statistically significant according to the Wilcoxon's rank sum test considering a significance level of 1% (0.01).

3.1.1 Original results

This subsection presents a comparison between the original results of M3GP (obtained using GPLAB toolbox[17]) and the results obtained with the Java implementation.

As expected, there are no discrepancies between the results obtained with the original implementation of M3GP using the GPLAB toolbox from MATLAB and the new Java implementation.

	HRT	**IM-3**	**WAV**	**SEG**	**IM-10**	**YST**	**VOW**	**M-L**
Training accuracy								
M3GP original[15]	94.7	99.6	90.7	98.1	93.0	68.5	100	100
M3GP new	95.0(2.6)	99.6(0.4)	90.8(0.4)	98.1(0.4)	93.3(0.4)	69.1(1.2)	100(0.0)	100(0.0)
Test accuracy								
M3GP original	79.0	95.4	84.3	95.6	91.0	56.3	93.8	57.1
M3GP new	79.0(4.5)	95.9(2.5)	84.0(0.9)	95.7(0.8)	91.3(0.7)	56.6(2.0)	95.5(1.9)	62.5(7.4)
# of nodes								
M3GP original	110	66	71	111	239	274	53	13
M3GP new	318(45-422)	66(35-701)	132(39-213)	367(62-367)	777(261-1283)	317(203-888)	42(42-72)	11(11-209)
# of dimensions								
M3GP original	12(1-17)	5(2-8)	31(29-37)	11(5-21)	12(11-16)	13(11-18)	20(16-20)	12(10-13)
M3GP new	13(3-26)	5(3-13)	34.5(25-38)	11.5(5-17)	17(10-21)	14(11-22)	23(20-24)	11(8-13)

Table 3.3: Comparison between the original[15, 17] and new implementation of M3GP.

Table 3.3 presents results of training accuracy and test accuracy that refer to a median of 30 runs, and in parenthesis the standard deviation (for the new implementation). There is also information regarding the number of nodes: for *M3GP original* the number of nodes of the best individual; for *M3GP new* the number of nodes of the best individual and, in parenthesis, the

mininum and maximum number of nodes obtained on the 30 runs. Additionally, the median number of dimensions and, in parenthesis, the minimum and maximum number of dimensions.

3.1.2 M3GP with a smaller population

In the original version of M3GP[15], the population size was set to 500 individuals. Per generation, $\{500 \times \#Classes\}$ cluster centroids, $\{500 \times \#Classes\}$ covariance matrices are created. For each invertible matrix its inverse is also created. This is a process which might take a lot of time (which depends on the dataset, on the implementation and on the machine used to run the algorithm), if the number of classes or the number of samples is high.

Therefore, here M3GP is ran setting the population size to 50 (while the remaining parameters of table 3.2 are left unchanged), and compared to the algorithm ran with 500 individuals. The results are stored in table 3.4. The values in bold represent the best version (between *M3GP 500* and *M3GP 50*), regarding training accuracy and test accuracy. When the difference is not statistically significant, both values are marked in bold.

	HRT	IM-3	WAV	SEG	IM-10	YST	VOW	M-L
Training accuracy								
M3GP 500	**95.0**(2.6)	**99.6**(0.4)	**90.8**(0.4)	**98.1**(0.4)	**93.3**(0.4)	**69.1**(1.2)	**100**(0.0)	**100**(0.0)
M3GP 50	93.1(3.2)	98.7(0.6)	89.6(0.6)	96.4(0.7)	92.3(0.8)	66.0(5.0)	**100**(0.0)	**100**(0.6)
Test accuracy								
M3GP 500	**79.0**(4.5)	**95.9**(2.5)	84.0(0.9)	**95.7**(0.8)	**91.3**(0.7)	**56.6**(2.0)	**95.5**(1.9)	**62.5**(7.4)
M3GP 50	**77.8**(4.0)	**94.3**(2.6)	**84.7**(0.9)	95.1(1.0)	**91.1**(0.9)	**55.6**(3.8)	**94.3**(2.3)	**60.6**(5.9)
# of nodes								
M3GP 500	318(45-422)	**66**(35-701)	132(39-213)	367(62-367)	777(261-1283)	**317**(203-888)	**42**(42-72)	**11**(11-209)
M3GP 50	**176**(45-359)	542(39-542)	**42**(20-177)	**73**(20-279)	**258**(85-818)	560(78-573)	45(45-111)	12(12-163)
# of dimensions								
M3GP 500	13(3-26)	5(3-13)	34.5(25-38)	11.5(5-17)	17(10-21)	14(11-22)	23(20-24)	11(8-13)
M3GP 50	14(1-19)	5.5(2-10)	26.5(20-32)	8(5-14)	15(8-24)	12(2-17)	22(19-25)	11(8-12)

Table 3.4: Comparison between M3GP with 500 individuals and 50 individuals.

Regarding training accuracy, *M3GP 500* is always significantly better than *M3GP 50* on all the problems (except for VOW and M-L, for which the median values are equal). Test accuracy was not significantly distinct between the two versions on all the datasets except for SEG and WAV datasets: for SEG, the version with 500 individuals was better in terms of test accuracy; for WAV, test accuracy was better setting the population size to 50 individuals.

Regarding number of nodes, the range of values ($max - min$) is always smaller when the population size is 50 (except for VOW dataset).

Given that running M3GP with a population of 50 individuals instead of 500 is much quicker (since it requires the computation of 10 times less individuals) and there is no statistical difference between the test values (except for SEG and WAV dataset, and for WAV it is better to have 50 individuals), in chapters 5 and 6 the experimental analysis is performed setting the population size to 50.

Chapter 4

General changes to M3GP

The use of the Mahalanobis distance can be seen as a disadvantage of M3GP, since it requires the computation of big matrices (although symmetric) and their inverses. Previous versions of the algorithm used the Euclidean distance, but the solutions were significantly worse (in terms of accuracy) than the ones found using the Mahalanobis distance[8]. Although, in our daily life, we are used to the Euclidean distance, by using it we assume that[10]:

- all variables have the same variance;
- variables are uncorrelated;

This is something that does not happen in M3GP, more specifically, it does not happen with the *branches/dimensions* of an M3GP *solution/individual*. Each *branch* is a variable, from now on referred to as w_i. These w_i variables are functions of the original features present in the dataset.

To better understand this, one can take a look at the tree in figure 4.1, which is a possible M3GP *solution*. There are two dimensions: $w_1(x_1, .., x_p) = x_1$ and $w_2(x_1, .., x_p) = 0.01x_1$ (where x_i, $i \in \{1, ..., p\}$, are the feature variables of the dataset).

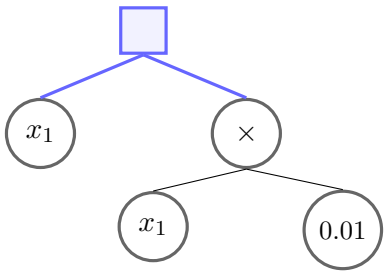


Figure 4.1: A possible M3GP *solution*.

The two dimensions of the tree are perfectly correlated (their correlation is 1) and have different variances (since the values of $w_2(x_1, ..., x_p)$ are always 100 times smaller than the values of $w_1(x_1, ..., x_p)$).

4.1 M3GP-N: *Min-Max Normalization*

One way to partially counteract the variances' problem and use the Euclidean distance, would be to scale each of the dimensions for their values to fall on the same interval (that should be small). For example, to scale the dimensions to the [0,1] interval. This way, although sometimes not equal, the variables' variances will fall on the [0,1] interval.

Let:

- w_i, or $w_i(\vec{x})$, be a *dimension* variable;
- w_i^{min} be the minimum output value of $w_i(\vec{x})$ when applying it to the training samples;
- w_i^{max} be the maximum output value of $w_i(\vec{x})$ when applying it to the training samples;

where $\vec{x} = (x_1, ..., x_p)$ is the vector of *feature variables* of the dataset.

Then,

$$w_i^{new} = \frac{w_i - w_i^{min}}{w_i^{max} - w_i^{min}},$$

when applied to the training samples will have all values falling on the interval [0,1].

This new version of M3GP is called **M3GP-N**. The individuals in **M3GP-N** are created in the same way as the ones created in M3GP. The difference stands in the trees' evaluation. In M3GP-N the training and test tree outputs are normalized (both test and training data normalized using $w^{min} = (w_1^{min}, ..., w_d^{min})$ and $w^{max} = (w_1^{max}, ..., w_d^{max})$ calculated on training data).

First, the mapped training samples are calculated, applying the dimension function/variable $w : \mathbb{R}^p \longrightarrow \mathbb{R}^d$ to each training sample $\vec{y_i} = (y_1, ..., y_p)$, with $i \in \{1, ..., n\}$ where n is the number of instances of $\mathcal{D}|_T$ (and $\mathcal{D}|_T$ defined as in section 2.3). Then, w^{min} and w^{max} are calculated. Finally, the mapped training samples and the mapped test samples are normalized.

Notice that:

- the tree is never altered, it is not normalized - only the mapped samples are calculated and then normalized;
- the normalization is always performed using w^{min} and w^{max} calculated using the mapped training samples;

Hence, there is an *original space* and a *normalized space*. The cluster centroids are calculated on the normalized space, using the *normalized mapped training samples* and the euclidean distance is considered.

If a dimension variable is a numeric constant or a function of numeric constants, then $w_i^{new} = 0/0$, which is undefined. Instead, w_i^{new} is set to 0. It is important to notice that these constant dimensions do not alter the training accuracy of the individuals, since the Euclidean distance is used. Let w_k be the constant dimension variable. Then,

$$d_E(w) = \sqrt{\sum_{i=1}^{d}(w_i - m_i)^2} = \sqrt{\sum_{i=1, i\neq k}^{d}(w_i - m_i)^2}$$

since $w_k = m_k$, where m_k is the k^{th} entry of a centroid vector.

The only problem of considering constant dimensions is that the number of nodes might be higher than necessary (necessary, since the constant dimensions do not add any "knowledge" to the tree[1]).

4.2 M3GP-S: *Standardization*

Another way to counteract the problem posed by the different variances and use the Euclidean distance, is to standardize the dimension variables. By standardizing the *dimension* variables, each variable will have an average value of 0 and a standard deviation of 1. Hence, all the dimensions will have equal variances, although they might still be correlated.

Let:

- w_i, or $w_i(\vec{x})$, be a *dimension* variable;
- μ_i be the average of output values of w_i, when applying it to the training samples;
- σ_i be the standard deviation of w_i, when applying it to the training samples;

where $\vec{x} = (x_1, ..., x_p)$ is the vector of *feature variables* of the dataset.

Then,

$$w_i^{new} = \frac{w_i - \mu_i}{\sigma_i}$$

will be a dimension variable with average 0 and standard deviation 1.

[1] This topic will be addressed again in chapter 7.

This new version of the algorithm is called **M3GP-S**. The individuals produced by **M3GP-S** are created in the same way as the ones in M3GP, but before evaluating the individual, the mapped training samples and the mapped test samples are standardized. As for M3GP-N, the trees are not changed, only the tree outputs are standardized. Hence, there is an *original space* and a *standardized space*. Variables μ_i and σ_i (with $i \in \{1, ..., d\}$) are calculated using the standardized mapped training samples. The same way, the cluster centroids are calculated using the standardized mapped training samples and then, the euclidean distance is used. As before, the tree is never altered, only the mapped samples are calculated and then standardized. The standardization is always performed using $\mu = (\mu_1, ..., \mu_d)$ and $\sigma = (\sigma_1, ..., \sigma_d)$ calculated using the mapped training samples;

If, for some $i \in \{1, ..., d\}$, $w_i(x)$ is constant, then w_i^{new} would give 0/0. As such, if that happens, w_i^{new} is set to 0.

4.3 Results and comparison to original M3GP

Since M3GP-N and M3GP-S are versions of M3GP in which the euclidean distance is used (instead of the mahalanobis distance), the experimental analysis was performed with a population of 500 individuals. As such, the datasets described in table 3.1 and the running parameters in table 3.2 remained unchanged for this analysis.

	HRT	IM-3	WAV	SEG	IM-10	YST	VOW	M-L
Training accuracy								
M3GP	**95.0**(2.6)	**99.6**(0.4)	**90.8**(0.4)	**98.1**(0.4)	**93.3**(0.4)	**69.1**(1.2)	**100**(0.0)	**100**(0.0)
M3GP-N	90.5(1.4)	98.2(0.9)	86.8(0.7)	95.1(0.9)	83.7(0.9)	59.1(2.2)	75.8(1.9)	77.3(2.7)
M3GP-S	91.5(1.5)	98.7(1.1)	86.6(0.5)	95.0(0.8)	84.5(1.1)	62.5(1.2)	76.5(2.3)	80.3(3.5)
Test accuracy								
M3GP	**79.0**(4.5)	**95.9**(2.5)	84.0(0.9)	**95.7**(0.8)	**91.3**(0.7)	**56.6**(2.0)	**95.5**(1.9)	**62.5**(7.4)
M3GP-N	**81.5**(3.9)	**93.8**(2.8)	**85.0**(1.0)	93.3(1.1)	82.8(1.1)	**54.7**(2.9)	67.3(2.9)	**61.6**(5.1)
M3GP-S	**81.5**(3.8)	**92.8**(2.5)	**84.6**(1.0)	93.4(1.4)	83.6(1.2)	**57.3**(2.3)	66.8(4.3)	**63.0**(3.9)
# of nodes								
M3GP	318(45-422)	**66**(35-701)	132(39-213)	367(62-267)	777(261-1283)	**317**(203-888)	**42**(42-72)	**11**(11-209)
M3GP-N	**97**(27-258)	167(44-482)	209(32-209)	**169**(56-388)	685(290-1224)	667(206-1895)	76(48-457)	118(50-483)
M3GP-S	105(29-225)	158(89-325)	**43**(32-240)	216(76-506)	**629**(95-1033)	1019(272-1303)	184(60-459)	689(87-689)
# of dimensions								
M3GP	13(3-26)	5(3-12)	34.5(25-38)	11.5(5-17)	17(10-21)	14(11-22)	23(20-24)	11(8-13)
M3GP-N	7.5(3-14)	5(2-10)	26(18-43)	12(7-17)	11(6-16)	15(9-24)	18(13-25)	13(8-19)
M3GP-S	8.5(3-18)	4(1-11)	25(18-33)	13(6-22)	9(5-15)	14(8-22)	18(11-24)	14(10-24)

Table 4.1: Comparison between the M3GP, M3GP-N and M3GP-S.

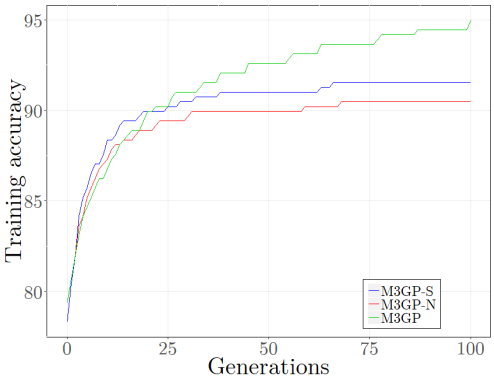
The results of the comparison between M3GP, M3GP-N and M3GP-S are reported in table 4.1. Regarding training accuracy, M3GP is significantly better than the remaining two versions. However, when it comes to test accuracy, the difference is not statistically significant, except for SEG, IM-10, WAV and VOW problems. For SEG, IM-10 and VOW, M3GP performs better than the remaining two. On WAV, M3GP-N and M3GP-S perform better than M3GP.

M3GP-S gets higher median test accuracy values for HRT, WAV, YST and M-L datasets when compared to M3GP (and a lower standard deviation on the first and last). M3GP-N gets higher test accuracy values for HRT and WAV datasets when compared to M3GP (and a smaller standard deviation on the first one). The individuals created by M3GP-N tend to have a smaller number of nodes (except on YST, VOW and M-L), although the best individual tends to have more nodes, when comparing to M3GP.

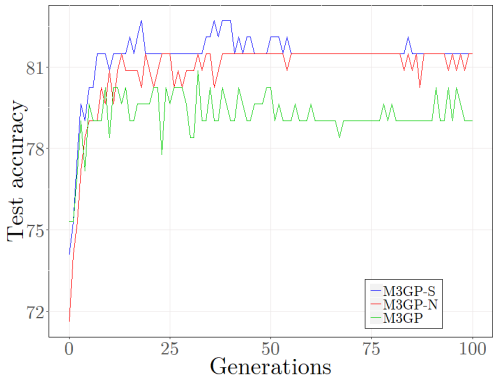
Figures 4.2 and 4.3 add more detail about the evolution of training and test accuracies as the generation number increases. Figures 4.4 and 4.5 add detail on last generation individuals' training and test acurracies.

Figures 4.2a,c,e,g, 4.3a,c,e,g, 4.4a,c,e,g, and 4.5a,c,e,g show that, from generation to generation and on the last generation, M3GP is able to produce much higher values of training accuracy.

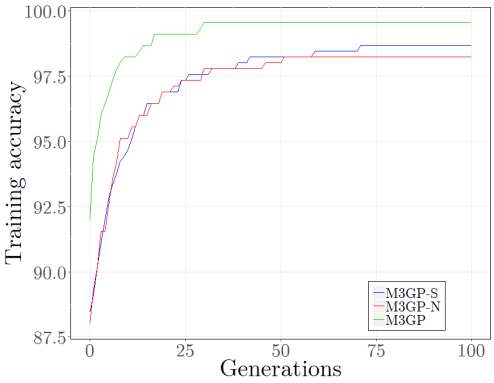
On HRT, figure 4.2b shows us that the median test accuracies of M3GP-S and M3GP-N stabilize after generation 50 and are always above the median test accuracy of M3GP. Figure 4.4b shows that, on HRT, M3GP produces individuals with higher test accuracy on the last generation, but also with a lower test accuracy, when comparing to the two remaining versions. The range of test accuracy values is smaller for M3GP-N and M3GP-S. These two do not produce individuals with higher test accuracy than M3GP, but the minimum test accuracy is not as low as for M3GP.



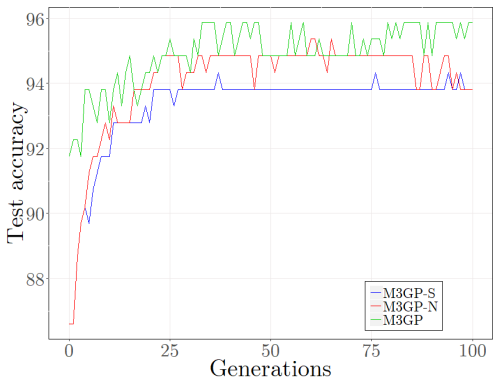
(a) HRT - Training.



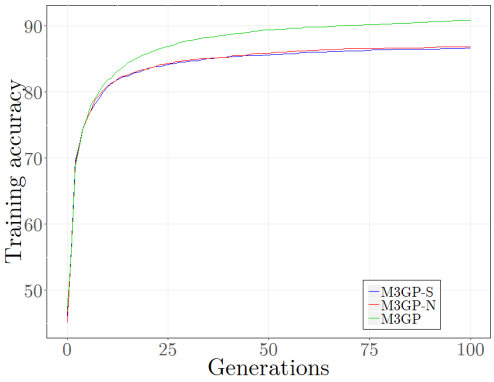
(b) HRT - Test.



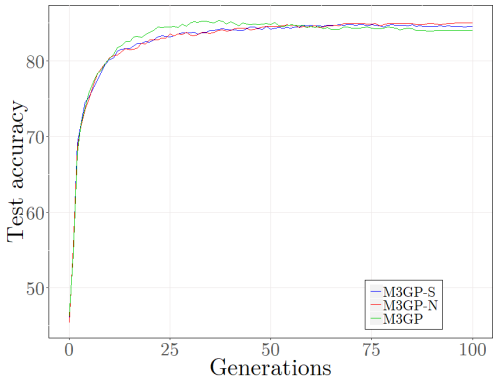
(c) IM-3 - Training.



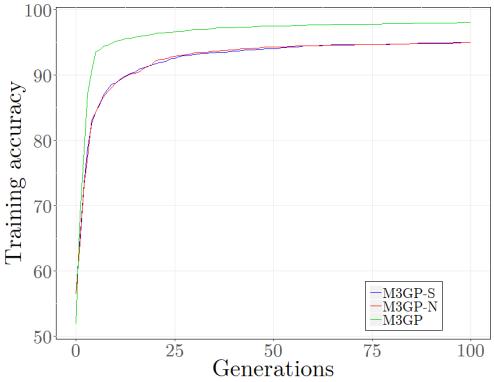
(d) IM-3 - Test.



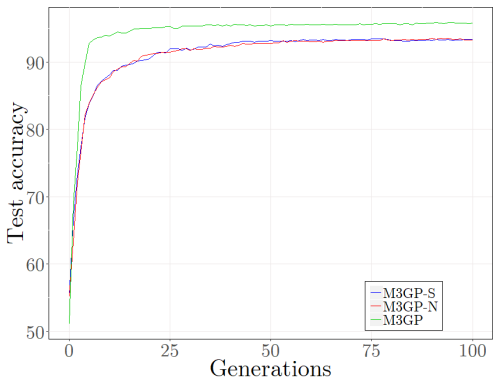
(e) WAV - Training.



(f) WAV - Test.

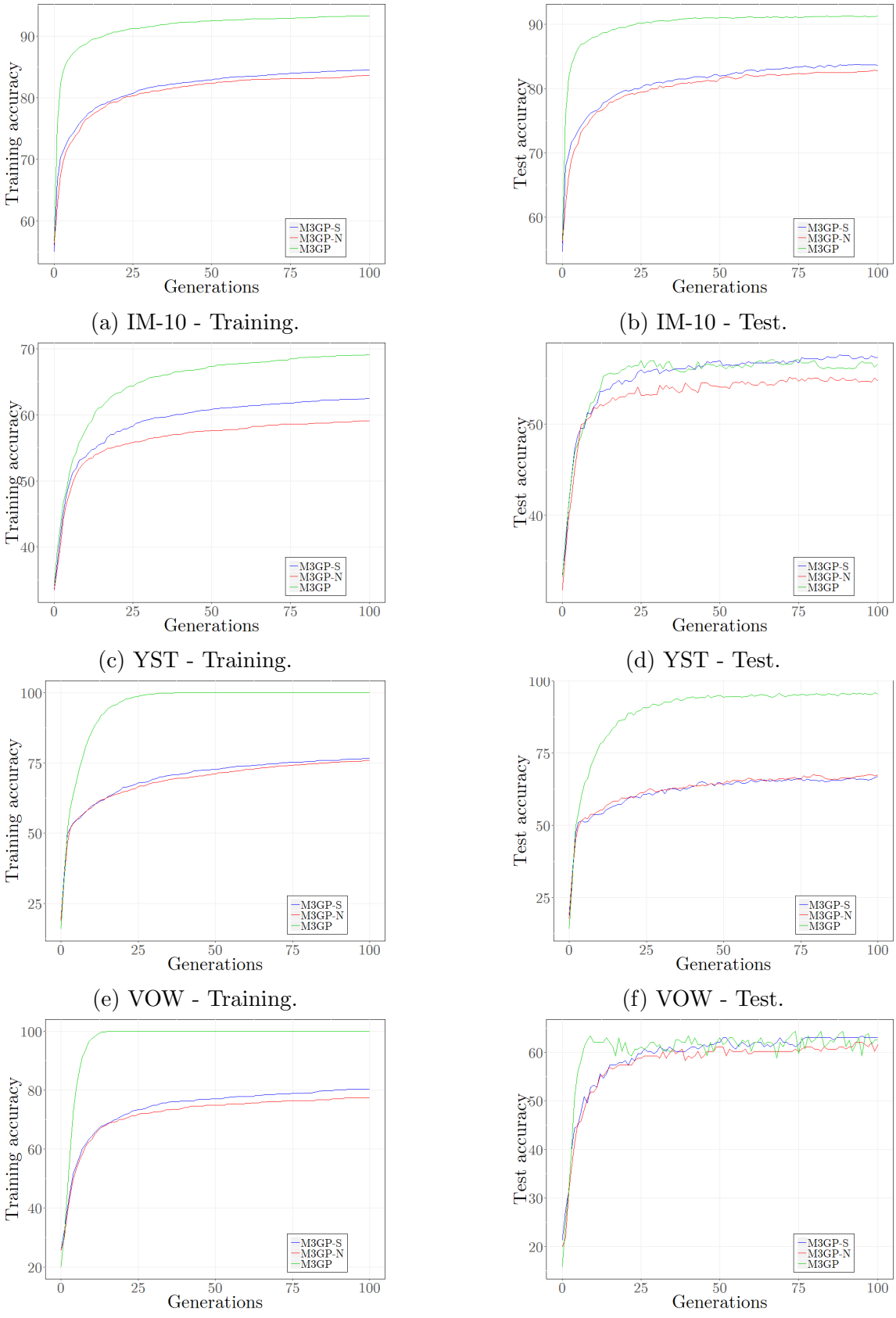


(g) SEG - Training.



(h) SEG - Test.

Figure 4.2: Comparison between median accuracies of M3GP, M3GP-N and M3GP-S algorithms on HRT, IM-3, WAV and SEG datasets.



(a) IM-10 - Training.

(b) IM-10 - Test.

(c) YST - Training.

(d) YST - Test.

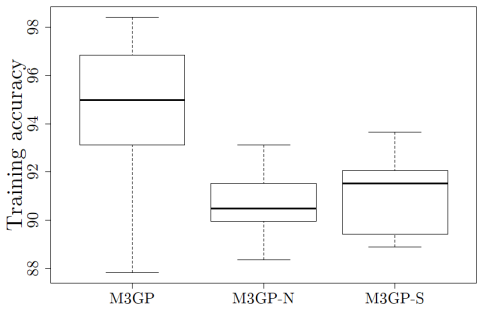
(e) VOW - Training.

(f) VOW - Test.

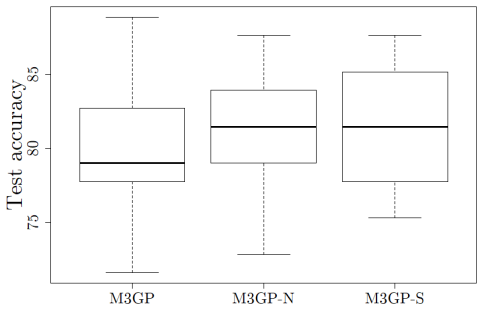
(g) M-L - Training.

(h) M-L - Test.

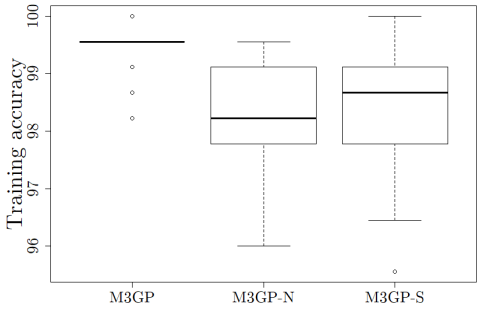
Figure 4.3: Comparison between median accuracies of M3GP, M3GP-N and M3GP-S algorithms on IM-10, YST, VOW and M-L datasets.



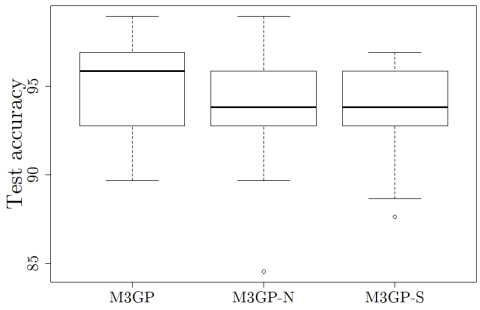
(a) HRT - Training.



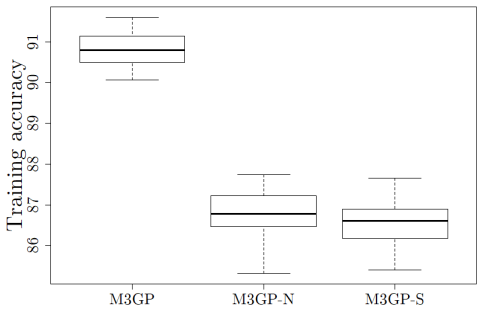
(b) HRT - Test.



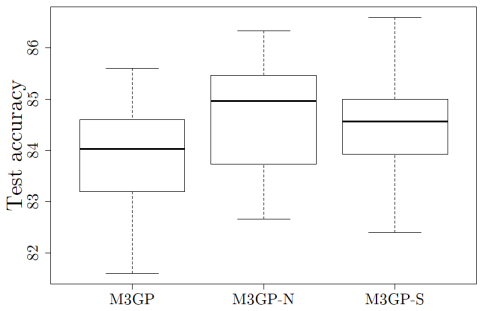
(c) IM-3 - Training.



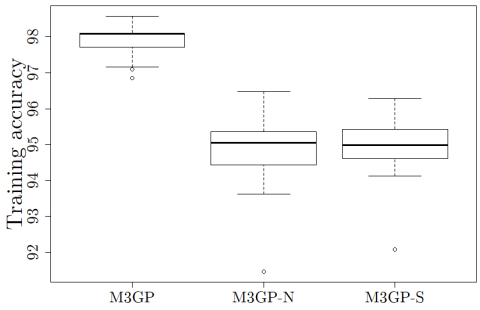
(d) IM-3 - Test.



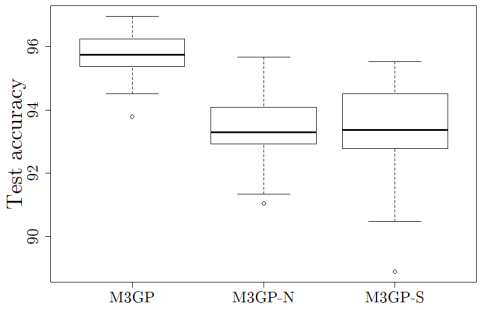
(e) WAV - Training.



(f) WAV - Test.

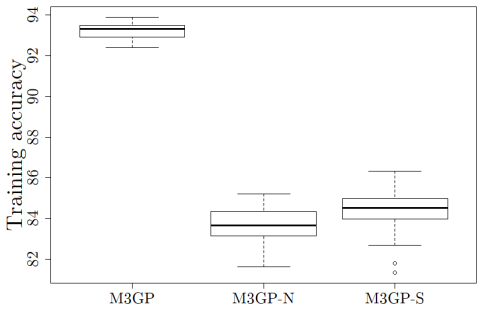


(g) SEG - Training.

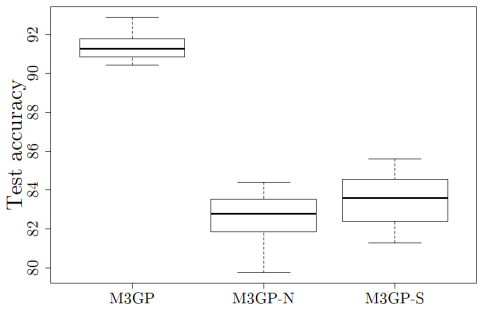


(h) SEG - Test.

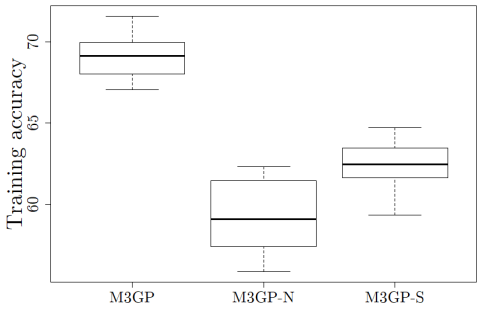
Figure 4.4: Comparison between last generation individuals' accuracies of M3GP, M3GP-N and M3GP-S algorithms on Heart, IM-3, WAV and SEG datasets.



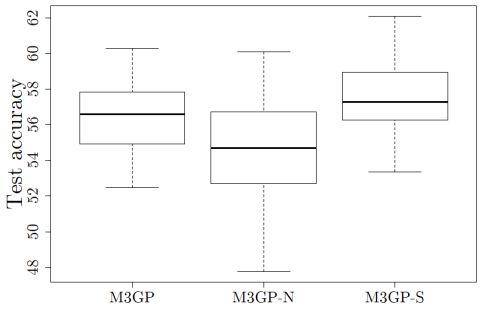
(a) IM-10 - Training.



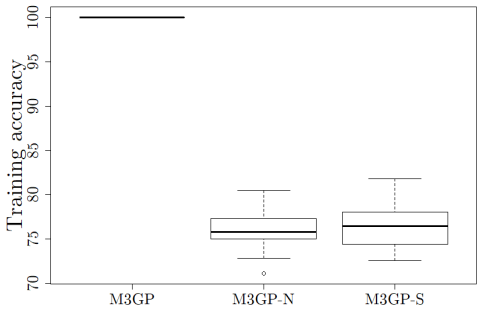
(b) IM-10 - Test.



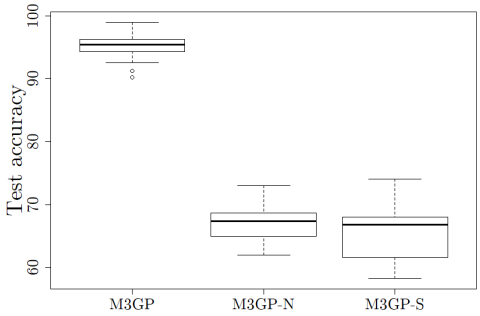
(c) YST - Training.



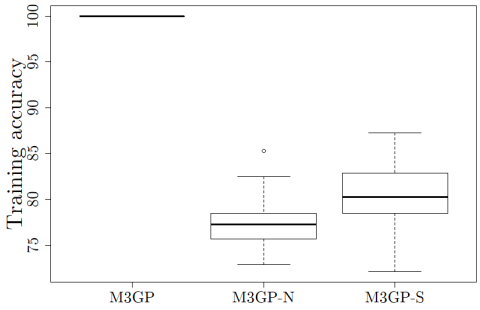
(d) YST - Test.



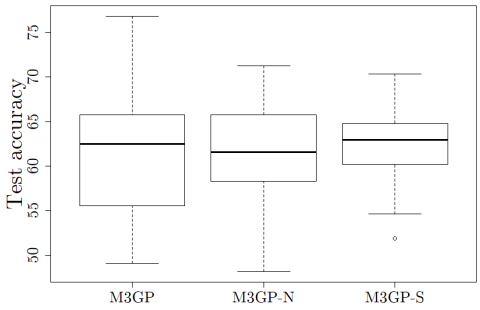
(e) VOW - Training.



(f) VOW - Test.



(g) M-L- Training.



(h) M-L - Test.

Figure 4.5: Comparison between last generation individuals' accuracies of M3GP, M3GP-N and M3GP-S algorithms on IM-10, YST, VOW and M-L datasets.

Figure 4.2d shows that M3GP outperforms M3GP-N and M3GP-S on test data. On the last generation, the range of training accuracy values of M3GP-N and M3GP-S is slightly bigger when compared to M3GP, although all versions achieve very high values of accuracy, as it can be seen in figure 4.4c. With figure 4.4d one can see that, although the range of test accuracy values for M3GP-N is the the same as for M3GP (with the exception of one outlier), the median is higher for M3GP and this difference is significant.

On WAV, figure 4.2f shows that the median test accuracy of M3GP-S and M3GP-N is higher than the median test accuracy of M3GP since approximately generation 68. Figure 4.4f shows that, on the last generation, the smallest test accuracy achieved for M3GP-N and M3GP-S is higher then the smallest test accuracy achieved for M3GP. Also, M3GP-N and M3GP-S are able to produce individuals with higher test accuracy than M3GP.

M3GP-N and M3GP-S produce a very similar behaviour on SEG, IM-10 and VOW as it can be seen in figures 4.2h, 4.3b, 4.3f - for these datasets, M3GP-N and M3GP-S do not produce competitive results of test accuracy. This can also be seen on the last generation, in figures 4.4h, 4.5b and 4.4h.

On YST test data, M3GP and M3GP-S behave similarly throughout the generations. From around generation 75, the median test accuracy for M3GP-S is always higher than the M3GP's median test accuracy (see figure 4.3d). Figure 4.5d shows that, although the minimum value of test accuracy achieved by M3GP and M3GP-S is roughly the same, M3GP-S achieves higher median and maximum values.

Chapter 5

Geometric semantic inspired mutation for M3GP

In this chapter, a multiclass classification algorithm is introduced. It can be seen as a new version of **M3GP**. The new algorithm is called **Geometric Semantic Inspired M3GP**, in short **GSI-M3GP**.

GSI-M3GP is, in everything, similar to the **M3GP** algorithm, except for the operators used to create new individuals. In **GSI-M3GP**, three operators are considered:

- *add a new branch* to the tree;
- *remove an existing branch* from the tree;
- *geometric semantic inspired mutation* (from now on, **gsi-mutation**).

The first two already existed in M3GP, while the third is new and specific to **GSI-M3GP**.

This chapter presents the new operator and the different variants that were tested. Nine different variants were considered: **Baseline**, **Hyperrectangle**, **Hyperellipse** (with 2 sub-variants), **Misclassified**, **Misclassified Hyperrectangle**, **Misclassified Hyperellipse**, **Donut**,

Around Misclassified (with 2 sub-verions) and **Correct-Misclassified distances** (with 2 sub-variants).

5.1 The new mutation operator

The mutation operator introduced in this section is called **geometric semantic inspired mutation** due to the inspiration coming from one of **GSGP**'s genetic operators: **Geometric Semantic Crossover** and **Geometric Semantic Mutation**[14], explained in sub-section 2.2.2.

As already mentioned, the geometric semantic crossover operator generates an offspring individual whose semantics are geometrically between the semantics of the parents. The new operator here presented is not referred to as *geometric semantic*, but as *geometric semantic inspired* since it is not directly acting on the individuals' semantics. The offspring created using **gsi-mutation** is based on an alteration made on the *mapped training space* (see sub-section 2.3.1), and this will have an impact on the semantic space. As the *geometric semantic mutation* operator, **gsi-mutation** is a unary operator, only needing one *parent individual* to be performed.

Remembering the clustering process of **M3GP**[15] and **M2GP**[8], the *mapped training samples* are separated and grouped according to the class to which the training samples belong. From these groups/sets, covariance matrices and centroid vectors are created. Finally, each *mapped training sample* is associated to its closest centroid (considering the Mahalanobis distance). Thus, each instance is classified as belonging to a specific class. However, this classification might be incorrect. Then, there are correctly classified and misclassified instances.

What if one could change a misclassified point in space, so that, that point could be correctly classified, but without moving the remaining points? The idea of **gsi-mutation** is to find a

way of changing the position of a misclassified instance, so that, that instance is closer to the correct centroid than to the other centroids (and so it gets correctly classified), while the other instances are not moved from their original place.

Let us consider an example: a case where the number of classes is equal to 3 and a 2-dimensional individual. Each mapped training sample is 2-dimensional, so it can be drawn on a 2-dimensional space, as in figure 5.1. The small points in pink, green and orange are correctly classified mapped training instances. The bigger points, C_1, C_2 and C_3 represent the cluster centroids for each class. The pink, green and orange ellipses contain the points that are being assigned with the centroid that has the same color as the ellipse - they are only informative. Finally, points P_1, P_2, P_3 and P_4 are misclassified mapped training samples, because they are closer to a different centroid than the one representing the class they belong to.

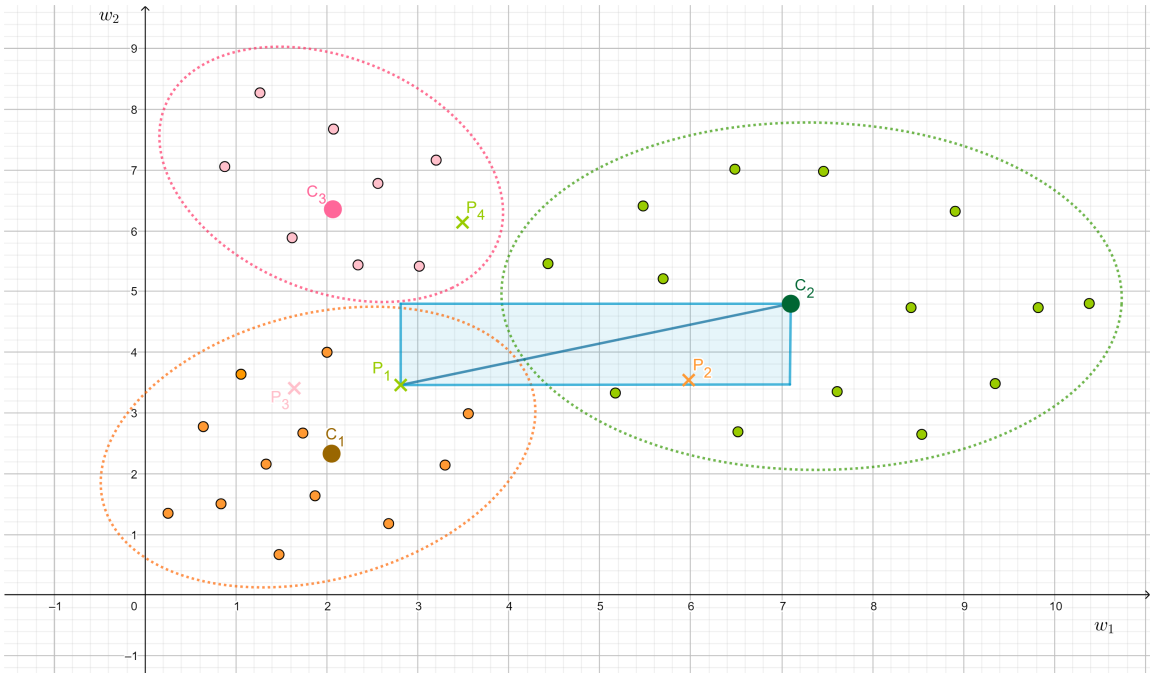


Figure 5.1: Toy example of the functioning of **gsi-mutation** applied to a 2-dimensional individual, and a dataset with 3 classes and 38 training samples.

Let us think that, for instance, P_1 is randomly chosen to move closer to a randomly chosen centroid, for example, C_2. We want P_1 to move somewhere along the blue line segment closer to C_2, while the other points remain in the same place. If an operator with these characteristics can be defined, there is a possibility that P_1 can get correctly classified.

Definition 5.1 ***Informally: gsi-mutation operator***

For simplicity, let $w(x) = (w_1(x), w_2(x), ..., w_d(x))$ *be a point in the dimension variables space, for a given p-dimensional point* x*. Let us suppose that:*

- $P_i = (p_{i1}, ..., p_{id})$ *is the* i^{th} *misclassified point chosen at random;*
- $C_j = (c_{j1}, ..., c_{jd})$ *is the* j^{th} *cluster centroid (where* $j \in \{1, ..., \#Centroids\}$*), also chosen at random.*

According to the previous description, we want to find a function h *such that, for each dimension* k*:*

$$h(w_k) = b(P_{ik}, C_{jk}) \times a(w_k) + w_k \times (1 - a(w_k)) \tag{5.1}$$

where:

- $b(P_{ik}, C_{jk})$ *informally represents a value between* P_{ik} *and* C_{jk} *- so in the end, in two dimensions, the blue segment is not a segment, but a rectangle (as the blue rectangle in figure 5.1);*
- $a(w_k) = \begin{cases} 1 & w = P_i \\ 0 & w \neq P_i \end{cases}$

This way,

- if $w = P_i$, then $h(w_k) = b(P_{ik}, C_{jk}) \times 1 + w_k(1 - 1) = b(P_{ik}, C_{jk})$
- if $w \neq P_i$, then $h(w_k) = b(P_{ik}, C_{jk}) \times 0 + w_k(1 - 0) = w_k$

which means that: if w is the misclassified instance P_i, that point is moving closer to C_j; if w is another point in space, it remains where it is.

Notice that, $h(w_k) = b(P_{ik}, C_{jk}) \times a(w_k) + w_k \times (1 - a(w_k)) = b(w_k, C_{jk}) \times a(w_k) + w_k \times (1 - a(w_k))$, because $a(w_k) = 1$ if $w = P_i$ and $a(w_k) = 0$ if $w \neq P_i$.

If functions with the characteristics of b and a exist, then **gsi-mutation** operator can be defined.

However, a branched function like a cannot be used, since it is not continuous. A continuous function with similar behaviour is used instead, as an approximation of a - a *Gaussian function*, as it is explained below.

Notice that h can be applied to every point in space. The idea of **gsi-mutation** is that, when applying h to the points in space only a selected point changes, while the remaining ones stay where they are. From now on, we call this selected point the *moving point*.

Definition 5.2 ***gsi-mutation operator***

Let P be the parent individual's tree, defined by $f : \mathbb{R}^p \to \mathbb{R}^d, f(x) = (f_1(x), ..., f_d(x))$. The new individual, I, is characterized by $g : \mathbb{R}^p \to \mathbb{R}^d, g(x, y) = (g_1(x, y), ..., g_d(x, y))$, where the i^{th} dimension of g, with $i = \{1, .., d\}$, is defined by:

$$g_i(x, y) = betw(f_i(x), c_i) \times peak(f_i(x), y_i) + f_i(x) \times (1 - peak(f_i(x), y_i)) \tag{5.2}$$

where:

- $betw(f_i(x), c_i) = r_i f_i(x) + (1 - r_i)c_i$, *where* $r_i \in [0, 1]$
- c_i *is the* i^{th} *coordinate of a random centroid,* $C = (c_1, ..., c_d)$, *considering the same centroid for each i;*
- $peak(f_i(x), y_i) = e^{-\frac{(f_i(x)-y_i)^2}{\sigma_i^2}}$
- y_i *is the* i^{th} *coordinate of the moving point: a point in the space of mapped training samples - typically a point representing a misclassified instance,* $y = (y_1, ..., y_d)$;
- σ_i *is a parameter and should be a low value, but not zero.*

When $f(x) = y$:

- $peak(f_i(x), y_i) = peak(y_i, y_i) = e^0 = 1$
- $g_i(x, y) = betw(f_i(x), c_i) \times 1 + f_i(x)(1 - 1) = betw(f_i(x), c_i)$

and so $g(x, y) = \big(betw(f_1(x), c_1), ..., betw(f_d(x), c_d)\big)$ is "between" $f(x)$ and the centroid C.

For values of $f(x) \neq y$:

- $peak(f_i(x), y_i) = \delta_i$, where $\delta_i \leq 1$, and for most of the values of $f_i(x)$, $\delta_i << 1$;
- $g_i(x, y) = betw(f_i(x), c_i) \times \delta_i + f_i(x)(1 - \delta_i) = \delta_i[betw(f_i(x), c_i) - f_i(x)] + f_i(x) = \eta_i + f_i(x)$, with $\eta_i = \delta_i[betw(f_i(x), c_i) - f_i(x)]$.

and so $g(x, y) = (f_1(x) + \eta_1, ..., f_d(x) + \eta_d) = f(x) + \eta$ is the original function plus a perturbation, with $\eta = (\eta_1, ..., \eta_d)$.

Notice that, $\delta_i \leq 1$ instead of $\delta_i < 1$, because of the way that the peak function is defined. Even if $f(x) \neq y$, there is a possibility that $peak(f_i(x), y_i) = 1$, if for some i we have $f_i(x) = y_i$.

The maximum value of the *peak function* is one. And although it never reaches zero, if σ_i is set to a very low value, then for most of the values $f_i(x) \neq y_i$, the $peak(f_i(x), y_i)$ is very low (see figure 5.2).

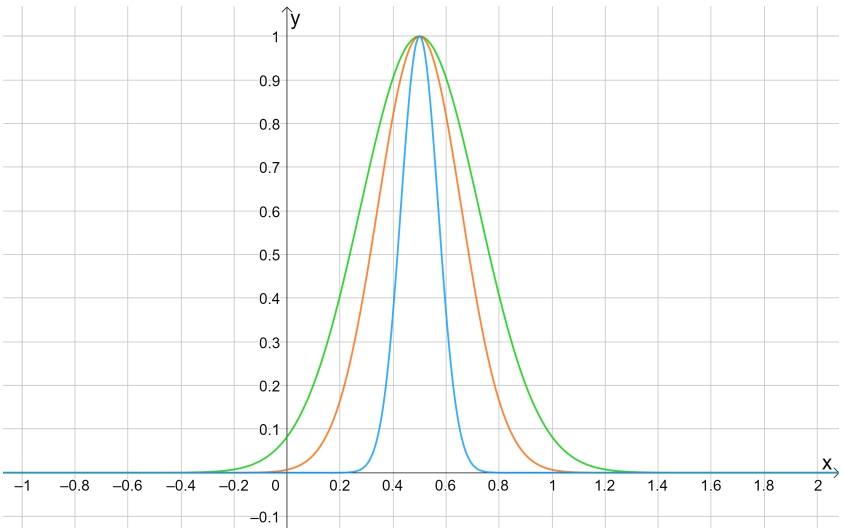
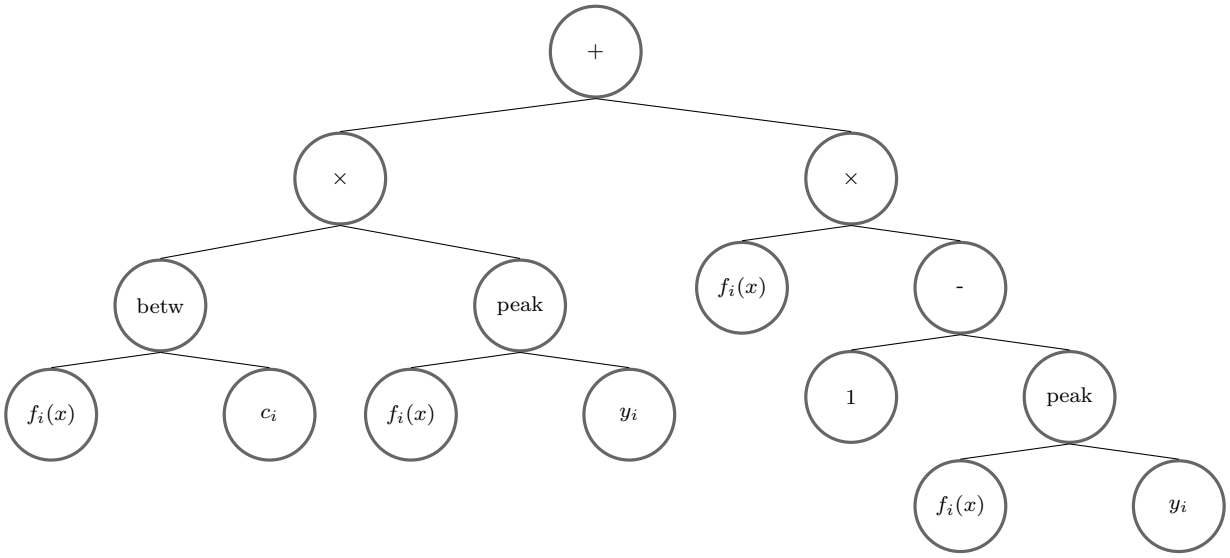


Figure 5.2: Peak functions with different values of σ_i^2: the green has σ_i^2=0.1, the orange has σ_i^2=0.05 and the blue has σ_i^2=0.01.

As represented in figure 5.2, the lower the value of σ_i is, the more accurately will the peak function approximate the branched function a.

In tree format, each branch of the offspring created applying the **gsi-mutation** operator, is represented by:

$g_i(x) =$

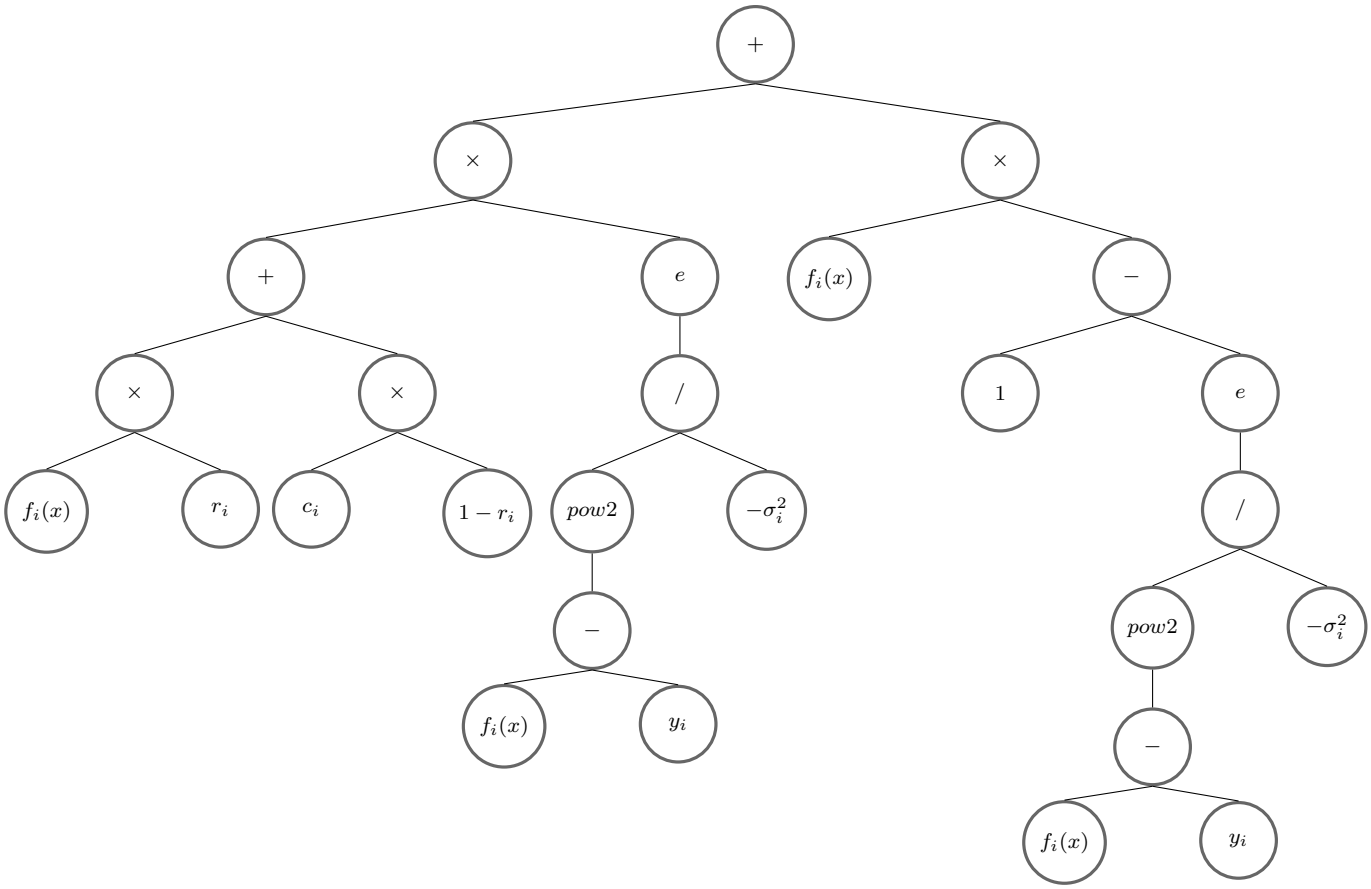


Or, using only:

- basic operations: sum (+), subtraction (−), multiplication (×), division (/) - all with arity[1] 2, since 2 arguments are needed;
- e: Euler's number, e, to the power of the argument, with arity 1;
- $pow2$: the argument to the power of 2, with arity 1;

then, each dimension of the offspring individual can be represented by:

$g_i(x) =$



The depth of dimension $g_i(x)$ is $7 + depth(f_i(x))$, where $depth(f_i(x))$ is the depth of the corresponding dimension of the parent individual's tree. The number of nodes of the offspring's

[1] *Arity* is the number of arguments that an operation/function takes.

i^{th} dimension is equal to $23 + 4nodes(f_i(x))$, where $nodes(f_i(x))$ is the number of nodes of the parent individual's tree i^{th} dimension.

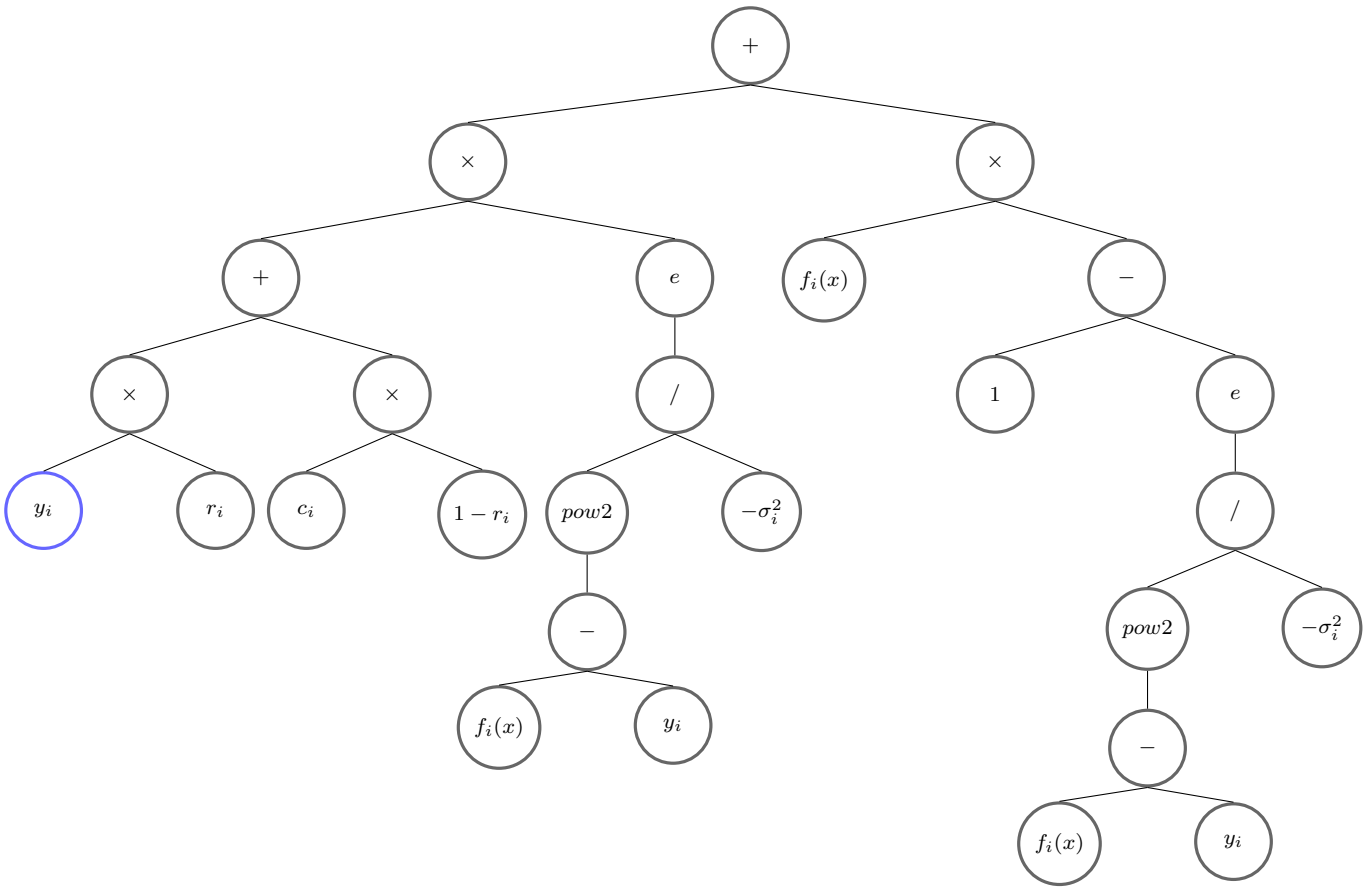
What if, instead of considering $betw(f_i(x), c_i)$, we considered $betw(y_i, c_i)$? In other words, what if, instead of considering a value between the i^{th} dimension of the parent tree and the i^{th} entry of the centroid vector (see function 5.2), one considers a value between the i^{th} entry of the moving point and the i^{th} entry of the centroid vector, that is, replacing the parent tree by the moving point? Is it the same?

According to the original idea used to create function 5.2, it is the same. However, function 5.1 uses function a in its definition, which only takes two values: 0 or 1. Since, in the formal definition of **gsi-mutation operator** (see function 5.2), a is replaced by the peak function (a continuous function with output values in]0,1]), having $betw(f_i(x), c_i)$ is not exactly the same as having $betw(y_i, c_i)$.

However, it is not that different. When $f_i(x)$ is very far from y_i, $peak(f_i(x), y_i)$ takes very low values which are very close to 0, and so, multiplying $peak(f_i(x), y_i)$ by $betw(f_i(x), c_i)$ or multiplying $peak(f_i(x), y_i)$ by $betw(y_i, c_i)$ is, informally, "almost the same". When $f_i(x)$ is close to y_i, then considering a value between y_i and c_i will be almost the same as considering a value between $f_i(x)$ and c_i.

By replacing $f_i(x)$ with y_i in $betw(f_i(x), c_i)$, we have:

$g_i(x) =$



Because function 5.2 uses the peak function instead of the branched function a, then having $betw(f_i(x), c_i)$ or $betw(y_i, c_i)$ does not make a difference, since the output will be a small perturbation of what was desired from **gsi-mutation** on both variants. However, there are advantages of using $betw(y_i, c_i)$ instead of $betw(f_i(x), c_i)$. The most important one is that the number of nodes of the i^{th} dimension of the offspring individual is $24 + 3nodes(f_i(x))$, instead of $23 + 4nodes(f_i(x))$. This simple change makes a difference, when applying the operator many times, as expressed in table 5.1.

Table 5.1 shows the comparison between the number of nodes when applying the **gsi-mutation** operator n times to a 1-one dimensional tree using $betw(y_i, c_i)$ and $betw(f_i(x), c_i)$ (in function 5.2). It is clear that having $betw(y_i, c_i)$ instead of $betw(f_i(x), c_i)$ is an advantage.

n	1	2	3	4	5	6	7	...	30
$betw(y_i, c_i)$	405	1239	3741	11247	33765	101319	303981	...	2.86×10^{16}
$betw(f_i(x), c_i)$	531	2147	8611	34467	137891	551587	2206371	...	1.55×10^{20}

Table 5.1: Comparison between the number of nodes of a 1-dimensional individual to which the mutation operator has been applied n times using $betw(y_i, c_i)$ and $betw(f_i(x), c_i)$ (considering individuals created with 6-depth *full method*[11]).

Since the depth of each dimension is not altered by replacing $betw(f_i(x), c_i)$ with $betw(y_i, c_i)$ in function 5.2, then, one gets:

- $depth(I) = max\{depth(g_1(x)), ..., depth(g_d(x))\} = max\{7+depth(f_1(x)), ..., 7+depth(f_d(x))\}$
  $= 7 + max\{depth(f_1(x)), ..., depth(f_d(x))\}$
- $nodes(I) = \sum_{i=1}^{d} nodes(g_i(x)) = \sum_{i=1}^{d} [24+3 \times nodes(f_i(x))] = 24d+3\sum_{i=1}^{d} nodes(f_i(x))$

where I is the offspring individual.

5.2 GSI-M3GP variants: *how to choose the moving point?*

The first question that arises when implementing the **gsi-mutation** operator is *how to choose the moving point*? It should be a point that represents a misclassified mapped training instance, but it should not be "far away" from where the mapped test instances might fall, so that, not only the training accuracy is maximized, but the algorithm has generalization ability.

A second, and also very important question, is *how to choose the value of* σ_i*?* - which represents the width of the peak function's "bell curve". To answer this second question, in every variant of **GSI-M3GP**, σ_i is defined as:

$$\sigma_i = pred\% \times ({w_i}^{\max} - {w_i}^{\min}),$$

where:

- *pred%* is a *predefined percentage* and it should be a small value, different from 0;
- ${w_i}^{\max}$ and ${w_i}^{\min}$ are, respectively, the maximum and minimum output values of $w_i(x)$, the i^{th} dimension of an individual's tree, when applying it to the training samples;

The way σ_i is defined reflects the fact that the variables w_i have different variances and different ranges. As an example, let us consider a tree with two dimensions. Considering the mapped training samples as outputs of $w = (w_1(x), w_2(x))$, it could happen that $w_1(x) \in [-1, 1]$ and $w_2(x) \in [3000, 100000]$. It would not make sense to set σ_i to be the same value for all the dimension variables. Instead, σ_i is defined as a percentage of the range of the outputs of variable w_i, since the effect would be different on variables with such different range of values.

If ${w_i}^{\max} = {w_i}^{\min}$, *i.e.* if dimension w_i is constant, then $\sigma_i = pred\%$, to avoid the division by 0 in the peak function (which would give NaN values when $f_i(x) = y_i$).

Different values of *pred*% are tested in subsections 5.2.1 and 5.2.2.

Regarding the way the *moving point* is chosen, it depends on the variant, as explained in the following subsections.

5.2.1 Baseline

The baseline variant is a very simple variant of **GSI-M3GP** algorithm. We choose the *moving point*, $y = (y_1, ..., y_d)$, to be a randomly chosen training instance.

The way that the *moving point* is being chosen is very simple, and even before running **GSI-M3GP** as described, it is easy to see that it is going to overfit - as it will work very well for training data, but it might not work so well for test data if the mapped test samples are far away from the mapped training samples.

5.2.2 Hyperrectangle

Here the *moving point* is chosen in a different way, as follows:

$$y_i \in [{w_i}^{\min}, {w_i}^{\max}], \text{ with } i \in \{1, ..., d\},$$

with ${w_i}^{\min}$ and ${w_i}^{\max}$ as defined previously.

Each entry of the *moving point* is a randomly generated value between the mininum and maximum values for that dimension (outputs considered on the training data). As such, the *moving point* is inside the hyperrectangle that contains all of the mapped training samples. It is an *hyperrectangle* since the number of dimensions d might be higher than 2. If:

- $d = 1$: it is a line;

- $d = 2$: it is a rectangle;
- $d > 2$: it is an hyperrectangle.

Considering again the toy example of section 5.1 (see figure 5.1), the *moving point* $y = (y_1, y_2, ..., y_d)$ is randomly generated with uniform distribution inside the smallest rectangle that contains all mapped training samples, as it is exemplified in the following figure.

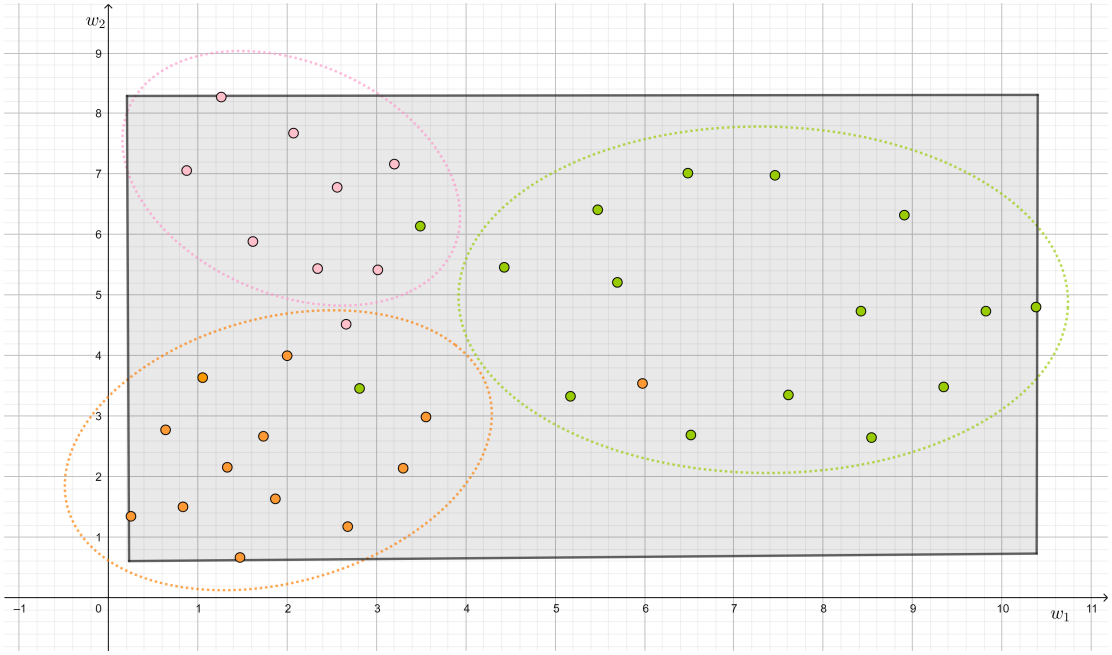


Figure 5.3: The same toy example as in figure 5.1, of a 2-dimensional individual and a dataset of 3 classes. The *moving point* is a randomly chosen point from inside the grey rectangle.

5.2.3 Hyperellipse

Equation

$$(w - \mu)^T S^{-1} (w - \mu) = a^2$$

can be interpreted as a^2 being the *squared Mahalanobis distance* between d-dimensional point w and the distribution defined by the d-dimensional centroid point μ and the $d \times d$ covariance matrix S. It can also be interpreted as the equation of the hyperellipse for which the axes are defined by S, it is centered in μ and such that the Mahalanobis distance between a point, w, belonging to the hyperellipse and μ is a.

If each variable w_i follows a normal distribution, and so if w follows a multivariate normal distribution, then:

$$P\{(w-\mu)^T S^{-1}(w-\mu) \leq \chi^2_{d,\alpha}\} = 1-\alpha$$

and $(w-\mu)^T S^{-1}(w-\mu) = \chi^2_{d,\alpha}$ is the $(1-\alpha)\times 100\%$ prediction hyperellipse for a multivariate normal random vector of variables w, with average point μ and covariance matrix S[9]. Variable d corresponds to the size of w and μ, and is also the order of the square matrix S. $\chi^2_{d,\alpha}$ is the critical value of the chi-square distribution with d degrees of freedom and α is the significance level.

The $(1-\alpha)\times 100\%$ prediction hyperellipse is the region that contains $(1-\alpha)\times 100\%$ of a new sample that is drawn from the original multivariate normal distribution.

In this variant, the *moving point* is drawn from inside an hyperellipse centered in one of centroids (calculated using the clustered mapped training samples) and with axes defined by the sample covariance matrix (also calculated using the clustered mapped training samples) and with distance between the centroid and the hyperellipse points equal to $\sqrt{\chi^2_{d,\alpha}}$.

As such, the *moving point* will be a randomly generated point inside the $(1-\alpha)\times 100\%$ prediction hyperellipse. Assuming that:

- the mapped test samples follow the same distribution as the mapped training samples;
- the mapped training samples follow a multivariate normal distribution;

then $(1-\alpha)\times 100\%$ of them will fall inside the prediction hyperellipse. If a lot of mapped samples (both training and test) fall outside the hyperellipses, it is an indication that the dimension variables might not follow a multivariate normal distribution.

The considered values of α are 0.01, 0.5 and 0.75 and so, respectively, considering 99%, 50% and 25% prediction hyperellipses. These 3 values were considered because, while a 99%

prediciton hypperellipse might contain 99% of the samples, the space to generate the *moving point* might be too wide only to contain all the samples, but almost all of the samples might not be far away from the centroid - there will be a lot of "void" space, and the *moving point* might fall on that empty space. As such, maybe a higher value of α (like 0.5 and 0.75) might be a better compromise between the existing "void space" and the number of mapped instances inside the hyperellipse.

The half-lengths of an hyperellipse, l_j,with $j \in \{1, ..., d\}$, are calculated using the eigenvalues of the covariance matrix: $l_j = \sqrt{\lambda_j \chi^2_{d,\alpha}}$, where λ_j is the j-th eigenvalue. Their directions, $\vec{e_j}$, with $j \in \{1, ..., d\}$, are the eigenvectors of the covariance matrix.

The first approach to generate a point inside an hyperellipse was to generate a point inside the hyperrectangle that contains the hyperellipse, and such that: the hyperrectangle diagonals' intersection is the clusters centroid; the hyperrectangle sides' direction are aligned with the hyperellipses axes' direction (see an example in figure 5.4). Then, if that point is not inside the hyperellipse, a new one is generated.

Figure 5.4 shows a 2-dimensional hyperellipse and its corresponding 2-dimensional hyperrectangle. Notice that this hyperellipse might not be aligned with the axes of the space.

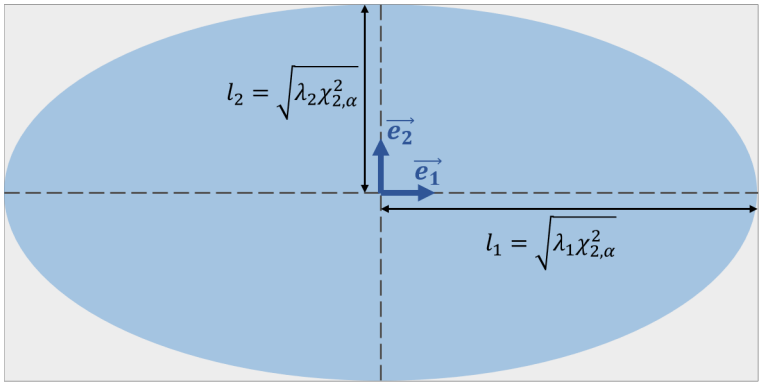


Figure 5.4: A 2-dimensional hyperellipse and the 2-dimensional hyperrectangle with sides alligned with the hyperellipse axes and diagonals' intersection equal to the hyperellipse centroid.

However, as the number of dimension variables increases, the probability of generating a point inside the hyperrectangle that is also inside the hyperellipse decreases[20](as expressed in table 5.2 and figure 5.5).

d	*Volume HR*	*Volume HE*	*Volume HE / Volume HR*
1	$2a_1$	$2a_1$	1
2	$4a_1a_2$	πa_1a_2	0.79
3	$8a_1a_2a_3$	$4\pi a_1a_2a_3/3$	0.52
4	$16a_1a_2a_3a_4$	$\pi^2a_1a_2a_3a_4/2$	0.31
5	$32a_1a_2a_3a_4a_5$	$\frac{8}{15}\pi^2a_1a_2a_3a_4a_5$	0.16
⋮	⋮	⋮	⋮
n	$2^n\prod_{i=1}^{n}a_i$	$\frac{2}{n}\frac{\pi^{n/2}}{\Gamma(n/2)}\prod_{i=1}^{n}a_i$	$\frac{2^{1-n}}{n}\frac{\pi^{n/2}}{\Gamma(n/2)}$

Table 5.2: Probability of finding a point inside the hyperellipse, given that the point was generated inside the hyperrectangle, according to the number of dimensions, *d*. The values of a_i represent the sizes of the hyperellipses axes' half-lengths.

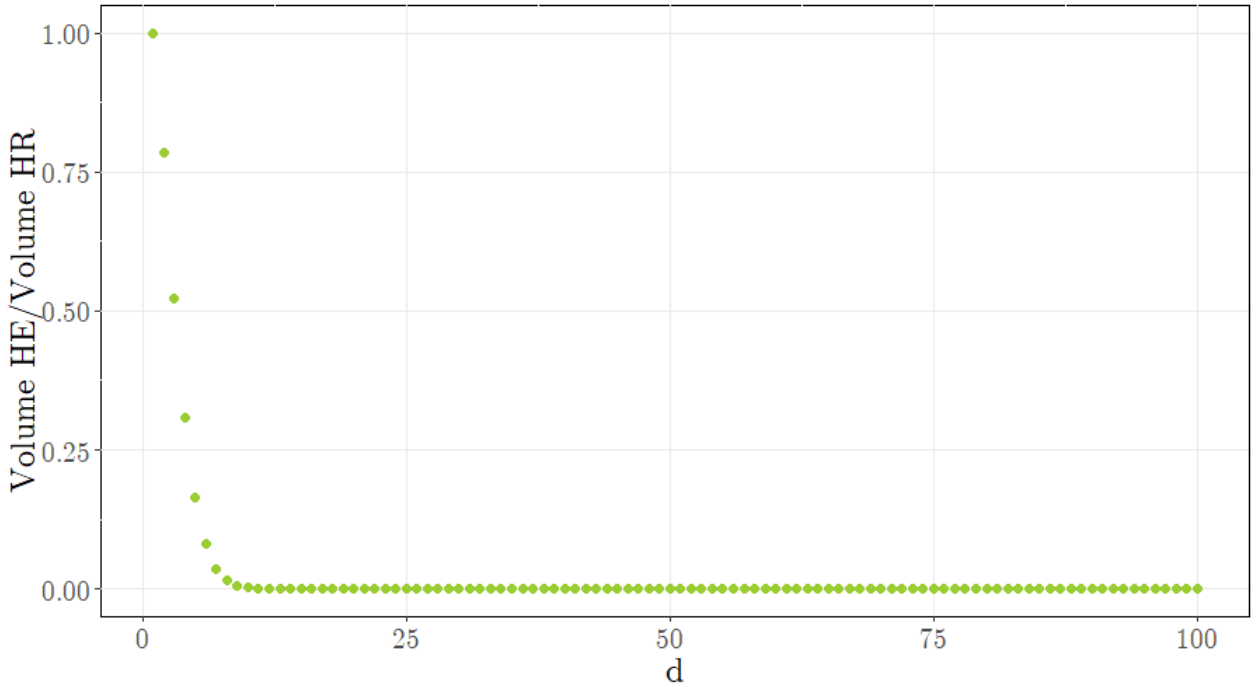


Figure 5.5: Probability of a point generated inside the hyperrectangle to be inside the hyperellipse, depending on the number of dimensions *d*.

To counteract this problem, a given number of points is generated inside the hyperrectangle (according to the number of dimensions, as it is explained in table 5.3). If none of those points are inside the hyperellipse, the one which is closer to it is chosen as the *moving point*. If one

or more points are inside the hyperellipse, one of them is chosen at random to be the *moving point*.

d	*#points to generate*
1	1
< 6	50
6,7	100
8	150
9	350
10	800
> 10	1000

Table 5.3: Number of points to generate given the number of dimensions, d, of the individual.

The values in column "*#points to generate*" of table 5.3 are such that they are higher than

$$\frac{1}{\text{Volume HE/Volume HR}} = \frac{\text{Volume HR}}{\text{Volume HE}},$$

which corresponds to the minimum number of points to generate to find 1 inside the hyperellipse (and those values are expressed in figure 5.6).

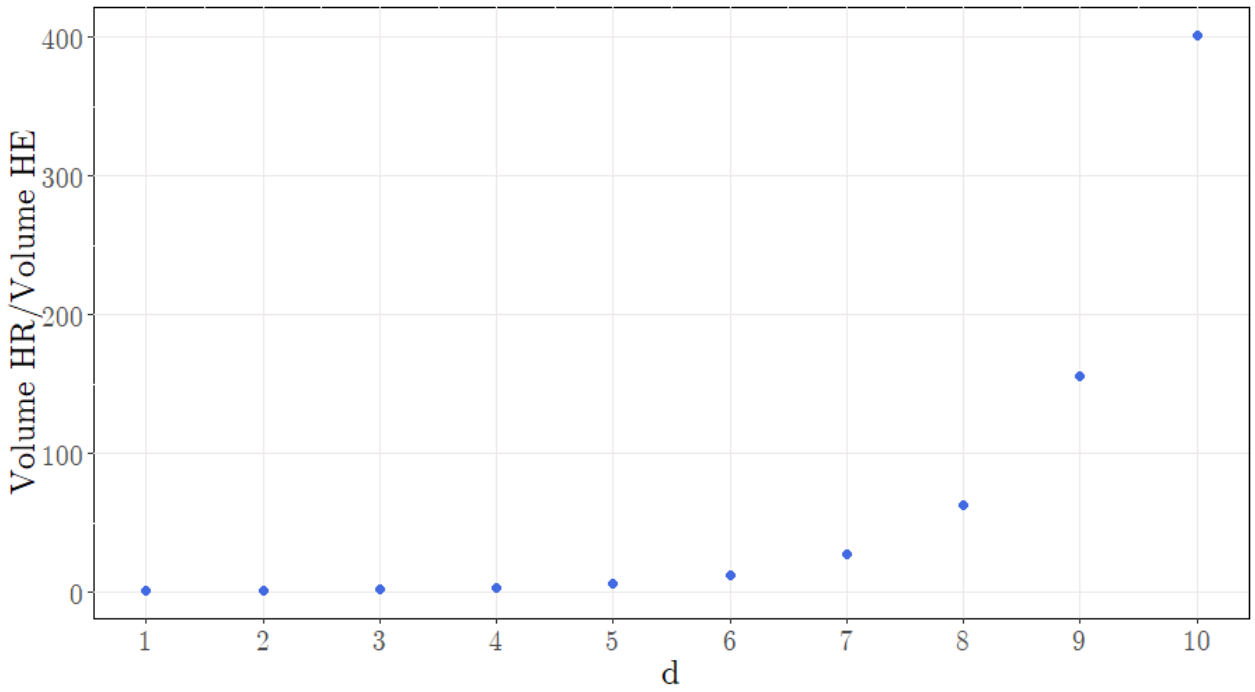


Figure 5.6: Given the number of dimensions, d, Volume HR/Volume HE gives the minimum number of points that have to be generated (inside the hyperrectangle) to find one inside the hyperellipse.

However, there are still things to consider and so 2 sub-variants of the Hyperellipse variant of **GSI-M3GP** are considered:

Sub-variant 1:

If the covariance matrix C is nonsingular and none of the eigenvalues of C^{-1} is too low (where we consider "too low" to be smaller than 10^{-10}), generate points as previously described. If the covariance matrix C is singular or at least one of the eigenvalues is too low, generate points inside the hypercircle with radius $\sqrt{\chi^2_{d,\alpha}}$ (an hypercircle is considered, because the Euclidean distance is used).

Three values of α are tested: 0.01, 0.5 and 0.75 (corresponding, respectively, to 99%, 50% and 25% prediction hyperellipses).

Sub-variant 2:

In sub-variant 2, only 2 values of α are tested: 0.01 and 0.5.

As sometimes there are misclassified points outside the hyperellipse, the percentage of those points is calculated. If that percentage is higher or equal to 50% then, a random point outside the hyperellipse is chosen (a point generated as in the hyperrectangle variant, that is not inside the hyperellipse). Otherwise, if the percentage of misclassified points outside the hyperellipse is smaller than 50%, do the same as in **sub-variant 1**.

This sub-variant considers the fact that the dimension variables might not follow a multivariate normal distribution.

5.2.4 Donut

As for the previous variant, a class is randomly chosen. Then, the instances being classified with that class but that do not belong to it are considered. These are referred to as misclassified instances.

In the Donut variant, one looks for points:

- in the hyperrectangle that contains all the mapped training samples (described in subsection 5.2.2), if the percentage of misclassified points outside the 99% prediction hyperellipse is higher or equal to 50%;
- inside the "donut", if the percentage of misclassified points outside the 99% prediction hyperellipse is smaller than 50% and, at the same time, the percentage of misclassified points outside the 50% prediction hyperellipse is higher or equal to 50%;
- inside the hyperrectangle that fits the 50% hyperellipse, *i.e.* the hyperrectangle with sides alligned with the hyperellipse axes and such that the hyperrectangle diagonals' intersection corresponds to the hypereelipse centroid (see figure 5.4), if the percentage of points inside the 50% prediction hyperellipse is smaller than 50%.

The "donut" refers to the space:

- inside the hyperrectangle that fits the 99% prediction hyperellipse;
- outside the hyperrectangle that fits the 50% prediction hyperellipse;

i.e. a rectangular "donut" or, more precisely, the points such that their Mahalanobis distance to the centroid is higher or equal to $\sqrt{\chi^2_{d,0.5}}$ and smaller or equal to $\sqrt{\chi^2_{d,0.01}}$.

It is important to notice that this variant only works while there are misclassified training instances.

5.2.5 Misclassified

A misclassified mapped training sample is chosen as the *moving point* - given a randomly chosen class, a training instance assigned with that class that does not belong to that class is chosen.

As for the previous variant, the algorithm can only work if there are misclassified training instances.

5.2.6 Misclassified-hyperrectangle

This variant is similar to the hyperrectangle variant. However, only the misclassified mapped training samples are considered. The *moving point* is a randomly generated point inside the hyperrectangle that contains the misclassified mapped training samples.

However, this variant only works while the individuals' training accuracy is less than 100%, since it only works if there are misclassified training instances.

5.2.7 Misclassified-hyperellipse

This variant is similar to the Hyperellipse variant. Here, as the name implies, an hyperellipse around the misclassified instances is considered and the *moving point* is a point generated from inside the hyperellipse.

First, a class is randomly chosen. Then the number of misclassified instances is counted. If there is only one misclassified instance (for that specific class), then the *moving point* y is a point in space such that $y_i \in [-\sigma_i + p_i, p_i + \sigma_i]$, where $i \in \{1, ..., d\}$, σ_i is the standard deviation of the i^{th} dimension of the parent individual (applied to the training instances) and $P = (p_1, ..., p_d)$ is the randomly chosen misclassified mapped training instance.

If there is more than one misclassified instance which is being assigned with the previous class then, a covariance matrix, Cov, and a centroid, $cent$, are computed based on the misclassified values' matrix. The *moving point* is a point inside the hyperellipse satisfying $(w - cent)^T Cov^{-1}(w - cent) \leq \chi^2_{d,\alpha}$.

As the Misclassified-hyperrectangle variant, this variant only works while there are misclassified training instances.

5.2.8 Around Misclassified

For a randomly chosen class, a misclassified mapped training sample $P = (p_1, ..., p_d)$ is randomly chosen. The *moving point* is randomly chosen, around that point. The way it is chosen depends on the variant. Two variants were considered, as follows.

σ-variant

The *moving point*, $y = (y_1, ..., y_d)$ satisfies $y_i \in [-a\sigma_i + p_i, p_i + a\sigma_i]$, where σ_i is the standard deviation of the i$^{\text{th}}$ dimension of the parent individual (applied to the training instances) and $a \in \mathbb{R}$.

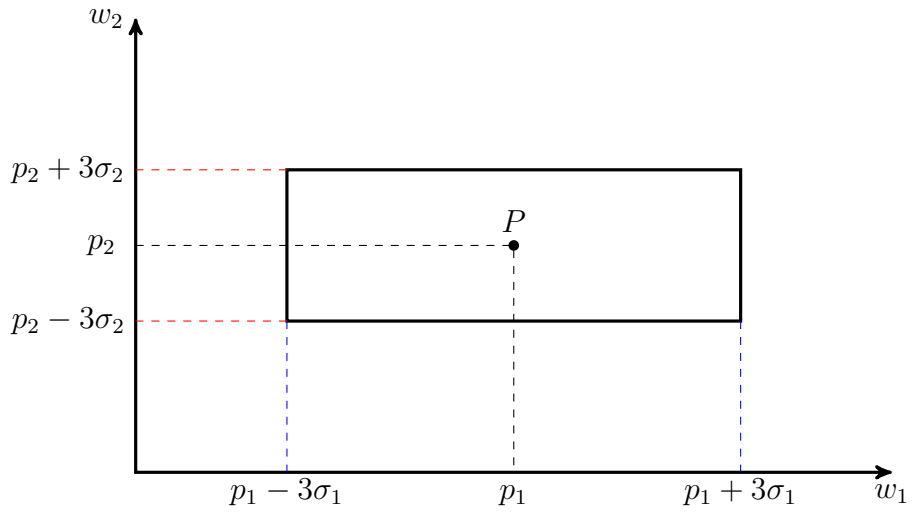


Figure 5.7: The 3σ-hyperrectangle for a 2-dimensional individual.

Hence, y is chosen inside an hyperrectangle that contains the points with each entry at maximum of a standard deviations from the corresponding entry of P (as it can be visualized for a 2-dimensional individual, in figure 5.7).

%-variant

The *moving point*, $y = (y_1, ..., y_d)$ satisfies $y_i \in [-kr_i + p_i, p_i + kr_i]$, with $k \in [0, 1]$ and $r_i = w_i^{max} - w_i^{min}$ (with w_i^{max} and w_i^{min} as previously defined). This variant is similar to the previous one, but here we consider a percentage of the range of values for each dimension.

Considering, as an example, a 2-dimensional individual for whom: $w_1^{max} = 50$, $w_1^{min} = 5$, $w_2^{max} = 2$, $w_2^{min} = -8$ and $k = 0.1$. Then, 10% of the range of values of dimension 1 and 2 are being considered, which corresponds to $r_1 = 0.1(50 - 5) = 4.5$ for dimension 1 and $r_2 = 0.1(2 - (-8)) = 1.0$ for dimension 2. Then, the *moving point* will be randomly generated from $[p_1 \pm 4.5] \times [p_2 \pm 1.0]$.

5.2.9 Correct-Misclassified distances

In order to create this new variant, the Hyperrectangle variant was considered and the distances between:

- each centroid and the correctly classified instances (the instances which are assigned to a specific class and belong to that class);
- the misclassified instances and the centroid of the class they are assigned to;

for each individual were calculated and averaged.

For each problem/dataset, a document with 5 columns and 150000 rows was created. The number of rows corresponds to the total number of individuals created (not considering the

initial population): 50 individuals per generation × 100 generations × 30 runs. The first column stores the operator that "created" the individual (*mutation*, *add branch* or *remove branch*). The remaining columns store, respectively:

- the average distance between the correctly classified instances and the centroid corresponding to the class they belong to, on the *training set*;
- the average distance between the misclassified instances and the centroid of the class they are assigned to, on the *training set*;
- the average distance between the correctly classified instances and the centroid corresponding to the class they belong to, on the *test set*;
- the average distance between the misclassified instances and the centroid of the class they are assigned to, on the *test set*;

From this file, 3 new ones were created, one for each operator. As such, the distances were grouped according to the operator used to create the individual.

For each column of these three files an average and median values were computed and the results are stored in table 5.4.

		Correct Train			Misclassified Train			Correct Test			Misclassified Test		
		Mutation	Add	Remove	Mutation	Add	Remove	Mutation	Add	Remove	Mutation	Add	Remove
HRT	avg	4.3×10^{7}	5.2×10^{12}	3.1×10^{5}	2.4×10^{7}	2.4×10^{12}	1.9×10^{5}	3.1×10^{7}	4.0×10^{12}	2.5×10^{5}	4.5×10^{7}	3.9×10^{12}	2.9×10^{5}
	med	3.315	36.69	3.178	3.325	38.11	3.275	3.521	35.65	3.369	3.322	33.51	3.254
IM-3	avg	4.9×10^{7}	8.1×10^{20}	3.8×10^{4}	2.8×10^{7}	2.5×10^{20}	3.3×10^{4}	3.3×10^{7}	9.9×10^{20}	3.5×10^{4}	4.0×10^{7}	6.5×10^{20}	3.6×10^{4}
	med	2.270	3.133	2.049	2.567	3.919	2.435	2.428	4.028	2.180	2.209	4.432	2.085
WAV	avg	4.397	9.9×10^{4}	4.351	4.448	1.0×10^{3}	4.404	4.445	2.6×10^{3}	4.395	4.443	1.4×10^{3}	4.394
	med	4.925	5.307	4.823	4.943	5.470	4.865	4.956	5.385	4.866	4.954	5.369	4.863
SEG	avg	2.5×10^{11}	2.3×10^{15}	4.3×10^{5}	1.6×10^{10}	1.3×10^{15}	1.3×10^{5}	9.2×10^{10}	4.5×10^{15}	2.6×10^{5}	2.1×10^{11}	1.0×10^{17}	3.2×10^{5}
	med	2.441	2.832	2.286	2.955	3.725	2.709	2.514	3.179	2.344	2.437	3.219	2.285
IM-10	avg	8.7×10^{12}	3.8×10^{22}	8.1×10^{8}	2.5×10^{12}	5.0×10^{21}	1.8×10^{8}	6.5×10^{12}	6.5×10^{21}	5.2×10^{8}	5.3×10^{12}	1.4×10^{22}	4.3×10^{8}
	med	3.010	466.8	2.885	3.514	372.1	3.371	3.086	423.4	2.965	3.057	433.4	2.944
YST	avg	2.611	443.9	2.428	2.842	258.2	2.341	3.395	421.8	2.589	3.096	338.6	2.515
	med	2.652	2.98	2.521	2.593	3.146	2.489	2.812	3.33	2.642	2.791	3.257	2.643
VOW	avg	20.218	3.9×10^{6}	3.804	5.231	7.5×10^{4}	4.239	11.185	1.2×10^{4}	4.628	9.667	1.2×10^{8}	4.200
	med	4.227	4.951	4.111	4.277	5.998	4.492	5.092	5.960	4.882	3.943	5.890	4.259
M-L	avg	2.963	214.0	2.813	2.500	26.32	2.597	5.759	22.34	4.807	4.499	44.35	4.638
	med	3.170	2.931	3.007	2.622	1.864	2.669	5.882	4.055	4.968	4.237	3.330	4.240

Table 5.4: Comparison between distances to centroid according to the operator, classification and partition, where *avg* referes to *average* and *med* refers to *median*.

Table 5.4 is organized in the following way. It has four main columns:

- *Correct Train (Test)*: with metrics concerning the correct *training (test)* instances;
- *Misclassified Train (Test):* with metrics concerning the misclassified *training (test)* instances.

Each of these columns is divided in columns *Mutation*, *Add* and *Remove*, each one storing metrics calculated for individuals created with *mutation*, *add branch* and *remove branch*, respectively.

From table 5.4 one can easily conclude that average distances for *Add* individuals are much higher than for *Mutation* and *Remove* individuals. One also notices that the median values are significantly smaller than the average values. As such, the mapped samples are quite dispersed in space.

Table 5.5 is organized in three main columns: Mutation, Add and Remove, corresponding, respectively, to the operators mutation, add branch and remove branch. Each of these columns is subdivided in Train, Test and Misc, corresponding, respectively, to a comparison between the training samples (both correctly classified and misclassified), test samples (both correctly classified and misclassified) and misclassified samples (training and test samples).

The following metrics were computed for each of the 150000 individuals and separated according to the operator used to create them:

- **Train**: Relative change$(Correct_{Train}, Misc_{Train}) = \frac{|\text{Misc}_{Train} - \text{Correct}_{Train}|}{\text{Correct}_{Train}} \times 100$
- **Test**: Relative change$(Correct_{Test}, Misc_{Test}) = \frac{|\text{Misc}_{Test} - \text{Correct}_{Test}|}{\text{Correct}_{Test}} \times 100$
- **Misc**: Relative change$(Misc_{Train}, Misc_{Test}) = \frac{|\text{Misc}_{Test} - \text{Misc}_{Train}|}{\text{Misc}_{Train}} \times 100$

Then, averages, standard deviations and medians of these values were computed and stored in table 5.5.

Dataset	Metrics	Mutation			Add			Remove		
		Train	Test	Misc	Train	Test	Misc	Train	Test	Misc
HRT	avg(s.d)	7%(7)	11%(25)	12%(44)	225%(> 100)	49%(> 100)	69%(> 100)	7%(6)	9%(13)	9%(20)
	median	6%	9%	8%	12%	9%	12%	6%	8%	7%
IM-3	avg(s.d)	16%(12)	24%(24)	28%(28)	42%(> 100)	17%(18)	23%(20)	19%(13)	20%(20)	23%(23)
	median	13%	17%	22%	20%	13%	19%	18%	14%	19%
WAV	avg(s.d)	3%(80)	4%(> 100)	5%(> 100)	9%(> 100)	3%(95)	6%(94)	3%(82)	4%(> 100)	6%(> 100)
	median	1%	1%	1%	4%	1%	3%	1%	1%	1%
SEG	avg(s.d)	136%(> 100)	3506%(> 100)	5536%(> 100)	436128%(> 100)	350%(> 100)	420%(> 100)	107%(> 100)	9%(> 100)	15%(> 100)
	median	19%	6%	15%	23%	6%	16%	18%	5%	12%
IM-10	avg(s.d)	16%(9)	2%(2)	12%(5)	104%(> 100)	3%(13)	17%(43)	16%(9)	2%(2)	12%(4)
	median	16%	2%	12%	18%	2%	12%	16%	2%	12%
YST	avg(s.d)	6%(53)	11%(80)	12%(> 100)	71%(> 100)	475%(> 100)	312%(> 100)	7%(18)	9%(44)	9%(41)
	median	4%	7%	8%	20%	10%	9%	4%	6%	7%
VOW	avg(s.d)	21%(> 100)	22%(> 100)	21%(> 100)	37%(> 100)	153%(> 100)	29%(> 100)	16%(49)	15%(> 100)	16%(> 100)
	median	12%	6%	8%	16%	6%	6%	13%	7%	9%
M-L	avg(s.d)	16%(> 100)	29%(57)	87%(99)	85%(> 100)	69%(> 100)	56%(> 100)	9%(11)	28%(> 100)	82%(> 100)
	median	10%	22%	63%	44%	10%	27%	8%	17%	6%

Table 5.5: Relative change on: **(Train)** the average distances for the correctly classified mapped training samples and the misclassified mapped training samples; **(Test)** the average distances for the correctly classified mapped test samples and the misclassified mapped test samples; **(Misc)** the average distances for the misclassified mapped training samples and the misclassified mapped test samples.

Dataset SEG is clearly an outlier when it comes to relative change, specially for mutation and add branch operators, for which the relative changes are very big.

For the remaining datasets, the average relative change on mutation and remove branch are very similar and the values are usually lower than 30%, always being below 100%. The median relative change is always smaller than the average relative change

For the add branch operator, the average relative change is higher than for the remaining two operators.

It is also interesting to notice that, for mutation and remove branch operators, the relative change between correctly classified and misclassified on test is usually higher than on train.

The values in table 5.4 and 5.5 imply that when generating the *moving point* taking into consideration its distance from the centroid, that distance should depend on the last operator applied to the individual.

Table 5.6 stores the results of the following analysis:

- (1): When $Correct_{Train} > Misc_{Train}$, then $Correct_{Test} > Misc_{Test}$? And when, $Correct_{Train} < Misc_{Train}$, then $Correct_{Test} < Misc_{Test}$?
- (2): Is $Misc_{Test} > Misc_{Train}$?

"SDirec $\longleftrightarrow$" represents the percentage of individuals for whom (1) happens and "Inc. $\nearrow$" represents the percentage of individuals for whom (2) happens.

Dataset	Metrics	Mutation	Add	Remove
HRT	SDirec $\longleftrightarrow$	58%	51%	50%
	Inc. $\nearrow$	48%	49%	53%
IM-3	SDirec $\longleftrightarrow$	43%	47%	45%
	Inc. $\nearrow$	69%	63%	70%
WAV	SDirec $\longleftrightarrow$	49%	53%	50%
	Inc. $\nearrow$	50%	78%	52%
SEG	SDirec $\longleftrightarrow$	40%	44%	45%
	Inc. $\nearrow$	88%	76%	81%
IM-10	SDirec $\longleftrightarrow$	43%	48%	43%
	Inc. $\nearrow$	97%	62%	97%
YST	SDirec $\longleftrightarrow$	49%	48%	52%
	Inc. $\nearrow$	15%	36%	15%
VOW	SDirec $\longleftrightarrow$	64%	50%	45%
	Inc. $\nearrow$	27%	53%	43%
M-L	SDirec $\longleftrightarrow$	74%	47%	57%
	Inc. $\nearrow$	14%	35%	10%

Table 5.6: Percentage of individuals for whom there was an increasing between

The value of "Inc. $\nearrow$" highly depends on the dataset ranging from 10% to 97%. It does also depend on the operator.

Regarding the value of "SDirec $\longleftrightarrow$", the values range from 40% to 74%, most of them being around 50% - implying that increase/decrease is in the same direction for approximately half of the individuals.

Two variants of **GSI-M3GP** were created based on the previous results. On both variants a class is randomly chosen and the algorithms runs while misclassified training instances exist.

Variant 1:

Variant 1 can still be sub-divided in the following sub-variants:

- **Average distances:**

  1. **All the points:**

     Consider the correctly classified mapped training samples. Let $Correct_{Train}$ be the *average* distance between a correctly classified mapped training sample and the centroid of the class it is being assigned with. Let $Misc_{Train}$ be the *average* distance between a mapped training sample and the centroid of the class it is being assigned to, although it does not belong to that class.

  2. **Assigned to class points:**

     Consider the correctly classified mapped training samples with class equal to the previously chosen class. Let $Correct_{Train}$ be the *average* distance between a correctly classified mapped training sample and the centroid of the chosen class. Let $Misc_{Train}$ be the *average* distance between a mapped training sample (that is being assigned to the previously chosen class, although it does not belong to that class) and the centroid of the previously chosen class.

- **Median distances:**

  1. **All the points:**

     Consider the correctly classified mapped training samples. Let $Correct_{Train}$ be the *median* distance between a correctly classified mapped training sample and the centroid of the class it is being assigned with. Let $Misc_{Train}$ be the *median* distance between a mapped training sample and the centroid of the class it is being assigned to, although it does not belong to that class.

2. **Assigned to class points:**

   Consider the correctly classified mapped training samples with class equal to the previously chosen class. Let $Correct_{Train}$ be the *median* distance between a correctly classified mapped training sample and the centroid of the chosen class. Let $Misc_{Train}$ be the *median* distance between a mapped training sample (that is being assigned to the previously chosen class, although it does not belong to that class) and the centroid of the previously chosen class.

Let:

- Relative Difference $= \frac{\text{Misc}_{Train} - \text{Correct}_{Train}}{\text{Correct}_{Train}}$;
- If using **Average** then: $a = \text{Correct}_{Train} \times (1 + rand \times \text{Relative Difference})$, where $rand \in [0, 1]$ is a randomly generated number.
- If using **Median** then: $a = \text{Correct}_{Train} \times (1 + \text{Relative Difference})$.

Then, the *moving point* is generated inside the hyperrectangle that fits (review figure 5.4) the hyperellipse defined by $(w - \mu)^T C^{-1}(w - \mu) = a^2$, where μ is the centroid and C is the covariance matrix.

If the covariance matrix is singular or one of the eigenvalues is too low then an hypercircle is considered (except for the constant dimensions for which the *moving point* will have the entry corresponding to that dimension equal to the constant).

As an example, let us consider that the **Average** sub-variant is considered and: $\text{Correct}_{Train} = 10$, $\text{Misc}_{Train} = 5$. Then, Relative Difference $= (5 - 10)/10 = -0.5$ and $a = 10 \times (1 - 0.5rand)$. Thus, $a \in [5, 10]$.

The reason why the value of a is different according to whether the average or median values are used is because the average values tend to be very big (as expressed in table 5.4).

Variant 2:

A misclassified mapped training sample that is being assigned with the previously chosen class is considered.

Consider the correctly classified mapped training samples. Let $Correct_{Train}$ be the *median* distance between a correctly classified mapped training sample and the centroid of the class it is being assigned with.

Let $a = \text{Correct}_{Train} \times rand \times inc$, where:

- $rand \in [0, 1]$ is a randomly generated number;
- $inc = 0.25$ or $inc = 1$;

Then, the *moving point* is generated inside the hyperrectangle that fits (review figure 5.4) the hyperellipse defined by $(w - \mu)^T C^{-1}(w - \mu) = a^2$, where μ is the centroid and C is the covariance matrix.

If the covariance matrix is singular or one of the eigenvalues is too low then an hypercircle is considered (except for the constant dimensions for which the *moving point* will have the entry corresponding to that dimension equal to the constant).

As such, one is looking for points around a misclassified training instance such that, the distance between the misclassified instance and that point is:

- at a maximum of 1/4 of the distance between the centroid and the misclassified instance (if $inc = 0.25$);
- at a maximum distance equal to the distance between the centroid and the misclassified instance (if $inc = 1$);

5.3 Experimental Setup and Results

The running parameters of **GSI-M3GP** are slightly different from the ones used in **M3GP**. Table 5.7 stores the running parameters which are common for all the variants of the algorithm.

Runs	30
Population size	50 individuals
Stopping criteria	100 generations (or $accuracy_{Train} = 100\%$)
Initialization	6-depth Full Initialization[11]
Operator probabilities	Equal probabilities
Function set	+, -, ×, ÷ protected as in [11]
Terminal set	Ephemeral random constants [0,1]
Bloat control	No depth limit
Selection	Tournament of size 5
Elitism	Keep best individual

Table 5.7: Running parameters of GSI-M3GP.

For some variants of **GSI-M3GP** a run of the algorithm terminates when the number of generations reaches 100 (as for **M3GP**), while for other variants the algorithm can only run while the training accuracy of the best individual is smaller than 100% (and if that value is never reached, then it stops at generation 100).

GSI-M3GP operators (*mutation*[2], *add new branch* and *remove branch*) have all the same probability of occurrence. The only thing to have into consideration is that one cannot remove a branch from an individual's tree if the tree is 1-dimensional.

The number of nodes is not being counted, since the number of nodes of each individual's dimension can easily get higher than $2^{63} - 1 = 9.223372 \times 10^{18}$. So, the tournament is not lexicographic. If there are individuals having the same training accuracy and that training accuracy is the highest, then one of the individuals is randomly chosen.

As it is done for **GSGP**, in **GSI-M3GP** it does not make sense to apply a depth limit.

[2]Where *mutation* is referring to *gsi-mutation*.

The datasets used for the experimental analysis are the ones from chapter 3 (see table 3.1).

The results of the following subsections are organized in different tables:

- **Accuracy and dimensions tables:** storing the training and test accuracies referring to a median of 30 runs and their standard deviations; in the same tables the median number of dimensions and, in parenthesis, the minimum and maximum number of dimensions. The values of accuracy marked in bold represent the best approaches for each dataset. If more than one value is in bold, it means that their difference is not statistically significant according to the Wilcoxon Rank-Sum Test with $\alpha = 0.01$.
- **Depth tables:** storing the median, maximum and minimum depth taking in consideration the depth of the 30 best individuals (1 per each run).
- **Mutation-Add-Remove percentage tables:** where the average percentage of each operator's usage (based on the 30 individuals resulting as outputs of the 30 runs) is stored. This is, for each final individual, the operations through which the individual passes since its creation (*mutation*, *add branch* and *remove branch* operations) are stored and, from that, the percentage of occurrence of *mutation*, *add branch* and *remove branch* are calculated. The values in the table are the averages on the 30 runs. M stands for the mutation operator, A stands for the add branch operator, and R stands for the remove branch operator.
- **Improvement tables:** tables storing the average, minimum and maximum number of operations through which the final individuals pass. For example, if a final individual is an individual such that, on each generation, an operation is applied to it, then the number of operations for that individual is 100. However, a smaller number of operations can be applied - if, for example, there is no improvement from a specific generation on.

5.3.1 Baseline

On the Baseline variant:

- stopping criterion: 100 generations;
- three values of *pred*% are tested: 10%, 1% and 0.0001%.

Table 5.8 stores the results of the comparison between M3GP and the baseline variant of GSI-M3GP. Baseline 10% refers to the baseline variant with $pred\% = 10\% = 0.1$, Baseline 1% refers to baseline variant with $pred\% = 1\% = 0.01$ and, finally, Baseline 0.0001% refers to the baseline variant with $pred\% = 0.0001\% = 0.000001$. Regarding training accuracy, the baseline variants can reach higher and significantly better training accuracies. However, test accuracy values are much smaller, and significantly worse, than M3GP ones (excepting: Baseline 0.0001% on WAV, SEG, IM-3 and M-L; and Baseline 10% on WAV). In general, the baseline variants are not competitive with M3GP.

The median number of dimensions is typically much higher for the baseline variants than for M3GP. That is due to the fact that gsi-mutation operator is not working well and so, the improvement from generation to generation mostly happens by adding dimensions.

However not competitive with M3GP, Baseline 0.0001% and Baseline 1% typically produce individuals with higher test accuracy than Baseline 10% (with the exception of HRT, WAV and VOW) - implying that a smaller value of *pred*% is favourable.

	HRT	**IM-3**	**WAV**	**SEG**	**IM-10**	**YST**	**VOW**	**M-L**
Training accuracy								
M3GP 50	93.1(3.1)	98.7(0.6)	89.6(0.6)	96.4(0.7)	92.3(0.8)	66.0(5.0)	**100**(0.0)	**100**(0.6)
Baseline 10%	99.2(0.6)	**100**(0.2)	90.3(0.3)	**97.2**(0.6)	**93.8**(0.6)	64.9(3.0)	**100**(0.0)	**100**(0.2)
Baseline 1%	**100**(0.4)	**100**(0.2)	**91.2**(0.7)	96.8(0.6)	93.2(0.5)	**66.8**(3.1)	**100**(0.0)	**100**(0.2)
Baseline 0.0001%	**99.7**(0.9)	**100**(0.4)	90.9(0.4)	96.5(0.5)	92.5(0.6)	**68.7**(3.8)	**100**(0.0)	**100**(0.0)
Test accuracy								
M3GP 50	**77.8**(4.0)	**94.3**(2.6)	**84.7**(0.9)	**95.1**(1.0)	**91.1**(0.9)	**55.6**(3.8)	**94.3**(2.3)	**60.6**(5.9)
Baseline 10%	51.2(11.6)	46.4(17.1)	**84.3**(0.9)	51.7(21.8)	36.4(14.4)	41.8(14.1)	16.2(6.3)	32.9(12.5)
Baseline 1%	53.7(9.0)	89.2(7.6)	83.0(1.3)	91.3(18.4)	85.2(8.3)	47.3(5.5)	8.9(5.4)	50.9(8.8)
Baseline 0.0001%	50.6(9.6)	88.7(6.7)	**84.6**(0.9)	**94.5**(4.8)	**90.7**(0.8)	50.2(4.8)	10.1(16.5)	**58.3**(0.9)
# of dimensions								
M3GP 50	14(1-19)	5.5(2-10)	26.5(20-32)	8(5-14)	15(8-24)	12(2-17)	22(19-25)	11(8-12)
Baseline 10%	31.5(20-46)	22(10-36)	30(24-35)	13(6-20)	22.5(15-27)	13(7-21)	34(27-43)	11(9-13)
Baseline 1%	34(21-44)	18(4-37)	39(33-57)	10(5-19)	19(14-24)	14.5(11-20)	37.5(29-45)	11(9-13)
Baseline 0.0001%	31.5(16-40)	18(5-32)	33(28-39)	9(5-17)	16(11-23)	17.5(4-26)	26(23-34)	11(9-13)

Table 5.8: Comparison between the M3GP and GSI-M3GP Baseline 0.0001%, Baseline 1% and Baseline 10% variants.

Table 5.9 stores the median, minimum and maximum depth of the trees produced with the baseline variants. Baseline 0.0001% produces deeper trees than Baseline 1%, and Baseline 1% produces deeper trees than Baseline 10%, implying that gsi-mutation is working better for variants with a smaller value of *pred*%.

	HRT	**IM-3**	**WAV**	**SEG**	**IM-10**	**YST**	**VOW**	**M-L**
Baseline 10%	77	101	1	21	35	20	63	49
	(41-129)	(29-155)	(0-14)	(10-49)	(22-50)	(3-78)	(42-84)	(6-77)
Baseline 1%	155	153	91	73.5	69.5	91.5	180.5	224.5
	(91-218)	(76-258)	(50-119)	(10-161)	(6-118)	(6-154)	(92-252)	(41-315)
Baseline 0.0001%	178	188	241.5	162	75.5	234	290.5	402
	(84-279)	(97-294)	(176-315)	(98-249)	(41-197)	(104-349)	(218-364)	(274-483)

Table 5.9: Comparison between Baseline 0.0001%, Baseline 1% and Baseline 10% depth values.

One can also reach that conclusion from table 5.10. *Mutation* percentages grow from Baseline 10% to Baseline 0.0001%. However, the *mutation* percentages are still low, when compared to the *add branch* percentages. Even for Baseline 0.0001%, the *mutation* percentages are lower than the *add* branch percentages (except on M-L dataset), implying that the best individuals of each run are the ones for which most of the operations applied were *add dimension* operations.

		HRT	**IM-3**	**WAV**	**SEG**	**IM-10**	**YST**	**VOW**	**M-L**
	M	13%	14%	1%	4.64%	6.5%	4.2%	9%	10%
Baseline 10%	*A*	66%	55%	83%	67.73%	72.1%	66.5%	63%	53%
	R	21%	31%	16%	27.63%	21.4%	29.3%	28%	37%
	M	23.5%	25%	14%	15%	13%	21%	25%	39%
Baseline 1%	*A*	58.3%	48%	76%	59%	67%	58%	57%	38%
	R	18.2%	27%	10%	26%	20%	21%	18%	23%
	M	28%	32%	41%	35%	16.3%	48%	42%	62.2%
Baseline 0.0001%	*A*	56%	45%	54%	45%	63.3%	42%	43%	24.4%
	R	16%	23%	5%	20%	20.4%	10%	15%	13.4%

Table 5.10: Comparison between Baseline 0.0001%, Baseline 10% and Baseline 1% operators' percentages

Table 5.11 shows the average, minimum and maximum number of operations applied to the final individuals. The average values are higher than 90 for HRT, IM-3 and VOW on all baseline variants. On WAV, SEG and YST the maximum values are always smaller than 100. These values suggest that on some generations there was no improvement, or that, from a specific

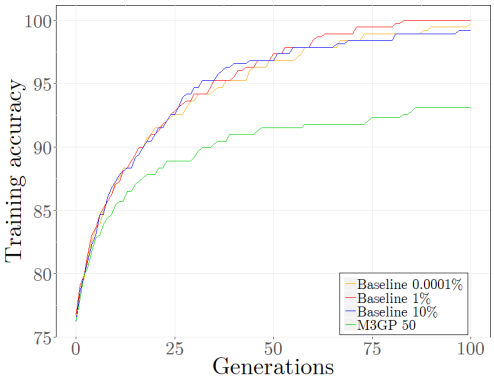
generation on, there is no more improvement until generation 100 is reached. As such, the algorithm might stop learning after a given number of generations. This is no surprise taking into consideration the way the moving point is being chosen: for datasets with low number of training samples (lower than 300) like HRT, IM-3, VOW and M-L it might be easy to change the mapped training instances from one place to another in such a way that they get correctly classified; however, for datasets like WAV, SEG, IM-10 and YST, which have a lot of training samples (respectively, 1500, 693, 2039 and 445), it might be more difficult. For the M-L dataset, the minimum number of operations is very low given that, for this algorithm, the 100% training accuracy is reached at a quite low generation (see figure 5.9g).

	HRT	**IM-3**	**WAV**	**SEG**	**IM-10**	**YST**	**VOW**	**M-L**
Baseline 10%	90.6 (69-100)	96.6 (82-100)	71.6 (53-85)	72.3 (57-88)	69.4 (45-83)	61.3 (28-95)	99.7 (97-100)	77.6 (25-100)
Baseline 1%	95.4 (88-100)	92.0 (66-100)	85.1 (71-94)	73.7 (50-90)	69.4 (52-90)	59.0 (37-89)	99.1 (97-100)	76.8 (24-100)
Baseline 0.0001%	92.9 (90-100)	91.5 (64-100)	84.4 (63-97)	69.2 (51-91)	70.4 (52-87)	66.6 (37-91)	97.6 (89-100)	92.4 (52-100)

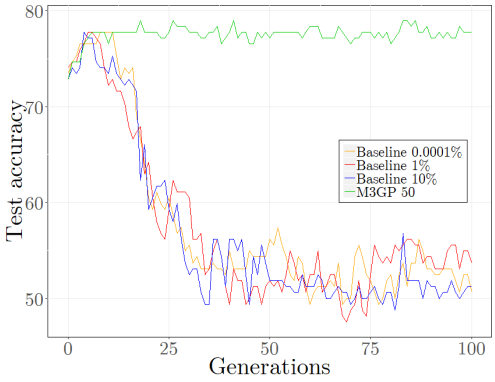
Table 5.11: Comparison between Baseline 0.0001%, Baseline 10% and Baseline 1% number of operations.

Figures 5.8 and 5.9 show the evolution of training and test accuracies as the number of generations increases. Baseline 10% and Baseline 1% overfit on every problem (except on WAV, where only Baseline 1% shows slight signs of overfitting). Baseline 0.0001% has a similar behavior as M3GP on WAV, SEG, IM-10 and M-L test sets (see figures 5.8e,f, 5.8g,h, 5.9a,b and 5.9g,h), overfitting on M-L. On the remaining datasets, Baseline 0.0001% overfits, although always less than Baseline 10% (except on HRT, where all the baseline variants behave in the same way).

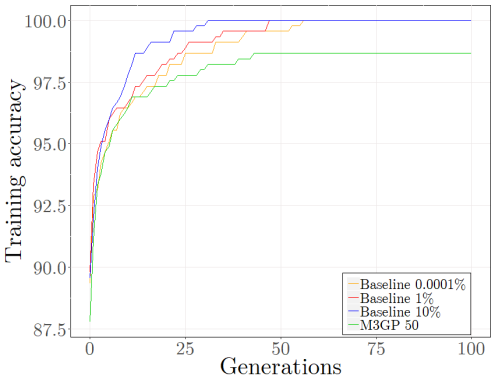
The behaviour of the baseline variants on the last generation, both on training and test set, can be seen in figures 5.10 and 5.11. All the baseline variants are typically able to produce last generation individuals with higher training accuracy when compared to M3GP. Regarding test accuracy values Baseline 1% and Baseline 10% produce smaller values than M3GP.



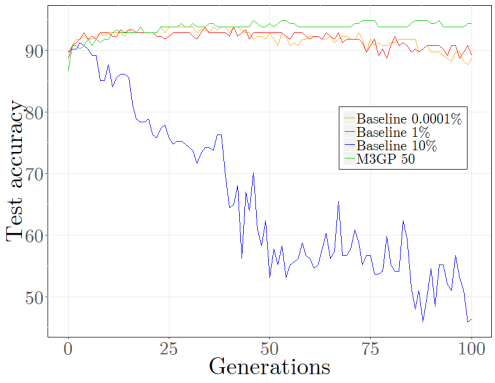
(a) HRT - Training.



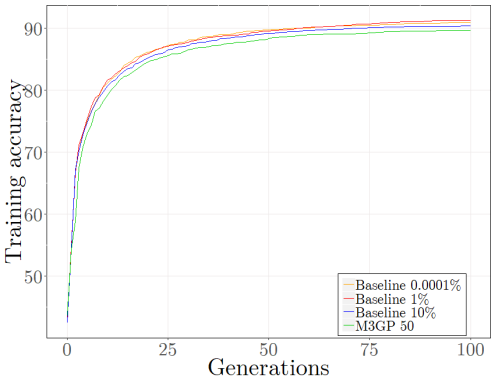
(b) HRT - Test.



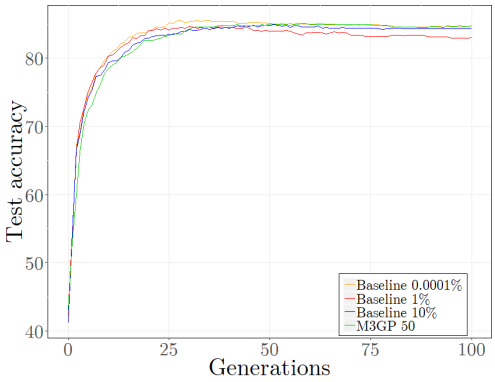
(c) IM-3 - Training.



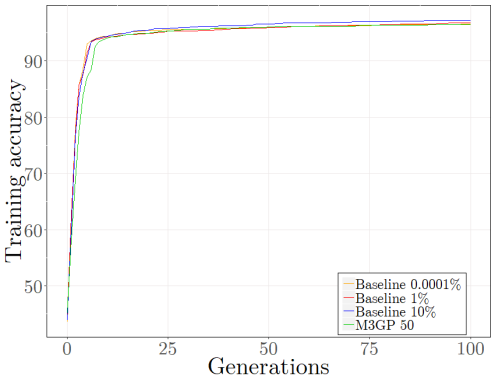
(d) IM-3 - Test.



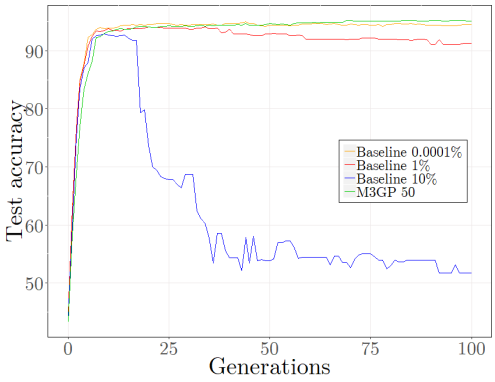
(e) WAV - Training.



(f) WAV - Test.

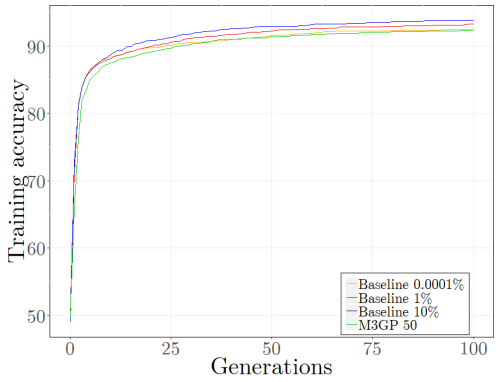


(g) SEG - Training.

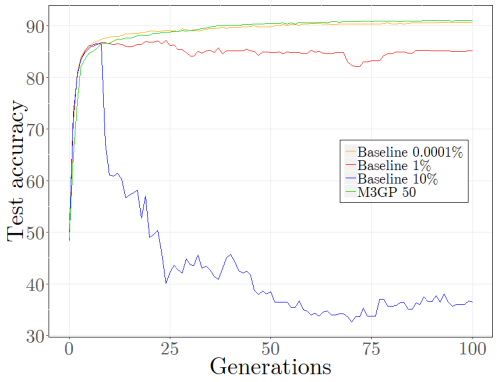


(h) SEG - Test.

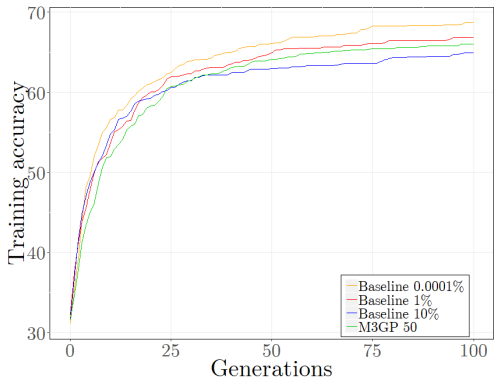
Figure 5.8: Comparison between median accuracies of M3GP 50, Baseline 0.0001% Baseline 1% and Baseline 10% algorithms on HRT, IM-3, WAV and SEG datasets.



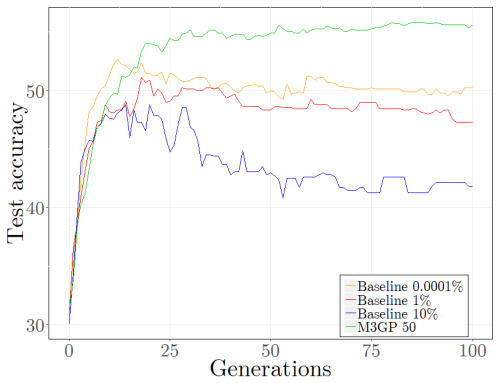
(a) IM-10 - Training.



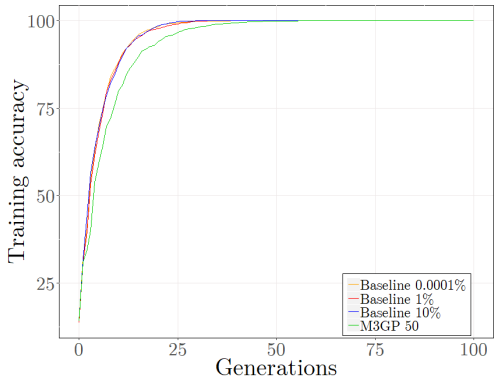
(b) IM-10 - Test.



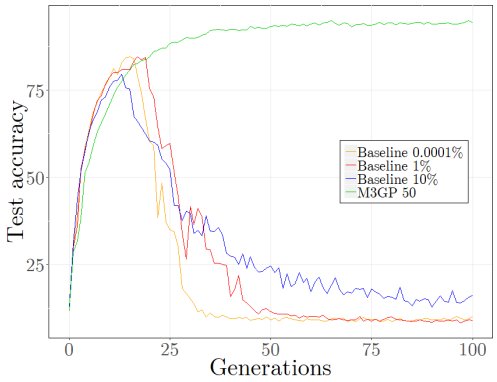
(c) YST - Training.



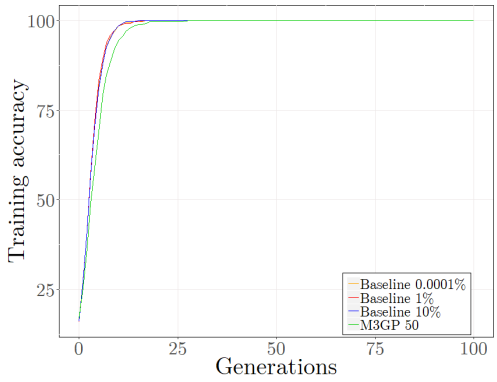
(d) YST - Test.



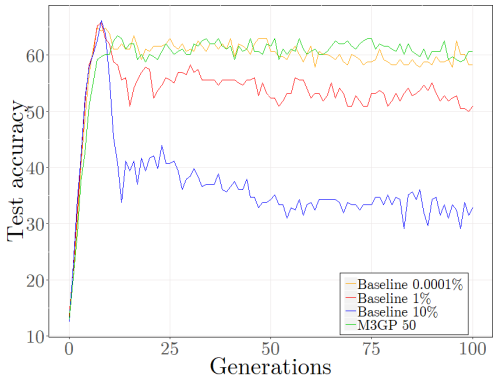
(e) VOW - Training.



(f) VOW - Test.

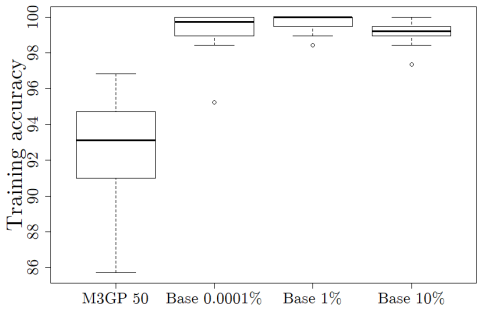


(g) M-L - Training.

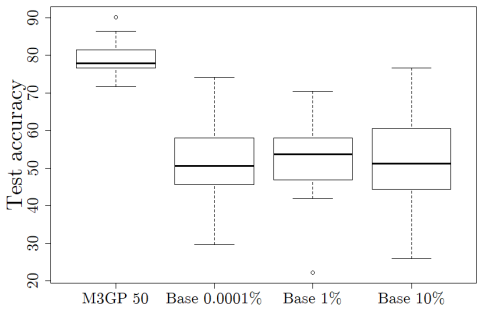


(h) M-L - Test.

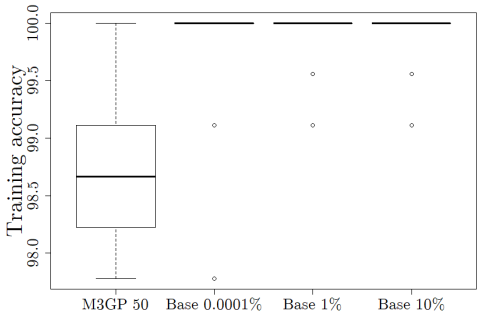
Figure 5.9: Comparison between median accuracies of M3GP 50, Baseline 0.0001%, Baseline 1% and Baseline 10% algorithms on IM-10, YST, VOW and M-L datasets.



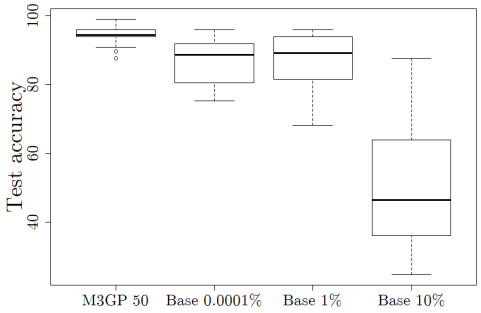
(a) HRT - Training.



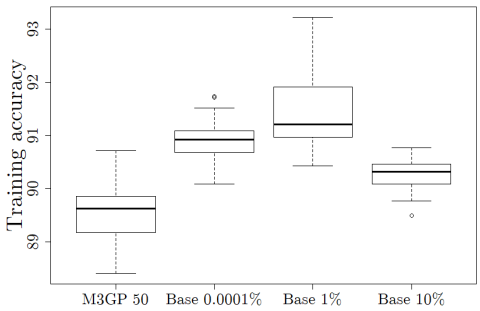
(b) HRT - Test.



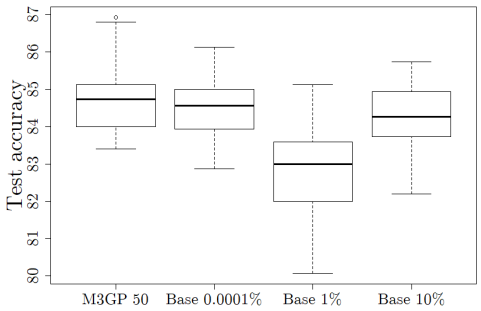
(c) IM-3 - Training.



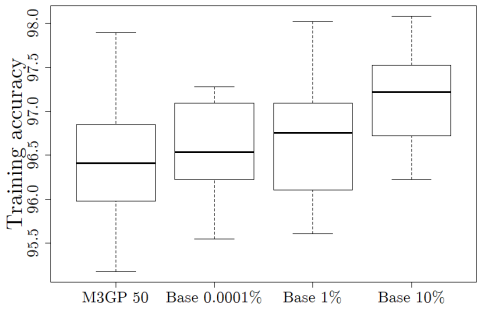
(d) IM-3 - Test.



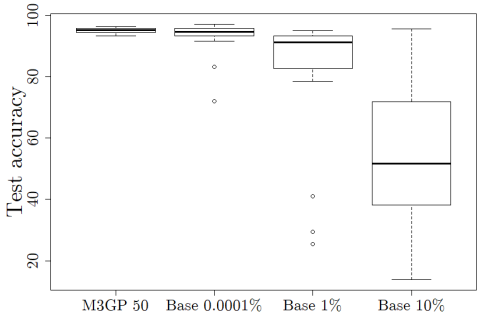
(e) WAV - Training.



(f) WAV - Test.

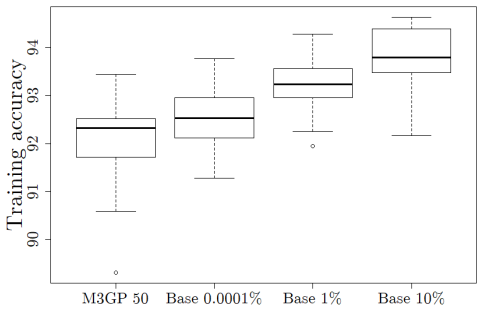


(g) SEG - Training.

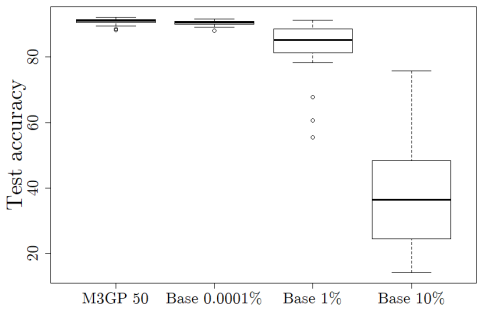


(h) SEG - Test.

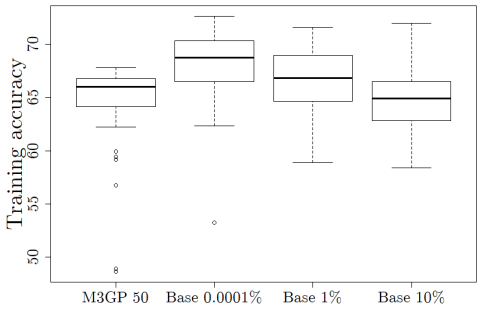
Figure 5.10: Comparison between last generation individuals' accuracies of M3GP 50, Baseline 0.0001%, Baseline 1% and Baseline 10% algorithms on HRT, IM-3, WAV and SEG datasets.



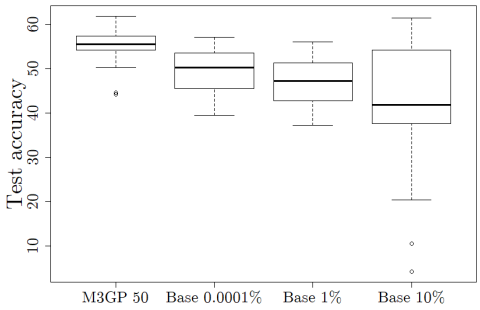
(a) IM-10 - Training.



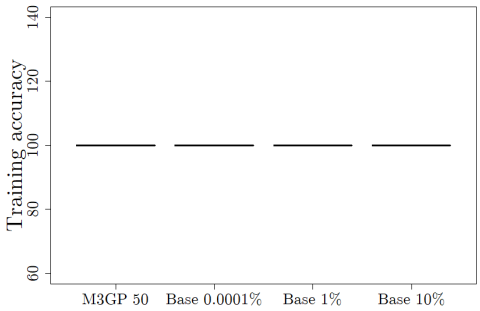
(b) IM-10 - Test.



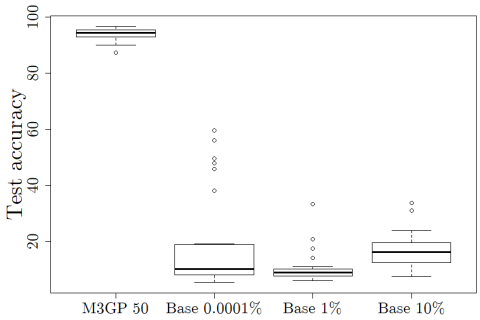
(c) YST- Training.



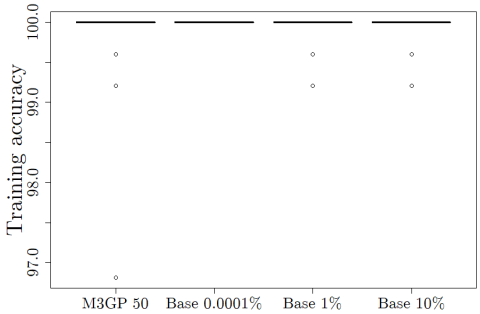
(d) YST - Test.



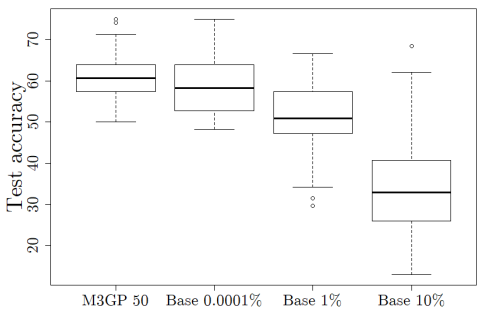
(e) VOW - Training.



(f) VOW - Test.



(g) M-L - Training.



(h) M-L - Test.

Figure 5.11: Comparison between last generation individuals' accuracies of M3GP 50, Baseline 0.0001%, Baseline 1% and Baseline 10% algorithms on IM-10, YST, VOW and M-L datasets.

On the last generation, Baseline 0.0001% behaves similarly to Baseline 10% on HRT and VOW test sets (see figures 5.10b and 5.11f). Figures 5.10f, 5.10h, 5.11b and 5.11b show that, on the last generation, Baseline 0.0001% behaves similarly to M3GP on WAV, SEG, IM-10 and M-L test sets.

For most of the datasets, Baseline 10% produces last generation individuals with a much wider range of test accuracy values than Baseline 1% and Baseline 0.0001%, implying that Baseline 10% is less stable than Baseline 1% and Baseline 0.0001%.

5.3.2 Hyperrectangle

On the Hyperrectangle variant (HR):

- the stopping criterion: 100 generations;
- $pred\% = 10^k\%$, with $k \in \{1, 0, -1, .., -6\}$, and $pred\% \in]0, 10^p]\%$, with $p \in \{-1, -2, -3\}$.

One of the conclusions taken from the previous section was that smaller values of $pred\%$ would imply a bigger usage of the mutation operator. As such, in this section, a wider range of $pred\%$ values is tested.

Standard variant:

Table 5.12 stores the results of training and test accuracy of M3GP and the hyperrectangle variants of GSI-M3GP. HR $10^k\%$ stands for hyperrectangle with $pred\% = 10^k\%$. Regarding training accuracy, higher values of $pred\%$ produce higher median training accuracy values which are, for most of the problems, higher than the training accuracy values on M3GP. HR $10^k\%$ with $k \in \{-6, ..., -2\}$ produce median test accuracy values which are not statistically different from the median test accuracy values produced by M3GP. Hence, by having a lower value of $pred\%$, the hyperrectangle variant of GSI-M3GP is competitive with M3GP on all datasets.

Regarding number of dimensions, individuals created using Hyperrectangle variants of GSI-M3GP with *pred*% $\leq$ 1% tend to have a higher median number of dimensions than M3GP individuals (except for VOW and M-L), but smaller than the variants with *pred*% = 10% (except for M-L). This is acceptable since M3GP works with 5 operators and GSI-M3GP with only 3.

	HRT	**IM-3**	**WAV**	**SEG**	**IM-10**	**YST**	**VOW**	**M-L**
Training accuracy								
M3GP 50	93.1(3.2)	98.7(0.6)	89.6(0.6)	96.4(0.7)	92.3(0.8)	66.0(5.0)	**100**(0.0)	**100**(0.0)
HR 10%	**99.5**(0.8)	**100**(0.0)	90.4(0.3)	**98.4**(0.6)	**95.0**(0.5)	**69.0**(2.1)	**100**(0.0)	**100**(1.3)
HR 1%	**99.5**(1.9)	100(0.3)	**91.4**(0.5)	96.8(0.6)	92.4(0.5)	**68.6**(4.0)	**100**(0.0)	**100**(0.3)
HR 10^{-1}%	97.1(1.8)	99.6(0.6)	91.1(0.5)	96.5(0.5)	91.8(0.9)	66.9(3.3)	**100**(0.0)	**100**(0.1)
HR 10^{-2}%	95.0(1.8)	99.6(0.8)	90.9(0.4)	96.7(0.5)	92.2(0.9)	66.3(2.9)	**100**(0.0)	**100**(0.3)
HR 10^{-3}%	93.9(2.2)	98.4(1.0)	90.9(0.5)	96.8(0.6)	92.3(0.6)	64.9(3.4)	**100**(0.0)	**100**(1.1)
HR 10^{-4}%	94.2(1.7)	98.7(0.6)	90.9(0.5)	96.6(0.7)	92.5(0.6)	65.5(2.6)	**100**(0.0)	**100**(0.4)
HR 10^{-5}%	93.9(2.5)	98.7(0.8)	90.7(0.3)	96.4(0.7)	92.2(0.5)	65.0(2.3)	**100**(0.0)	**100**(0.4)
HR 10^{-6}%	93.7(2.0)	98.7(1.2)	90.7(0.4)	96.4(0.7)	92.2(0.7)	65.6(2.2)	**100**(0.0)	**100**(0.7)
Test accuracy								
M3GP 50	**77.8**(4.0)	**94.3**(2.6)	**84.7**(0.9)	**95.1**(1.0)	**91.1**(0.9)	**55.6**(3.8)	**94.3**(2.3)	**60.6**(5.9)
HR 10%	51.9(10.4)	38.1(16.9)	83.7(1.2)	44.6(19.5)	29.2(10.7)	31.3(8.6)	10.4(3.5)	24.1(13.0)
HR 1%	69.1(13.4)	91.2(4.9)	81.9(1.3)	93.3(9.2)	89.5(2.4)	52.2(6.2)	64.3(16.5)	49.1(8.3)
HR 10^{-1}%	**77.8**(8.3)	**94.8**(3.0)	84.0(0.9)	**95.5**(0.4)	90.5(1.2)	**55.9**(2.5)	85.4(6.2)	**62.5**(7.1)
HR 10^{-2}%	**77.2**(5.5)	**94.8**(3.4)	**84.2**(1.0)	**95.2**(1.1)	**91.0**(1.1)	**55.9**(2.9)	**92.6**(3.1)	**60.2**(6.3)
HR 10^{-3}%	**79.0**(4.5)	**94.3**(3.0)	**84.5**(0.9)	**95.2**(0.9)	**91.0**(0.7)	**55.7**(2.1)	**93.6**(3.0)	**59.3**(6.1)
HR 10^{-4}%	**79.0**(3.8)	**95.4**(2.3)	**84.0**(0.9)	**94.8**(0.8)	**91.3**(0.7)	**55.4**(2.4)	**93.8**(2.1)	**61.1**(5.2)
HR 10^{-5}%	**77.8**(4.7)	**94.8**(2.3)	**84.0**(1.0)	**94.7**(1.0)	**90.9**(0.5)	**55.5**(2.5)	**93.6**(1.9)	**61.6**(7.6)
HR 10^{-6}%	**79.0**(3.6)	**95.4**(2.7)	**84.1**(0.9)	**94.8**(1.3)	**91.2**(1.0)	**55.8**(1.9)	**93.8**(2.0)	**58.3**(6.6)
# of dimensions								
M3GP 50	14(1-19)	5.5(2-10)	26.5(20-32)	8(5-14)	15(8-24)	12(2-17)	22(19-25)	11(8-12)
HR 10%	32(19-44)	31(13-38)	31(28-39)	19(8-31)	30(21-36)	17.5(10-26)	41(35-55)	11(7-13)
HR 1%	29.5(9-38)	11.5(2-24)	36.5(29-47)	9(5-16)	16(11-21)	16(11-25)	38(35-45)	11(8-12)
HR 10^{-1}%	19.5(7-29)	8(1-22)	32(22-39)	9.5(6-15)	13.5(8-21)	14(10-24)	30(24-36)	11(8-13)
HR 10^{-2}%	16.5(5-21)	8(3-14)	32(24-38)	9(5-14)	14(7-19)	14(10-25)	23.5(19-28)	11(9-12)
HR 10^{-3}%	14.5(6-21)	8(2-16)	32.5(26-37)	11(6-20)	15(10-21)	14(8-21)	22(19-27)	10.5(8-13)
HR 10^{-4}%	16(7-24)	9(4-15)	34(26-38)	11(6-18)	17(9-24)	14.5(10-28)	23(18-27)	11(8-12)
HR 10^{-5}%	15(4-22)	8.5(4-14)	33(25-37)	11(6-16)	17(12-24)	14(11-21)	22.5(19-27)	11(9-12)
HR 10^{-6}%	16(9-21)	9(3-12)	33(26-38)	11(7-20)	18.5(8-21)	15(10-21)	21.5(18-25)	11(8-13)

Table 5.12: Comparison between the M3GP and GSI-M3GP hyperrectangle variants with *pred*% equal to 10^k%, with $k \in \{1, 0, -1, ..., -6\}$.

For WAV and VOW datasets, depth median values tend to increase as the *pred*% decreases, not knowing what happens for *pred*% $< 10^{-6}$ (see table 5.13). For the remaining problems, the median depth tends to increase as *pred*% decreases until a certain value, and then the median depth starts decreasing. This suggests that, by increasing the value of *pred*%, the mutation operator becomes more and more powerful. This can also be seen by looking at the operators' percentages: as the predefined percentage decreases, the *mutation percentage* increases until a certain point, and then it decreases slightly and stabilizes (see table 5.14).

	HRT	**IM-3**	**WAV**	**SEG**	**IM-10**	**YST**	**VOW**	**M-L**
HR 10%	148.5 (90-211)	161.5 (93-230)	7 (0-28)	57.5 (9-140)	85.5 (54-125)	108.5 (55-273)	164 (119-212)	1015 (21-230)
HR 1%	252 (160-449)	230 (71-377)	119 (65-177)	98.5 (9-238)	35.5 (4-86)	223 (90-426)	206.5 (128-267)	297 (139-461)
HR 10^{-1}%	398 (279-559)	442.5 (236-615)	274.5 (196-357)	214 (71-321)	90.5 (29-279)	356 (251-510)	301 (247-350)	482.5 (356-601)
HR 10^{-2}%	497.5 (413-601)	548.5 (387-629)	315 (245-406)	444.5 (329-562)	265.5 (139-545)	493.5 (251-545)	407 (284-448)	556.5 (414-643)
HR 10^{-3}%	525 (413-671)	558 (449-636)	371 (315-448)	498 (401-638)	459 (344-587)	538 (420-601)	410.5 (323-532)	580.5 (412-643)
HR 10^{-4}%	507 (285-552)	538 (406-622)	371 (303-448)	514.5 (434-609)	471.5 (384-580)	524 (349-608)	424.5 (330-560)	528 (373-601)
HR 10^{-5}%	512 (419-657)	539 (387-657)	385 (343-476)	532.5 (441-615)	486 (393-559)	517 (443-573)	434.5 (332-503)	577.5 (377-643)
HR 10^{-6}%	507.5 (426-636)	531 (447-657)	392 (330-455)	518 (441-616)	479 (385-608)	520.5 (384-629)	434 (366-509)	555.5 (392- 636)

Table 5.13: Comparing depth values between GSI-M3GP hyperrectangle variants with *pred*% equal to 10^k%, with $k \in \{1, 0, -1, ..., -6\}$.

		HRT	**IM-3**	**WAV**	**SEG**	**IM-10**	**YST**	**VOW**	**M-L**
	M	23.5%	23%	2%	11.4%	16%	25.2%	24%	23%
HR 10%	*A*	58.8%	54%	82%	66.3%	69%	57.4%	59%	47%
	R	17.7%	23%	16%	22.3%	15%	17.4%	17%	30%
	M	41%	37%	20%	22%	9%	47%	30%	54%
HR 1%	*A*	46%	40%	72%	54%	68%	43%	55%	30%
	R	13%	23%	8%	24%	23%	10%	15%	16%
	M	62%	65%	46%	41%	26%	66%	44%	69.3%
HR 10^{-1}%	*A*	30%	22%	50%	42%	56%	28%	44%	20.4%
	R	8%	13%	4%	17%	18%	6%	12%	10.3%
	M	71.8%	77%	50%	71%	55%	73.8%	57%	78%
HR 10^{-2}%	*A*	22.6%	16%	46%	21%	36%	21.7%	34%	16%
	R	5.6%	7%	4%	8%	9%	4.5%	9%	6%
	M	75.4%	78.6%	55.4%	75%	71%	77%	59%	80%
HR 10^{-3}%	*A*	20.1%	14.7%	42.1%	19%	24%	20%	32%	15%
	R	4.5%	6.7%	2.5%	6%	5%	3%	9%	5%
	M	72%	76.2%	55.8%	75%	70%	75%	61.6%	74%
HR 10^{-4}%	*A*	23%	16.5%	41.6%	20%	26%	21%	30.7%	18%
	R	5%	7.3%	2.6%	5%	4%	4%	7.7%	8%
	M	74%	78%	57%	77%	70%	75%	62%	80%
HR 10^{-5}%	*A*	21%	15%	41%	18%	25%	21%	31%	15%
	R	5%	7%	2%	5%	5%	4%	7%	5%
	M	74%	77%	57%	76%	70%	75%	62%	77.4%
HR 10^{-6}%	*A*	21%	16%	41%	19%	26%	21%	30%	16.3%
	R	5%	7%	2%	5%	4%	4%	8%	6.3%

Table 5.14: Comparing operators' percentages between the GSI-M3GP hyperrectangle variants with *pred*% equal to 10^k%, with $k \in \{1, 0, -1, ..., -6\}$.

For values of *pred*% smaller than 10^{-1}%, the *mutation percentage* is higher than the *add branch percentage* for all problems. This is good, since it means that the gsi-mutation operator is working well, making the individuals improve.

Table 5.15 stores the average number of operations for the different values of *pred*% on the hyperrectangle variants. On the hyperrectangle variants with *pred*% $\leq 10^{-3}$%, the average number of operations is higher than 90 on all problems. On variants with *pred*% $\leq 10^{-4}$%, the average and minimum number of operations are higher than 90. This means that the training accuracy improves on almost all generations, in all of the runs.

	HRT	**IM-3**	**WAV**	**SEG**	**IM-10**	**YST**	**VOW**	**M-L**
HR 10%	93.8 (77-100)	97.6 (84-100)	71.0 (62-81)	80.3 (63-94)	78.1 (64-96)	65.2 (39-87)	99.5 (98-100)	66.9 (22-100)
HR 1%	92.3 (80-100)	86.7 (60-100)	83.1 (65-96)	70.5 (47-94)	61.2 (42-92)	71.1 (43-93)	98.9 (95-100)	77.8 (33-100)
HR 10^{-1}%	93.9 (83-100)	94.8 (81-100)	85.1 (64-95)	73.8 (54-88)	61.0 (36-75)	77.6 (52-100)	98.5 (94-100)	98.3 (86-100)
HR 10^{-2}%	98.5 (95-100)	98.6 (91-100)	91.4 (80-98)	90.5 (74-99)	70.6 (59-85)	92.0 (82-100)	99.1 (96-100)	99.4 (95-100)
HR 10^{-3}%	99.1 (97-100)	99.4 (98-100)	96.1 (92-100)	96.7 (90-99)	93.0 (84-98)	96.6 (92-100)	99.2 (97-100)	99.3 (97-100)
HR 10^{-4}%	98.3 (96-100)	99.2 (97-100)	97.7 (91-100)	98.2 (93-100)	96.6 (91-100)	97.7 (95-100)	99.1 (95-100)	99.1 (97-100)
HR 10^{-5}%	98.3 (94-100)	99.3 (97-100)	97.9 (94-100)	98.2 (95-100)	97.7 (91-100)	97.6 (92-100)	99.4 (97-100)	99.7 (98-100)
HR 10^{-6}%	99.0 (96-100)	98.9 (95-100)	98.1 (93-100)	98.8 (94-100)	97.3 (94-100)	98.4 (95-100)	99.4 (97-100)	99.2 (96-100)

Table 5.15: Comparing the number of operations between the GSI-M3GP hyperrectangle variants with *pred*% equal to 10^k%, with $k \in \{1, 0, -1, ..., -6\}$.

Moving point in a centroid's position:

Remembering the definition of gsi-mutation operator, in section 5.1, the i^{th} dimension of the new individual, $g_i(x)$ is defined using the i^{th} dimension of the parent individual, $f_i(x)$, as follows: $g_i(x, y) = betw(y_i(x), c_i) \times peak(f_i(x), y_i) + f_i(x) \times (1 - peak(f_i(x), y_i))$, where $betw(y_i(x), c_i) = r_i \times y_i + (1 - r_i) \times c_i$.

The following variants, HR 10^k%C, differ from the previous ones in the value of r_i. In the original variant, r_i is a random number in [0,1]. The HR 10^k%C variants have $r_i = 0$, and so, the *moving point* moves to $C = (c_1, ..., c_d)$ - the centroid. The *moving point* is, for sure, assigned with a new class (unless the chosen centroid is the centroid of the class to which it was already assigned to).

	HRT	**IM-3**	**WAV**	**SEG**	**IM-10**	**YST**	**VOW**	**M-L**
Training accuracy								
M3GP 50	93.1(3.2)	98.7(0.6)	89.6(0.6)	**96.4**(0.7)	**92.3**(0.8)	66.0(5.0)	**100**(0.0)	**100**(0.0)
HR 10^{-1}%C	**97.4**(1.9)	**100**(0.7)	**91.3**(0.4)	**96.3**(0.6)	91.6(0.8)	**67.7**(2.1)	**100**(0.0)	**100**(0.6)
HR 10^{-2}%C	94.7(1.7)	99.3(0.8)	**91.1**(0.5)	**96.4**(0.5)	**92.0**(0.8)	66.0(1.5)	**100**(0.0)	**100**(0.8)
HR 10^{-3}%C	93.7(2.8)	98.7(0.7)	90.9(0.5)	**96.5**(0.5)	**92.5**(1.0)	65.6(3.3)	**100**(0.0)	**100**(0.8)
HR 10^{-4}%C	93.1(2.5)	98.7(0.7)	90.7(0.5)	**96.4**(0.7)	**92.3**(1.0)	65.6(2.4)	**100**(0.1)	**100**(0.4)
HR 10^{-5}%C	93.7(2.1)	99.1(1.1)	90.6(0.5)	**96.6**(0.7)	**92.3**(1.2)	65.1(2.2)	**100**(0.0)	**100**(0.3)
HR 10^{-6}%C	94.2(1.5)	98.7(0.7)	90.8(0.4)	**96.7**(0.7)	**92.0**(0.7)	65.0(2.3)	**100**(0.0)	**100**(0.4)
Test accuracy								
M3GP 50	**77.8**(4.0)	**94.3**(2.6)	**84.7**(0.9)	**95.1**(1.0)	**91.1**(0.9)	**55.6**(3.8)	**94.3**(2.3)	**60.6**(5.9)
HR 10^{-1}%C	**77.8**(7.8)	93.3(3.1)	83.1(0.8)	**94.7**(1.0)	**90.6**(0.9)	**56.7**(2.7)	83.2(10.4)	56.5(7.6)
HR 10^{-2}%C	**78.4**(5.3)	**93.8**(2.5)	**84.0**(0.8)	**95.2**(0.8)	**90.4**(1.1)	**55.6**(2.1)	92.6(2.0)	**63.0**(7.6)
HR 10^{-3}%C	**79.0**(3.7)	93.8(2.4)	**84.2**(0.8)	**95.0**(1.2)	**91.0**(1.2)	**56.2**(2.4)	**92.6**(2.1)	**62.0**(7.1)
HR 10^{-4}%C	**77.8**(4.8)	**94.8**(1.7)	**84.1**(1.1)	**94.7**(1.1)	**91.0**(0.9)	**55.5**(2.7)	**93.4**(2.3)	**62.5**(4.9)
HR 10^{-5}%C	**79.6**(4.9)	92.8(3.3)	**84.1**(0.9)	**95.1**(1.0)	**90.9**(1.3)	**55.2**(2.3)	**93.4**(2.7)	**61.1**(7.8)
HR 10^{-6}%C	**77.8**(4.2)	**93.8**(2.4)	**84.0**(0.9)	**95.2**(0.9)	**91.0**(0.9)	**56.1**(2.3)	**93.6**(2.4)	**62.0**(6.0)
# of dimensions								
M3GP 50	14(1-19)	5.5(2-10)	26.5(20-32)	8(5-14)	15(8-24)	12(2-17)	22(19-25)	11(8-12)
HR 10^{-1}%C	20.5(9-31)	8(3-20)	35.5(28-48)	9(5-18)	14(7-21)	13.5(7-19)	24(26-38)	11(9-12)
HR 10^{-2}%C	16(8-23)	7(3-13)	32(23-39)	9(6-20)	13(6-17)	13(9-17)	24(20-28)	10(8-13)
HR 10^{-3}%C	15.5(1-23)	8(4-13)	33(25-39)	9.5(4-17)	14(5-22)	14(4-20)	22(18-28)	11(8-13)
HR 10^{-4}%C	15(5-22)	7.5(4-15)	32(23-36)	10(5-18)	15.5(6-23)	14(8-21)	22(16-24)	11(9-12)
HR 10^{-5}%C	15(7-22)	9(2-17)	33(25-38)	12(5-20)	17.5(7-25)	14(6-20)	22.5(18-26)	11(9-14)
HR 10^{-6}%C	16(8-22)	8(4-12)	33(26-39)	11(8-16)	16(9-24)	13(8-22)	22.5(18-27)	11(8-12)

Table 5.16: Comparison between the M3GP and GSI-M3GP hyperrectangle variants with *pred*% equal to 10^k%, with $k \in \{1, 0, -1, ..., -6\}$, and $r_i = 0$.

	HRT	**IM-3**	**WAV**	**SEG**	**IM-10**	**YST**	**VOW**	**M-L**
HR 10^{-1}%C	387.5 (263-552)	328 (163-547)	217 (133-322)	147 (36-266)	38.5 (6-217)	314.5 (195-517)	263.5 (227-344)	440 (349-580)
HR 10^{-2}%C	496.5 (406-580)	531 (434-624)	309 (168-406)	409.5 (211-532)	249.5 (174-538)	492.5 (419-609)	396 (336-499)	580 (363-623)
HR 10^{-3}%C	510 (441-678)	562.5 (407-636)	378 (323-470)	525 (427-569)	451 (364-615)	520.5 (454-657)	423.5 (305-540)	531 (342-616)
HR 10^{-4}%C	527.5 (441-615)	559.5 (424-629)	409.5 (364-497)	525 (434-616)	505.5 (394-623)	520.5 (431-619)	421 (317-553)	531.5 (440-608)
HR 10^{-5}%C	513.5 (400-594)	549 (391-678)	392 (336-455)	515 (427-588)	499.5 (401-636)	531 (412-580)	416.5 (361-492)	552.5 (357-615)
HR 10^{-6}%C	511.5 (393-587)	561. 5(465-643)	395.5 (329-490)	500 (421-589)	484 (412-573)	527.5 (433-636)	409.5 (324-505)	553 (399-629)

Table 5.17: Comparing depth values between GSI-M3GP hyperrectangle variants with *pred*% equal to 10^k%, with $k \in \{1, 0, -1, ..., -6\}$, and $r_i = 0$.

HR 10^k%C variants of GSI-M3GP are able to produce individuals with higher median training accuracy than M3GP. Regarding test accuracy, most of these hyperrectangle variants are able to produce median test accuracies which are not significantly different from M3GP median test accuracies (see table 5.16). However, for some of the predefined percentage values, these variants were not competitive with M3GP on IM-3, VOW and M-L datasets. Notice that, for these variants, the values of *pred*% equal to 1% and 10% were not ran.

The behaviour of the depth values is the same for HR 10^k%C and HR 10^k% (see tables 5.17 and 5.13).

Although the values of *mutation percentage* are increasing as the predefined percentage decreases (see table 5.18), the values in variant HR 10^{-1}%C are lower than the values in variant HR 10^{-1}% (see table 5.14). This might imply that, when *pred*% $= 10^{-1}$%, the mutation operator works better if r_i is chosen at random.

		HRT	**IM-3**	**WAV**	**SEG**	**IM-10**	**YST**	**VOW**	**M-L**
HR 10^{-1}%C	*M*	59%	54%	36%	32%	13%	63%	39%	66%
	A	33%	29%	59%	48%	65%	30%	49%	22%
	R	8%	17%	5%	20%	22%	7%	12%	12%
HR 10^{-2}%C	*M*	71%	78%	50%	67.8%	56%	78%	58%	78%
	A	23%	15%	47%	20.6%	35%	19%	34%	16%
	R	5%	7%	3%	8.6%	9%	3%	8%	6%
HR 10^{-3}%C	*M*	73%	80%	56%	76.3%	71%	78%	61%	74.81%
	A	22%	14%	42%	18.3%	24%	19%	31%	17.65%
	R	5%	6%	2%	5.4%	5%	3%	8%	7.54%
HR 10^{-4}%C	*M*	75%	78%	59.8%	77%	74%	76%	61%	77%
	A	20%	15%	38.5%	18%	23%	20%	31%	16%
	R	5%	7%	1.7%	5%	3%	4%	8%	7%
HR 10^{-5}%C	*M*	72.467%	78%	58%	74%	72%	76.6%	60%	76%
	A	22.255%	15%	40%	20%	24%	19.6%	32%	17%
	R	5.378%	7%	2%	6%	4%	3.8%	8%	7%
HR 10^{-6}%C	*M*	73%	81%	58%	74%	72%	76%	59%	76.6668%
	A	22%	14%	40%	20%	24%	20%	32%	16.6655%
	R	5%	5%	2%	6%	4%	4%	9%	6.6677%

Table 5.18: Comparing operators' percentages between the GSI-M3GP hyperrectangle variants with *pred*% equal to 10^k%, with $k \in \{1, 0, -1, ..., -6\}$, and $r_i = 0$.

As for the results in table 5.15, the number of operations is higher than 90 for the hyperrectangle variants with predefined percentage smaller than 0.001% and setting $r_i = 0$ (see table 5.19).

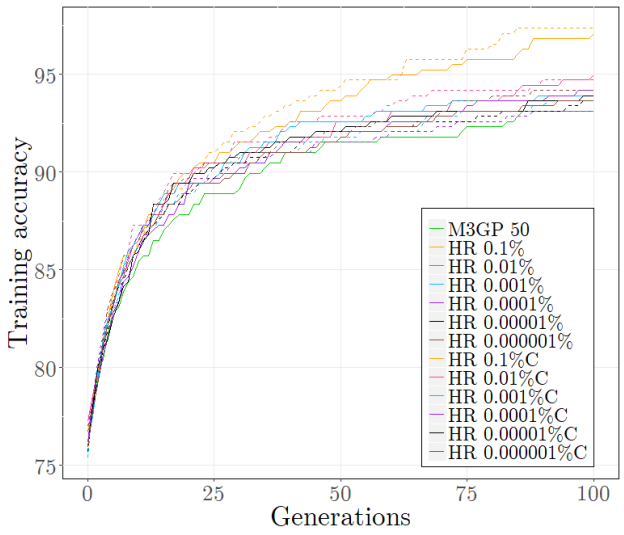
	HRT	**IM-3**	**WAV**	**SEG**	**IM-10**	**YST**	**VOW**	**M-L**
HR 0.1%C	93.8 (83-100)	93.2 (74-100)	85.1 (71-99)	67.8 (48-83)	61.4 (37-83)	72.3 (43-89)	96.7 (78-100)	95.4 (73-100)
HR 0.01%C	98.4 (96-100)	98.7 (95-100)	89.1 (76-99)	87.2 (65-97)	69.7 (58-91)	89.7 (78-99)	98.5 (95-100)	98.9 (95-100)
HR 0.001%C	98.7 (95-100)	99.0 (94-100)	96.5 (90-100)	96.9 (92-100)	90.7 (82-97)	96.0 (91-99)	99.5 (98-100)	99.0 (95-100)
HR 0.0001%C	98.8 (97-100)	99.2 (93-100)	97.8 (94-100)	98.0 (94-100)	96.1 (91-100)	97.4 (93-100)	99.5 (97-100)	99.1 (96-100)
HR 0.00001%C	99.1 (94-100)	99.5 (98-100)	98.1 (96-100)	98.0 (94-100)	97.3 (92-100)	97.6 (95-100)	99.1 (97-100)	99.3 (97-100)
HR 0.000001%C	98.5 (94-100)	98.9 (94-100)	98.5 (95-100)	97.9 (94-100)	97.2 (89-100)	97.9 (94-100)	99.0 (96-100)	99.3 (95-100)

Table 5.19: Comparing the number of operations between the GSI-M3GP hyperrectangle variants with *pred*% equal to 10^k%, with $k \in \{1, 0, -1, ..., -6\}$, and $r_i = 0$.

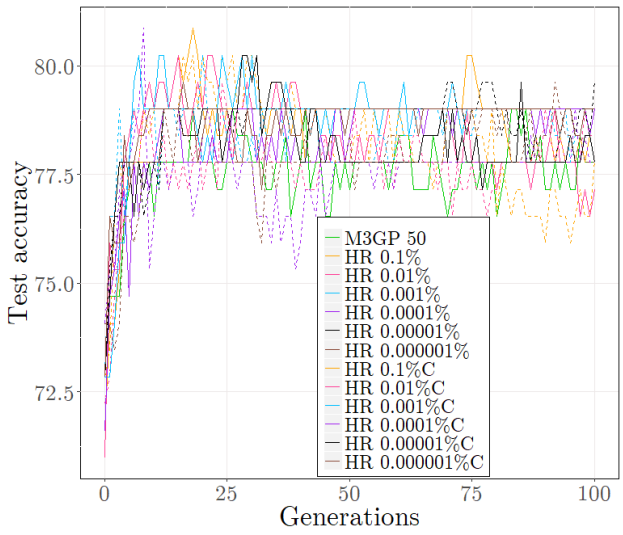
Figures 5.12, 5.13 and 5.14 show the evolution of training and test accuracies of M3GP, HR 10^k% and HR 10^k%C, with $k \in \{-1, ..., -6\}$, as the number of generations increases. HR 10% and HR 1% are not presented since they overfit on all problems. The *dashed lines* represent the variants HR 10^k%C (*i.e.* the variants for which $r_i = 0$).

HR 10^{-4}%, HR 10^{-5}% and HR 10^{-6}% are the best approaches for IM-3 regarding test accuracy (see figure 5.12d). HR 10^{-1}%C is the best approach both on YST train and test (see figure 5.13f). HR 10^{-1}%C overfits on WAV and VOW (see figures 5.12e,f and 5.14a,b). HR 10^{-1}% also overfits on VOW. All the variants behave similary both on training and test sets for SEG and IM-10 (see figures 5.13a,b and 5.13c). All variants sliglty overfit on M-L (compare figures 5.14c and 5.14d).

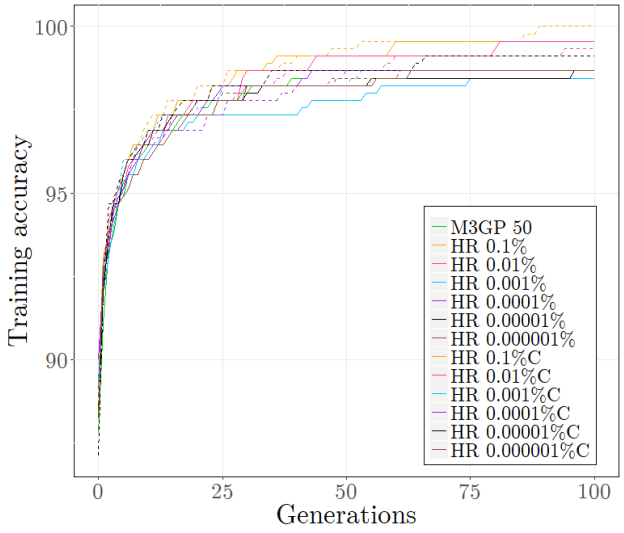
The behaviour of M3GP, HR 10^k% and HR 10^k%C, with $k \in \{-1, ..., -6\}$, on the last generation, for the test set, can be seen in figures 5.15, 5.16 and 5.17.



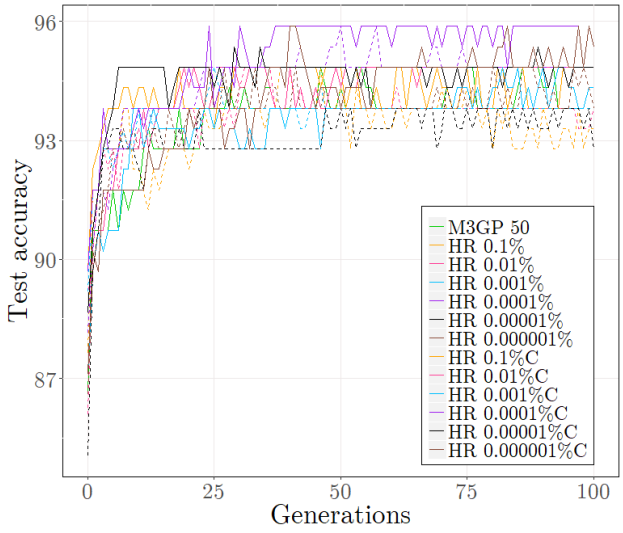
(a) HRT - Training.



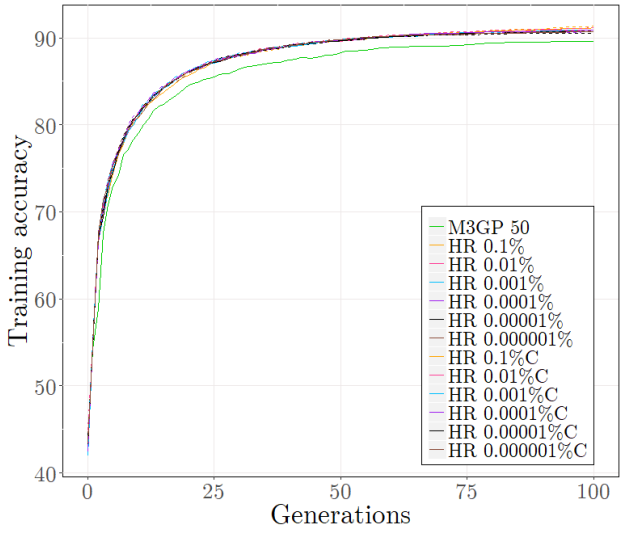
(b) HRT - Test.



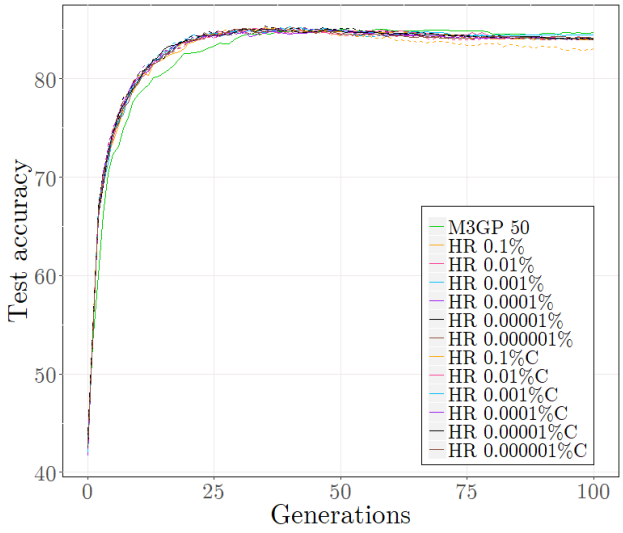
(c) IM-3 - Training.



(d) IM-3 - Test.

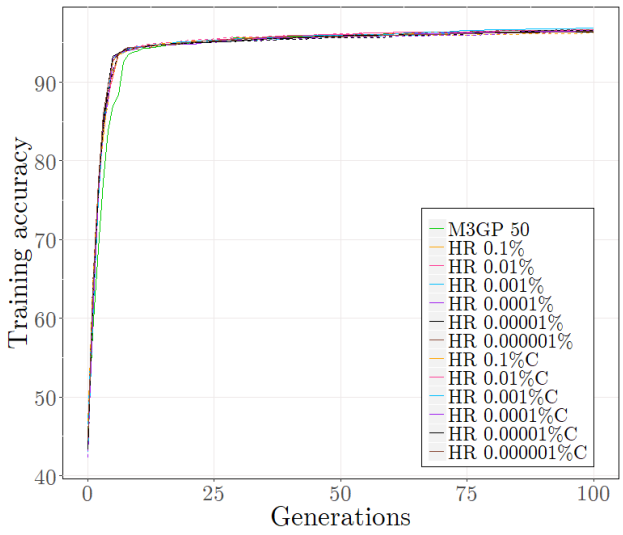


(e) WAV - Training.

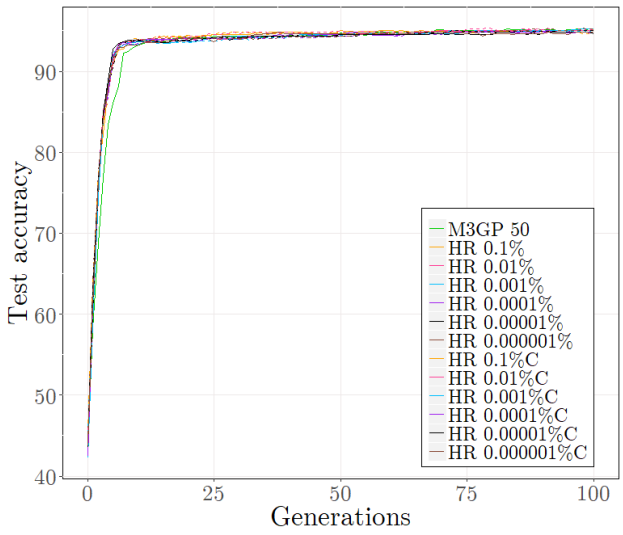


(f) WAV - Test.

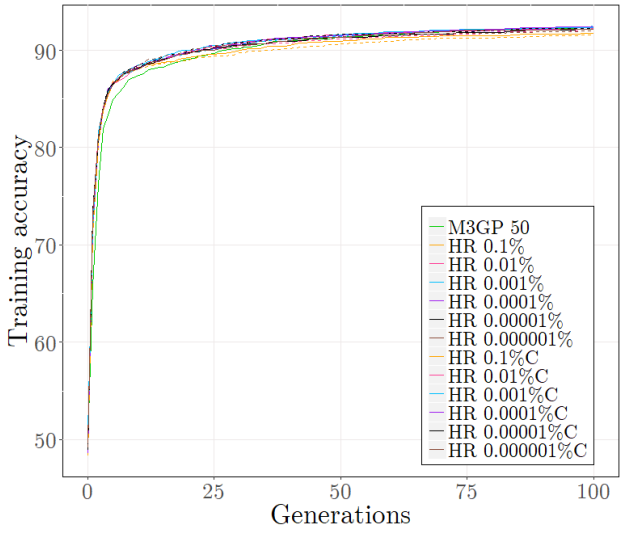
Figure 5.12: Comparison between median accuracies of M3GP, HR $10^k\%$ and HR $10^k\%C$ (dashed lines) algorithms, with $k \in \{-1, ..., -6\}$, on HRT, IM-3 and WAV datasets.



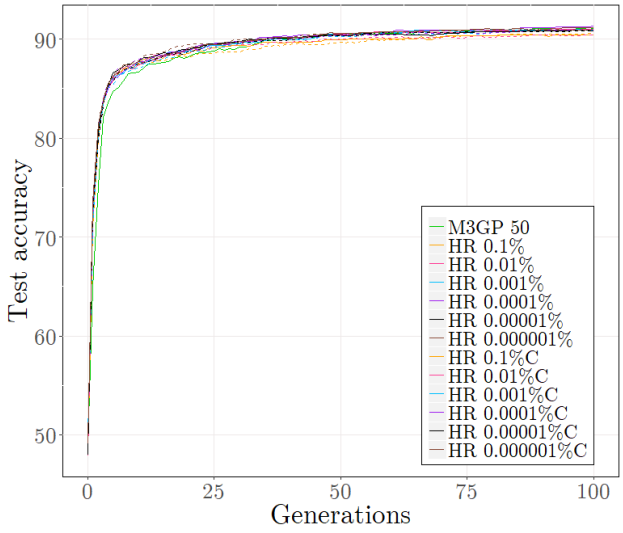
(a) SEG - Training.



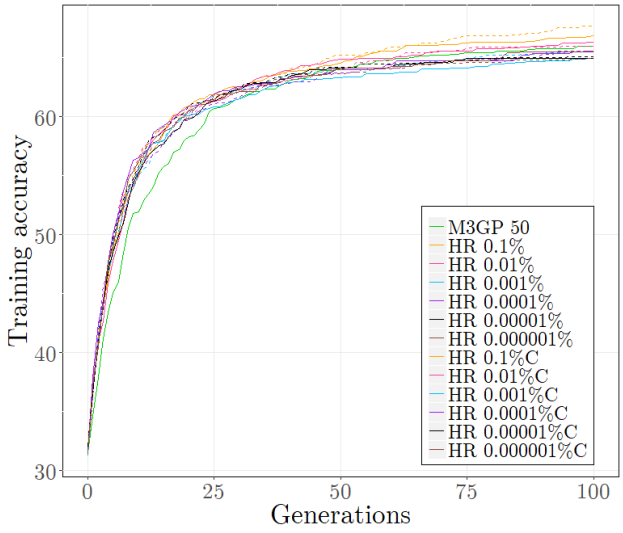
(b) SEG - Test.



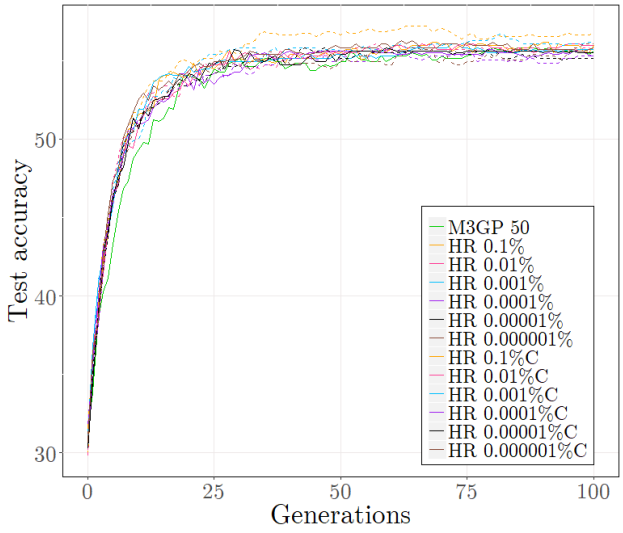
(c) IM-10 - Training.



(d) IM-10 - Test.

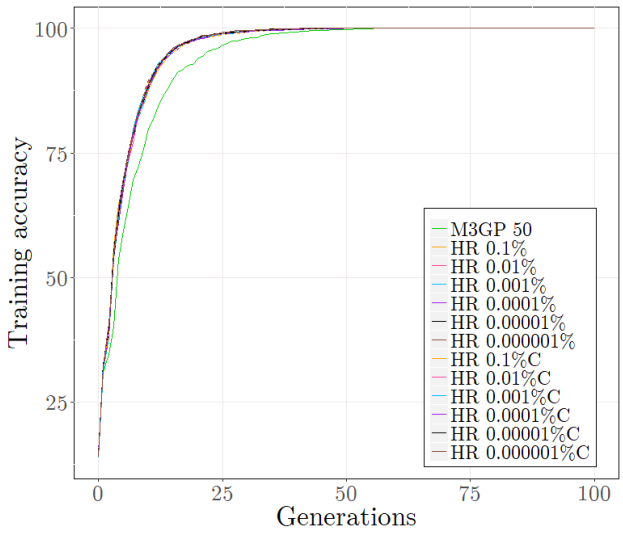


(e) YST - Training.

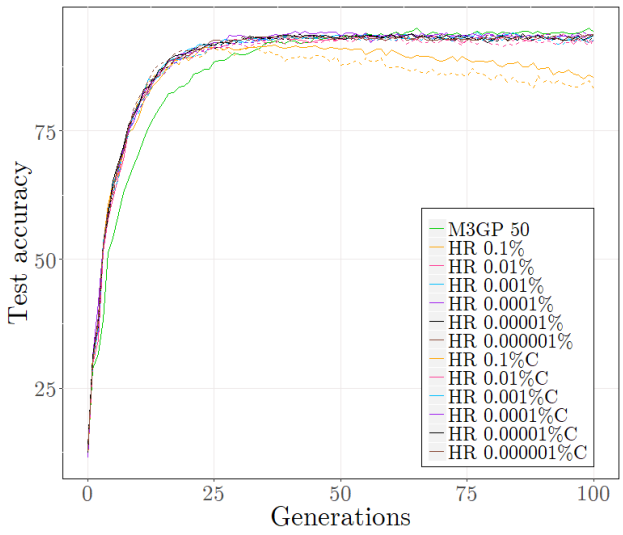


(f) YST - Test.

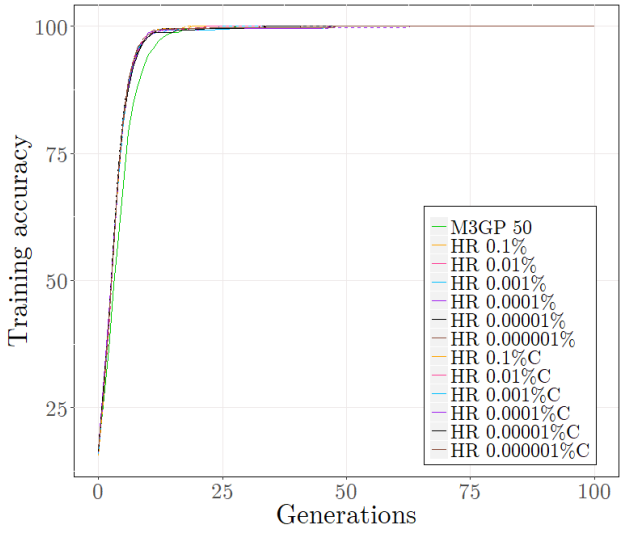
Figure 5.13: Comparison between median accuracies of M3GP, HR $10^k\%$ and HR $10^k\%C$ (dashed lines) algorithms, with $k \in \{-1, ..., -6\}$, on SEG, IM-10 and YST datasets.



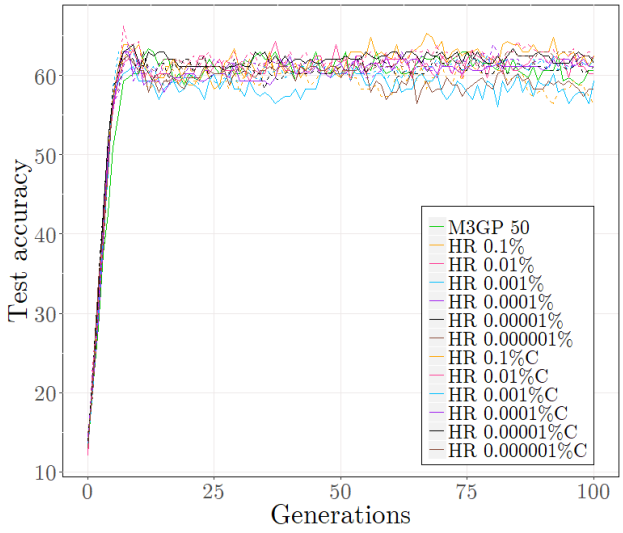
(a) VOW - Training.



(b) VOW - Test.



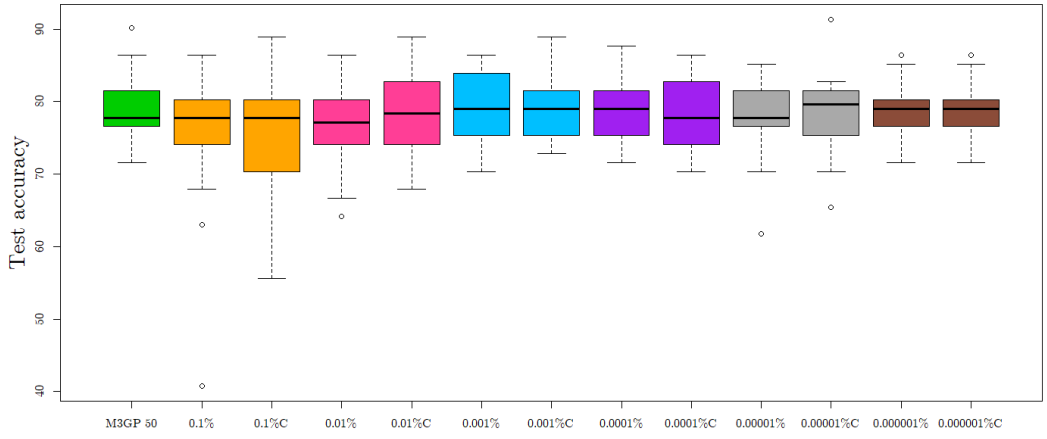
(c) M-L - Training.



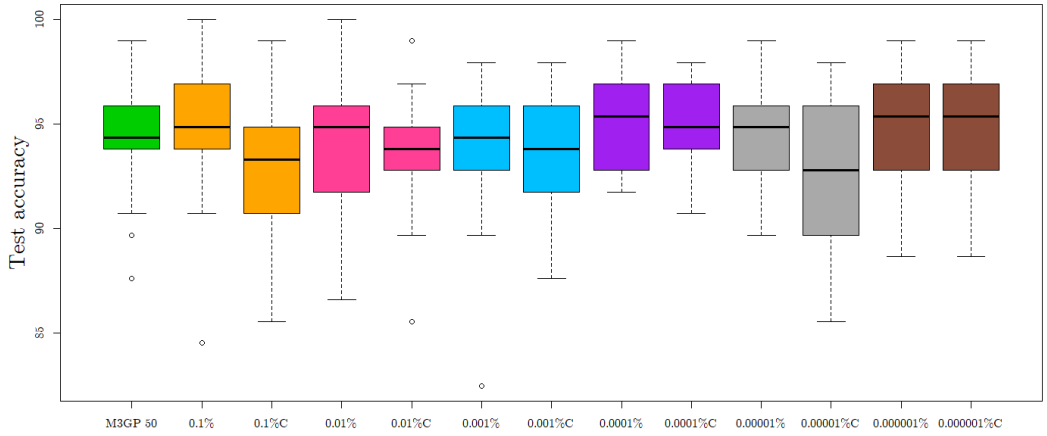
(d) M-L - Test.

Figure 5.14: Comparison between median accuracies of M3GP, HR 10^k% and HR 10^k%C (dashed lines) algorithms, with $k \in \{-1, ..., -6\}$, on VOW and M-L datasets.

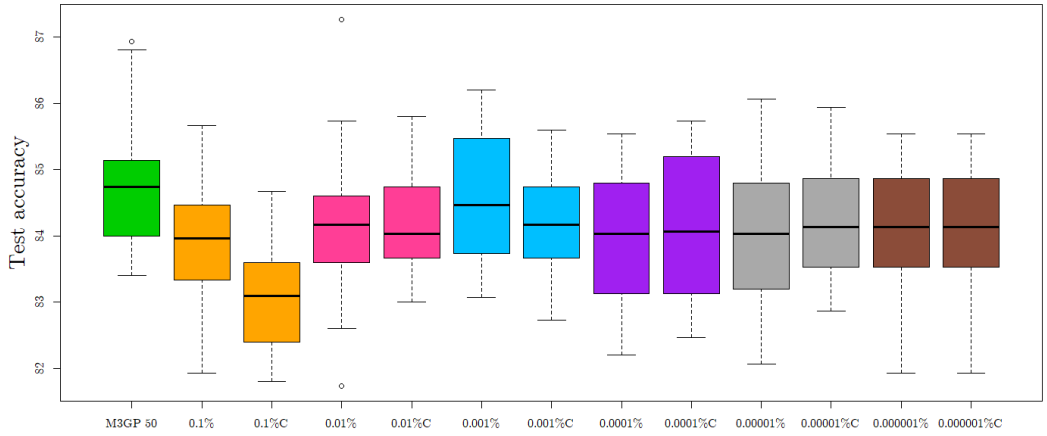
On HRT, all the algorithms seem to perform the same way on the last generation except for HR 10^{-1}%C (although having approximately the same median as the others, the range of values for HR 10^{-1}%C is much bigger; see figure 5.15a). On IM-3, the HR 10^k% variants seem to perform slightly better than the HR 10^k%C variants (see figure 5.15b). M3GP is the best approach on WAV (as in figure 5.15c). The range of test values is wider on HR 10^{-3}%C than for the remaining algorithms, on the SEG dataset. Also, variant HR 10^{-2}%C produces solutions for which, both the minimum and maximum test accuracies are higher than the minimum and maximum test accuracies reached on M3GP (see figure 5.16a).



(a) HRT - Test.

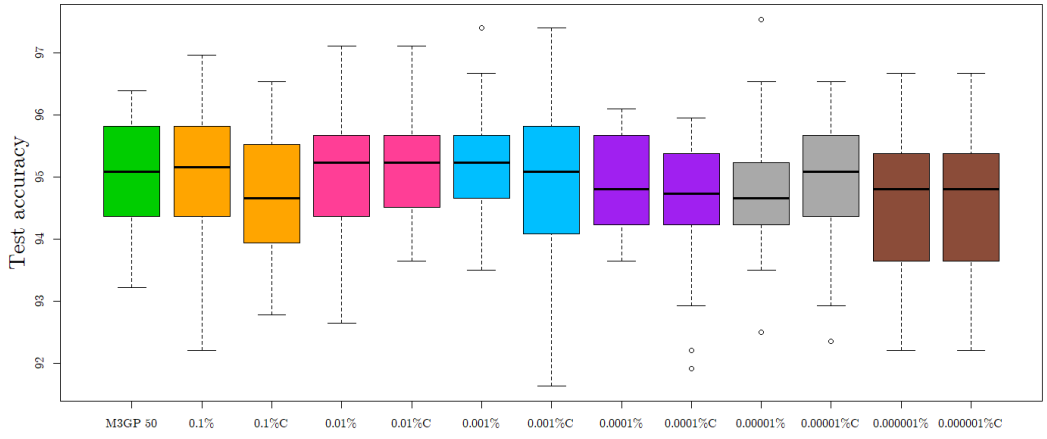


(b) IM-3 - Test.

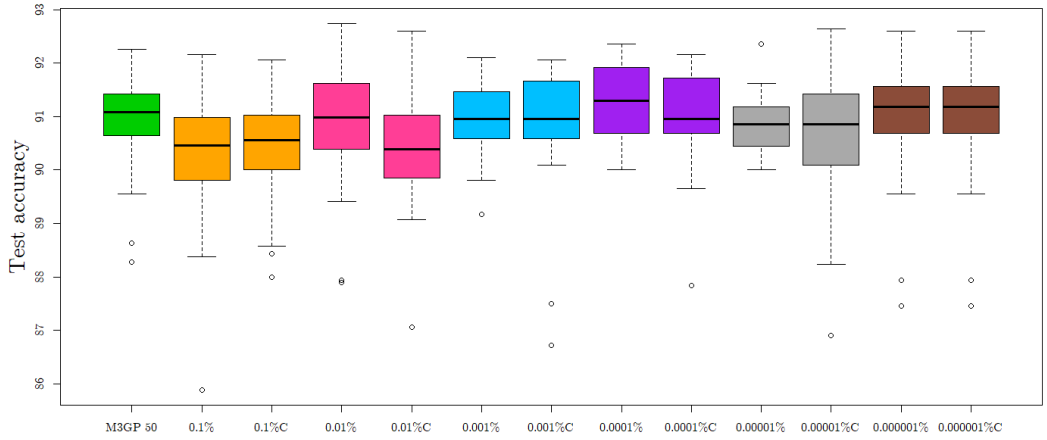


(c) WAV - Test.

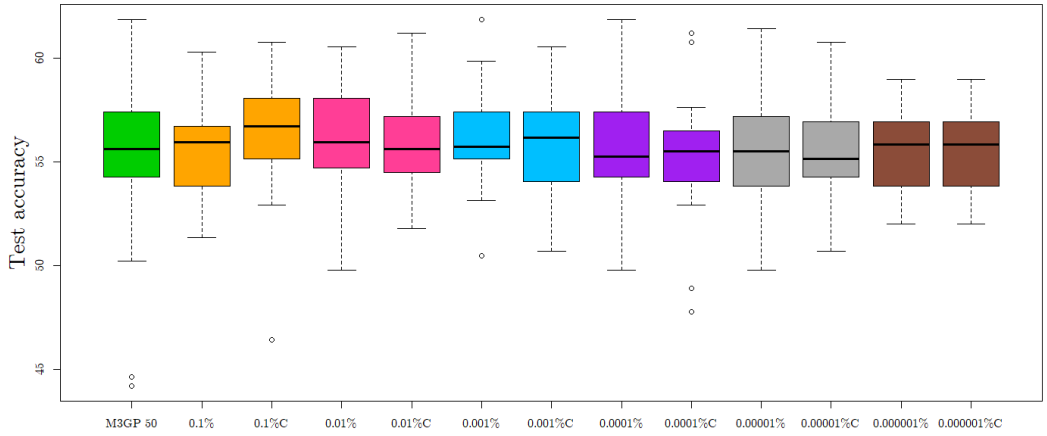
Figure 5.15: Comparison between last generation individuals' accuracies of M3GP, HR $10^k\%$ and HR $10^k\%C$ algorithms, with $k \in \{-1, ..., -6\}$, on HRT, IM-3 and WAV datasets.



(a) SEG - Test.

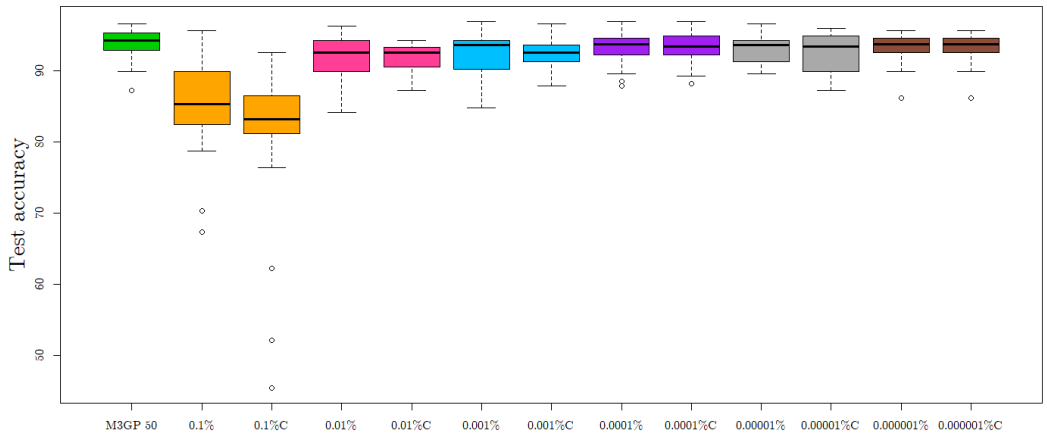


(b) IM-10 - Test.

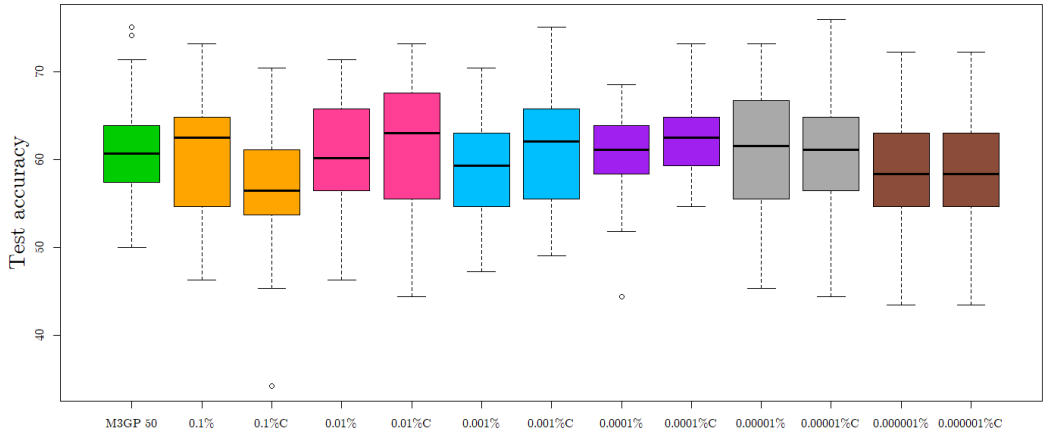


(c) YST - Test.

Figure 5.16: Comparison between last generation individuals' accuracies of M3GP, HR $10^k\%$ and HR $10^k\%C$ algorithms, with $k \in \{-1, ..., -6\}$, on SEG, IM-10 and YST datasets.



(a) VOW - Test.



(b) M-L - Test.

Figure 5.17: Comparison between last generation individuals' accuracies of M3GP, HR $10^k\%$ and HR $10^k\%C$ algorithms, with $k \in \{-1, ..., -6\}$, on WAV and M-L datasets.

On IM-10 and YST all the algorithms seem to perform similarly (see figures 5.16b and 5.16b). On VOW only the variants HR 0.1% and HR 0.1%C have a much lower median test accuracies than the the remaining ones, although all the hyperrectangle variants produce a median test accuracy lower than the one reached by M3GP (see figure 5.17a). On M-L, the variants which are able to achieve higher median values of test accuracy, also produce much lower test accuracies - *i.e.* the range of values is quite big (see figure 5.17b).

Randomly generating the *predefined percentage:*

The *predefined percentage* is a parameter of the GSI-M3GP algorithm. However, setting it to a specific value might not be the best approach. It would be better to have it being chosen at random on a specific interval. Also, looking at values in table 5.12 it seems that lower values of *pred*% improve the algorithm's test accuracy. As such, 3 intervals were considered and tested:]0,0.1]%,]0,0.01]% and]0,0.001]%. For example, if using the first one: each time the gsi-mutation operator is used, the predefined percentage is randomly generated from]0,0.1]%.

The results of these variants are stored in table 5.20. Regarding test accuracy, all the variants are competitive with M3GP (except for HR]0,0.1]% on IM-10 and VOW). The number of dimensions in the hyperrectangle variants is not much higher than the number of dimensions for M3GP.

	HRT	**IM-3**	**WAV**	**SEG**	**IM-10**	**YST**	**VOW**	**M-L**
Training accuracy								
M3GP 50	93.1(3.2)	98.7(0.6)	89.6(0.6)	**96.4**(0.7)	**92.3**(0.8)	66.0(5.0)	**100**(0.0)	**100**(0.0)
HR]0,0.1]%	**96.6**(2.4)	**99.6**(0.8)	**90.8**(0.5)	**96.6**(0.5)	91.8(0.5)	**67.1**(3.3)	**100**(0.0)	**100**(0.1)
HR]0,0.01]%	94.2(1.8)	99.1(0.8)	**90.8**(0.7)	**96.6**(0.5)	**92.3**(0.6)	**66.3**(1.4)	**100**(0.0)	**100**(0.5)
HR]0,0.001]%	93.9(2.1)	98.9(0.9)	**90.6**(0.5)	**96.4**(0.6)	**92.5**(0.8)	65.2(2.3)	**100**(0.0)	**100**(1.0)
HR]0,0.01]%C	94.2(3.5)	99.1(0.9)	**90.9**(0.5)	**96.6**(0.6)	**92.3**(0.8)	65.1(2.6)	**100**(0.0)	**100**(0.6)
HR]0,0.001]%C	94.2(2.4)	98.7(1.1)	**90.6**(0.6)	**96.5**(0.6)	**92.4**(0.7)	**66.0**(1.2)	**100**(0.0)	**100**(0.5)
Test accuracy								
M3GP 50	**77.8**(4.0)	**94.3**(2.6)	**84.7**(0.9)	**95.1**(1.0)	**91.1**(0.9)	**55.6**(3.8)	**94.3**(2.3)	**60.6**(5.9)
HR]0,0.1]%	**79.0**(8.2)	**92.8**(2.4)	**84.1**(1.0)	**94.9**(1.3)	90.4(0.9)	**57.0**(2.3)	87.9(4.0)	**60.2**(5.5)
HR]0,0.01]%	**80.2**(3.8)	**93.8**(2.6)	**84.5**(0.8)	**95.5**(0.9)	**91.2**(0.8)	**55.4**(2.3)	**93.6**(2.4)	**60.2**(5.7)
HR]0,0.001]%	**79.0**(4.1)	**94.3**(2.8)	**84.5**(0.8)	**95.2**(1.1)	**91.1**(0.9)	**56.3**(2.4)	**93.8**(1.7)	**59.3**(6.6)
HR]0,0.01]%C	**78.4**(4.9)	**94.8**(2.3)	**84.1**(0.8)	**95.2**(0.7)	**91.0**(0.9)	**55.6**(2.8)	92.3(2.8)	**60.2**(5.0)
HR]0,0.001]%C	**76.5**(4.4)	**93.3**(2.9)	**84.4**(1.0)	**95.1**(0.9)	**91.2**(0.6)	**56.4**(2.4)	**92.4**(2.2)	**62.0**(6.9)
# of dimensions								
M3GP 50	14(1-19)	5.5(2-10)	26.5(20-32)	8(5-14)	15(8-24)	12(2-17)	22(19-25)	11(8-12)
HR]0,0.1]%	18(7-25)	7.5(2-16)	31(24-36)	9(5-17)	14(11-21)	13(11-20)	29(24-34)	11(9-12)
HR]0,0.01]%	17(8-21)	9(4-16)	31(21-36)	10(6-15)	15(9-20)	14(10-20)	23(18-28)	11(8-13)
HR]0,0.001]%	16.5(7-21)	9(4-15)	32(25-39)	10(7-16)	16(10-22)	14(9-21)	22(18-28)	11(7-13)
HR]0,0.01]%C	15(2-23)	6(3-14)	32.5(22-37)	9(6-20)	14(7-22)	14(9-26)	22(19-26)	11(7-13)
HR]0,0.001]%C	16(5-22)	9(3-18)	33(21-38)	10(6-14)	16(10-21)	14(10-21)	21.5(18-28)	11(9-13)

Table 5.20: Comparison between the M3GP and GSI-M3GP hyperrectangle variants with $pred\% \in]0, 10^k]\%$, with $k \in \{-1, -2, -3\}$.

Table 5.21 stores the median, maximum and minimum depth on the 30 runs. Looking attentively and comparing these values to the ones in tables 5.13 and 5.17, these new values are not much higher. As such, even though here the predefined percentage might be a very low

value for a specific usage of gsi-mutation operator, the depth does not increase indefinitely.

	HRT	**IM-3**	**WAV**	**SEG**	**IM-10**	**YST**	**VOW**	**M-L**
HR]0,0.1]%	561.5 (337-559)	496 (288-608)	290.5 (154-406)	294 (154-449)	171.5 (65-288)	440 (346-489)	329 (252-434)	499.5 (357-622)
HR]0,0.01]%	507 (421-601)	573 (295-643)	371.5 (315-427)	493.5 (400-588)	368 (223-496)	513.5 (440-505)	409 (336-505)	553.5 (336-629)
HR]0,0.001]%	497.5 (435-608)	539 (426-671)	392 (336-434)	545.5 (455-602)	477 (399-587)	517 (419-594)	410 (325-535)	560.5 (356-643)
HR]0,0.01]%C	518.5 (406-657)	566 (429-643)	343 (238-421)	494 (385-581)	321 (184-476)	506.5 (363-608)	422 (347-525)	546 (314-629)
HR]0,0.001]%C	511.5 (434-608)	561 (427-636)	381.5 (329-511)	536 (434-616)	483.5 (381-559)	510 (385-576)	437.5 (343-532)	541.5 (378-622)

Table 5.21: Comparing depth values between GSI-M3GP hyperrectangle variants with $pred\% \in]0, 10^k]\%$, with $k \in \{-1, -2, -3\}$.

The operators' percentage values (see table 5.22) are roughly around the same values as in tables 5.14 and 5.18. Also, the *mutation percentages* are bigger than the *add branch percentages* (except for HR]0,0.1]% on WAV and IM-10).

		HRT	**IM-3**	**WAV**	**SEG**	**IM-10**	**YST**	**VOW**	**M-L**
HR]0,0.1]%	*M*	67%	70%	48%	53%	37%	72%	49.1%	74%
	A	26%	19%	48%	33%	48%	23%	40.5%	18%
	R	7%	11%	4%	14%	15%	5%	10.4%	8%
HR]0,0.01]%	*M*	73.8%	80%	56%	75%	63%	76.5%	59%	75.6%
	A	21.7%	14%	41%	19%	30%	20.2%	32%	17.8%
	R	4.5%	6%	3%	6%	7%	3.3%	9%	7.6%
HR]0,0.001]%	*M*	73%	77%	57%	78%	72%	76%	60.3%	78%
	A	22%	16%	41%	17%	24%	20%	31.4%	16%
	R	5%	7%	2%	5%	4%	4%	8.3%	6%
HR]0,0.01]%C	*M*	75.7%	81%	53%	75.28%	60%	76%	61%	75.4%
	A	19.7%	13%	44%	18.46%	32%	20%	31%	17.4%
	R	4.6%	6%	3%	6.26%	8%	4%	8%	7.2%
HR]0,0.001]%C	*M*	75%	78%	57%	78%	71%	74%	63%	76%
	A	20%	15%	41%	17%	24%	21%	30%	17%
	R	5%	7%	2%	5%	5%	5%	7%	7%

Table 5.22: Comparing operators' percentages between the GSI-M3GP hyperrectangle variants with $pred\% \in]0, 10^k]\%$, with $k \in \{-1, -2, -3\}$.

The number of operations is always bigger than 90 on HR]0,0.001]% and HR]0,0.001]%C and so, for these two variants, there is improvement on almost all generations.

	HRT	IM-3	WAV	SEG	IM-10	YST	VOW	M-L
HR]0,0.1]%	97.3 (87-100)	97.2 (83-100)	85.9 (71-97)	79.2 (64-94)	69.1 (57-87)	84.8 (70-100)	98.4 (94-100)	97.8 (77-100)
HR]0,0.01]%	98.4 (94-100)	99.3 (97-100)	94.0 (83-99)	93.8 (87-100)	81.3 (66-91)	94.8 (88-100)	99.3 (97-100)	99.2 (92-100)
HR]0,0.001]%	98.4 (95-100)	99.2 (96-100)	97.2 (93-100)	97.8 (93-100)	94.7 (90-99)	97.2 (93-100)	99.3 (97-100)	99.6 (96-100)
HR]0,0.01]%C	98.4 (94-100)	98.4 (95-100)	91.2 (82-100)	93.5 (83-98)	78.7 (63-95)	94.1 (88-100)	98.9 (95-100)	99.5 (96-100)
HR]0,0.001]%C	99.0 (96-100)	99.3 (96-100)	97.3 (93-100)	97.4 (93-100)	94.5 (86-100)	96.6 (92-99)	98.7 (95-100)	99.5 (97-100)

Table 5.23: Comparing the number of operations between the GSI-M3GP hyperrectangle variants with $pred\% \in]0, 10^k]\%$, with $k \in \{-1, -2, -3\}$.

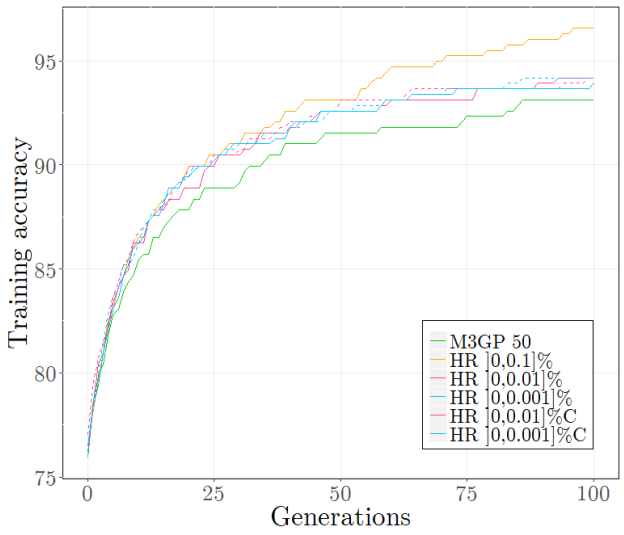
Figures 5.18, 5.19 and 5.20 show the evolution of training and test accuracies of M3GP, HR $]0, 10^k]\%$, with $k \in \{-1, -2, -3\}$, and HR $]0, 10^p]\%$C, with $p \in \{-2, -3\}$ as the number of generations increases.

HR]0,0.1]% overfits on IM-3 and VOW (see figures 5.18c,d and 5.20a,b).

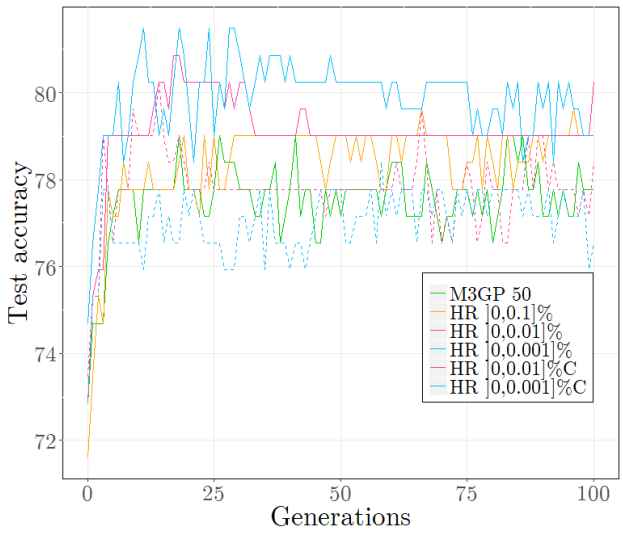
Regarding training accuracy: HR]0,0.1]% is the best approach on HRT and IM-3 datasets (see figures 5.18a and 5.18c); all the hyperrectangle variants behave similarly on WAV and better than M3GP (see figure 5.18e); M3GP and the hyperrectangle variants behave in the same way on SEG, IM-10, YST, VOW and M-L (see figures 5.19a,c,e and 5.20a,c).

Considering median test accuracy: HR]0,0.01]% and HR]0,0.001]% achieve the highest median test accuracies on HRT, as it can be seen in 5.18a; On IM-3, HR]0,0.001]%, HR]0,0.01]%C and M3GP are the approaches achieving the highest test accuracies, as expressed in figure 5.18c; HR]0,0.1]%, HR]0,0.001]% and HR]0,0.001]%C achieve the highest median test accuracies on YST (see figure 5.19e).

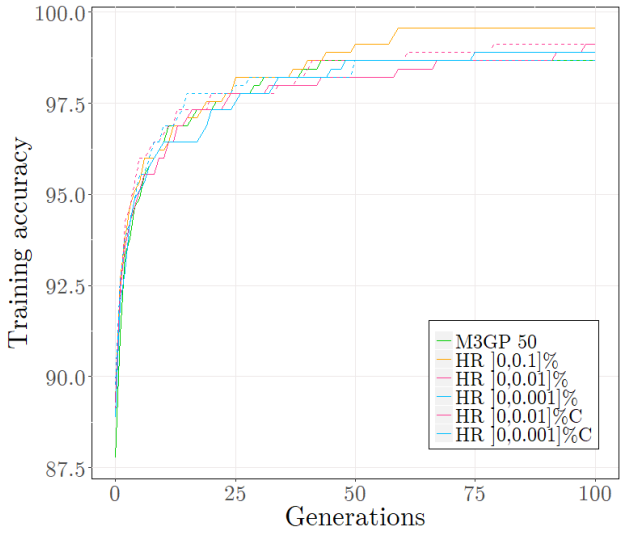
The behaviour of M3GP, HR $]0,10^k]\%$, with $k \in \{-1, -2, -3\}$, and HR $]0,10^p]\%$C, with $p \in \{-2, -3\}$, on the last generation, for the test set, can be seen in figures 5.21, 5.22 and 5.23.



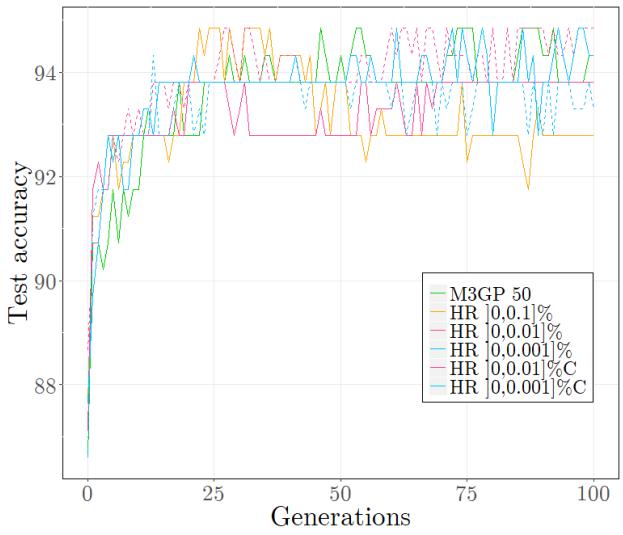
(a) HRT - Training.



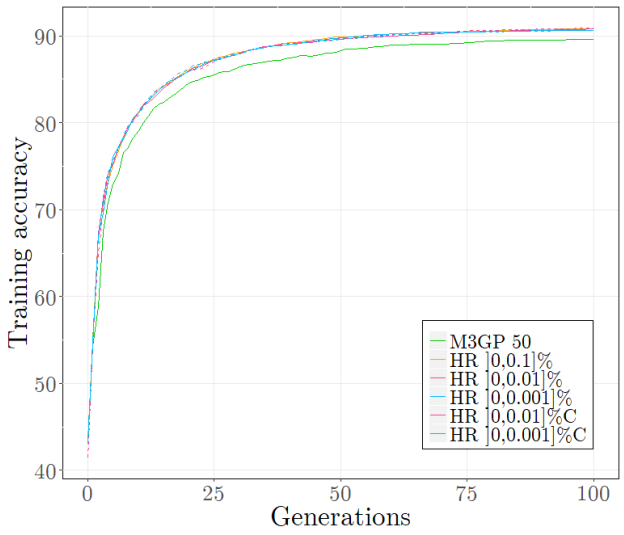
(b) HRT - Test.



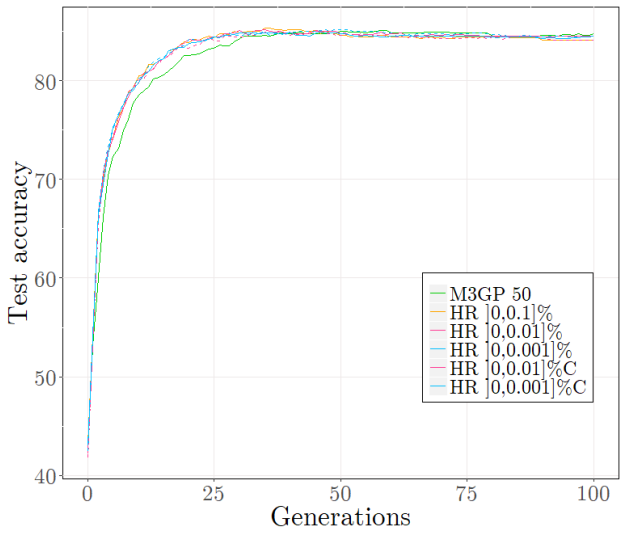
(c) IM-3 - Training.



(d) IM-3 - Test.

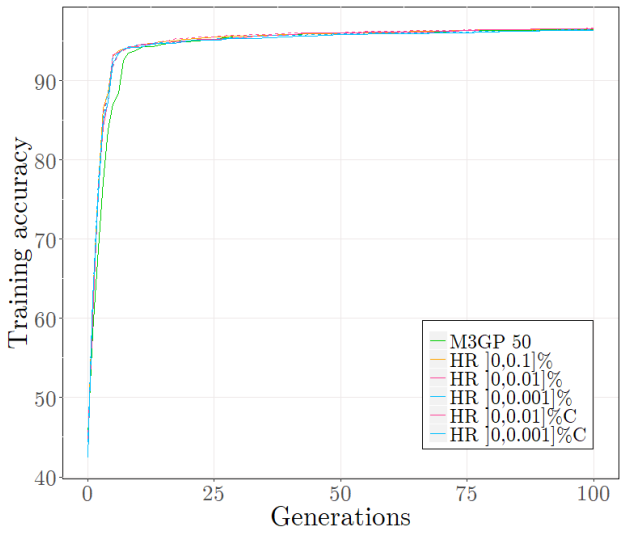


(e) WAV - Training.

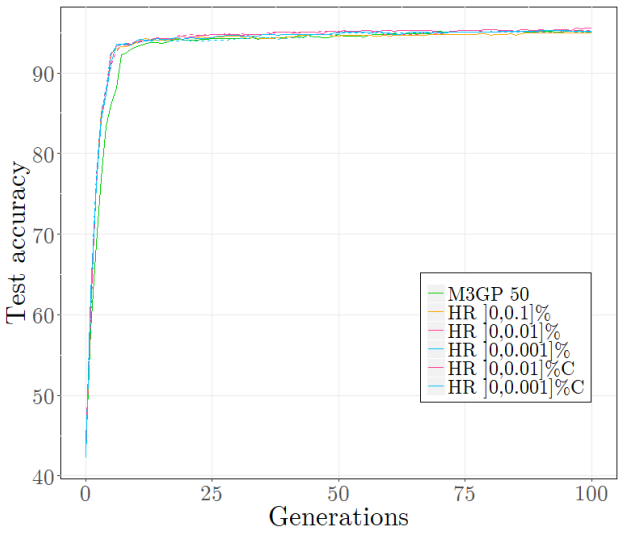


(f) WAV - Test.

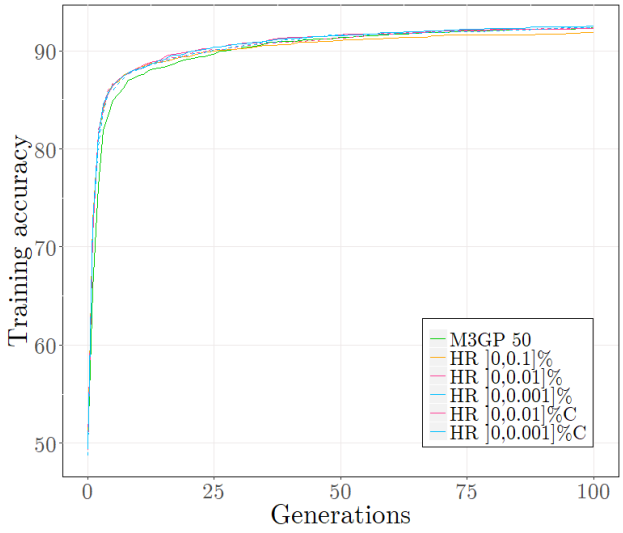
Figure 5.18: Comparison between median accuracies of M3GP and the hyperrectangle variants on HRT, IM-3 and WAV datasets.



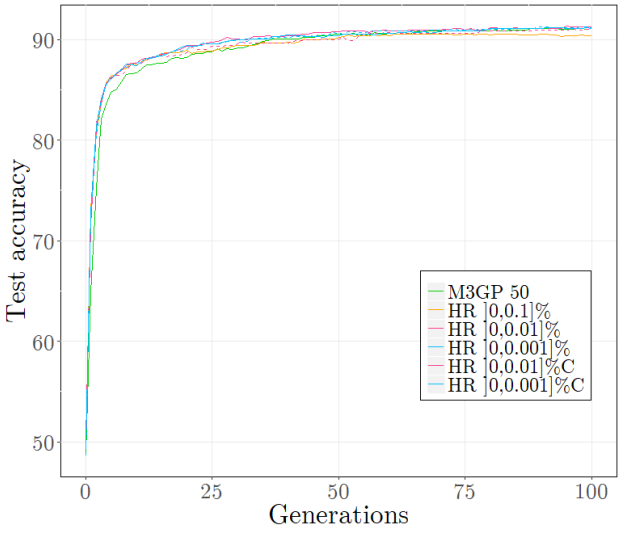
(a) SEG- Training.



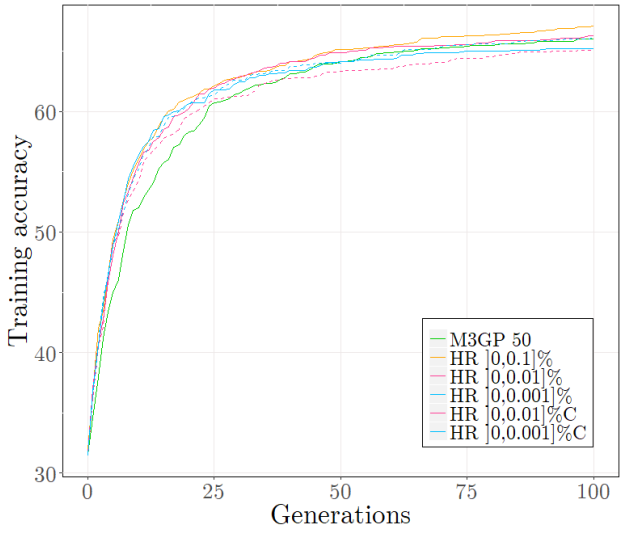
(b) SEG - Test.



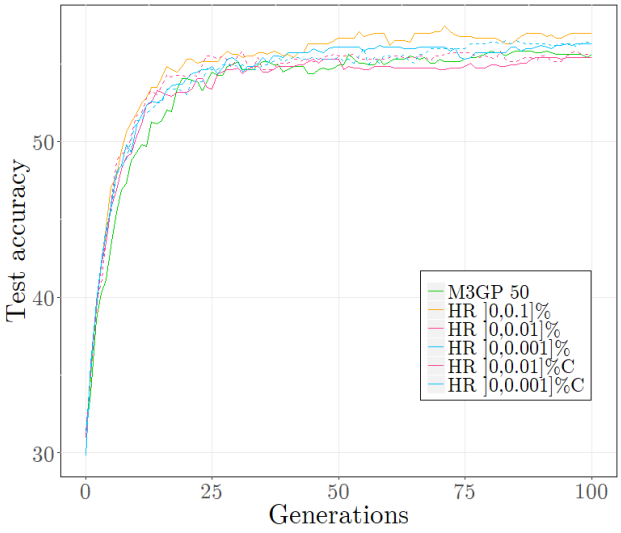
(c) IM-10 - Training.



(d) IM-10 - Test.

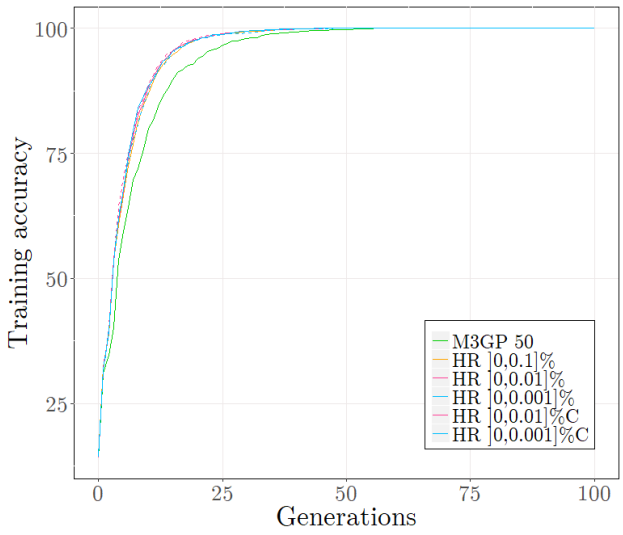


(e) YST - Training.

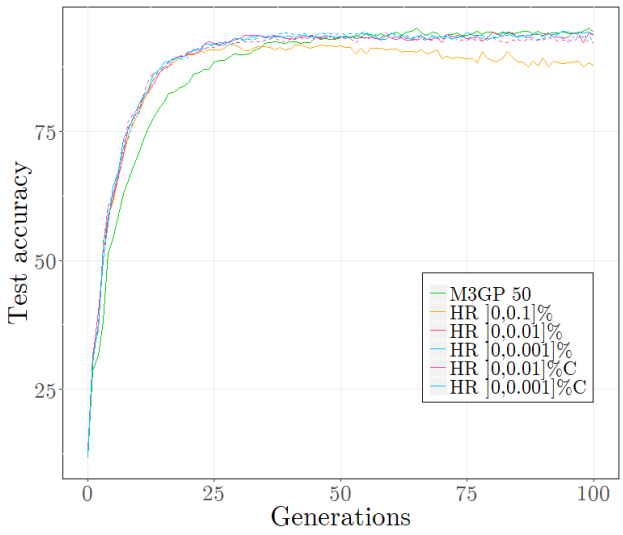


(f) YST - Test.

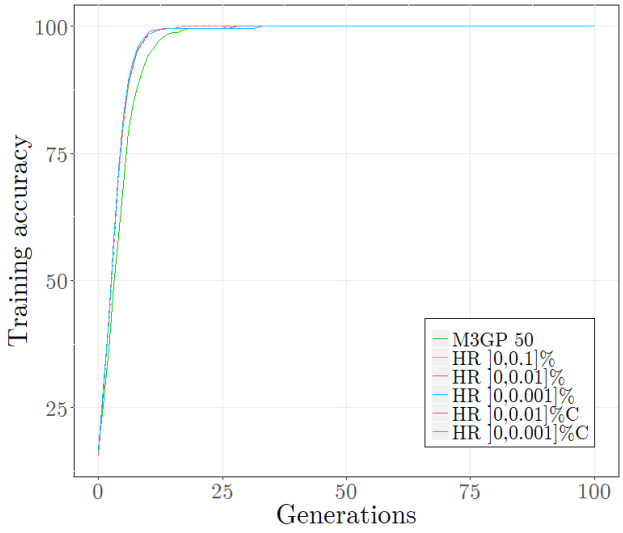
Figure 5.19: Comparison between median accuracies of M3GP and the hyperrectangle variants on SEG, IM-10 and YST datasets.



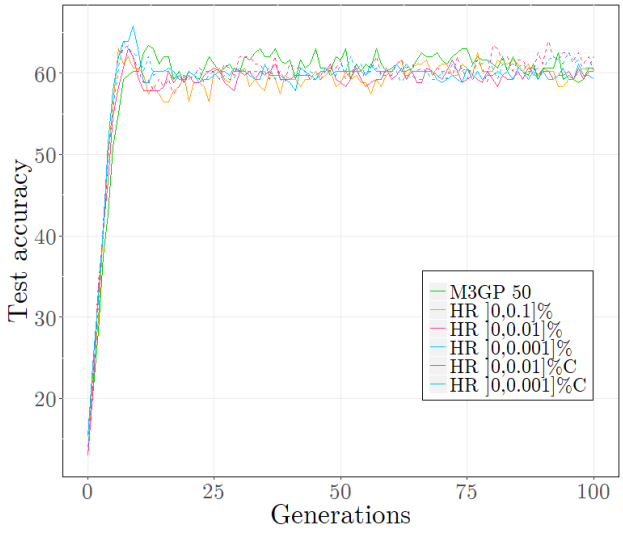
(a) VOW - Training.



(b) VOW - Test.



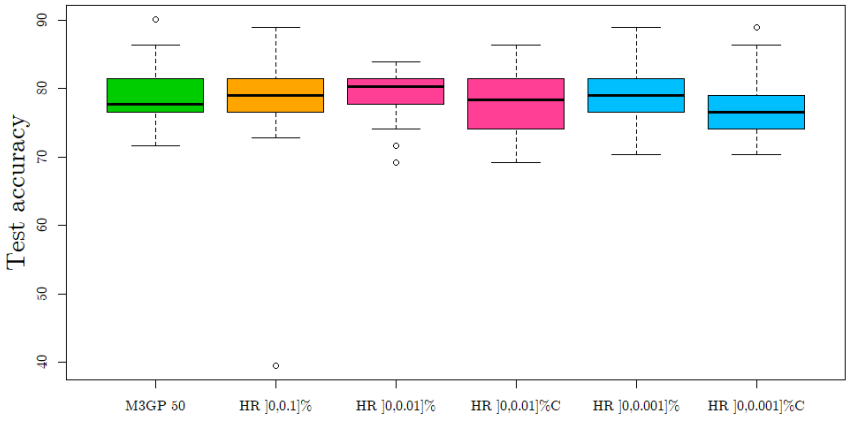
(c) M-L - Training.



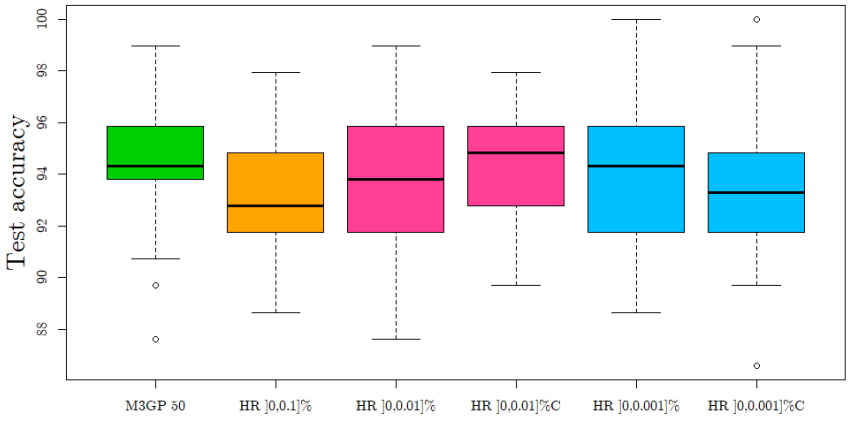
(d) M-L - Test.

Figure 5.20: Comparison between median accuracies of M3GP and the hyperrectangle variants on VOW and MOVL datasets.

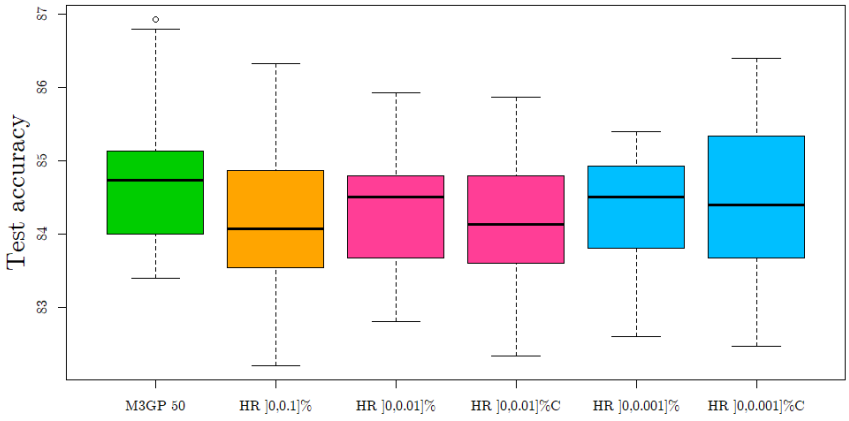
On HRT, all the algorithms seem to perform in the same way on the last generation except for HR]0,0.01]%C, which, although having a higher median than the remaining ones, the range of values is smaller - not being able to get to higher values like the remaining ones do. HR]0,0.1]% and HR]0,0.001]% are the ones reaching the highest values (see figure 5.21a). On the IM-3 dataset, HR]0,0.001]% produces higher than M3GP, but it also produces lower values (see figure 5.21b). On WAV, M3GP is the best approach, as one can see in figure 5.21c). HR]0,0.001]% is the best approach on SEG, reaching higher test values than all the others and having the minimum test accuracy higher than M3GP (if ignoring the outlier, in figure 5.22a.



(a) HRT - Unseen.

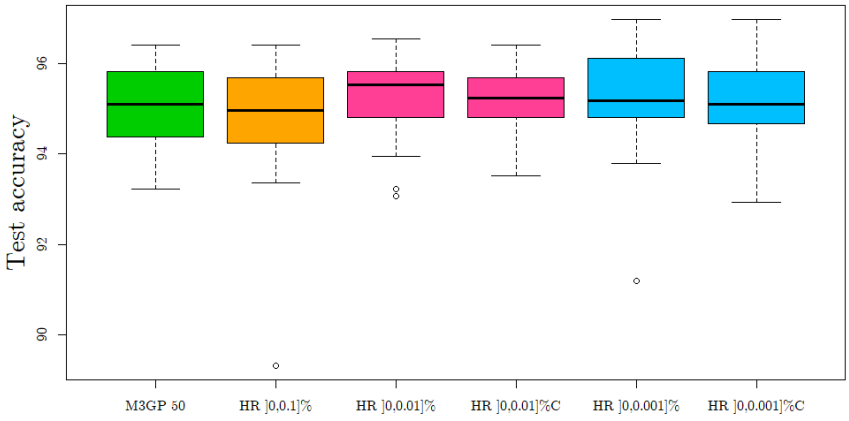


(b) IM-3 - Unseen.

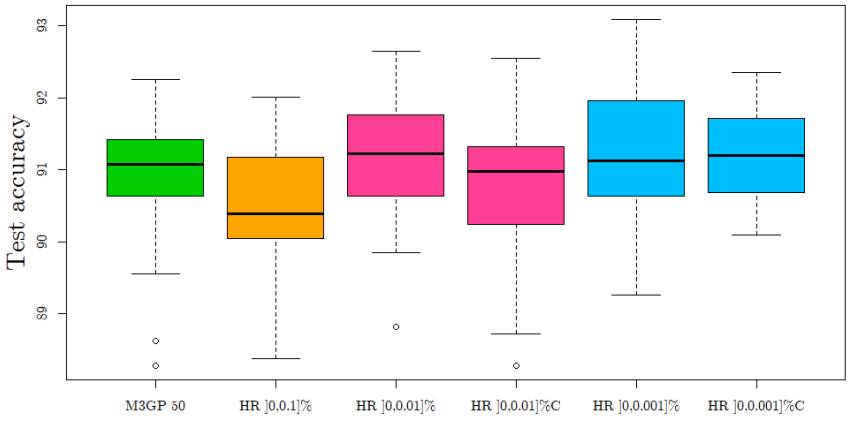


(c) WAV - Unseen.

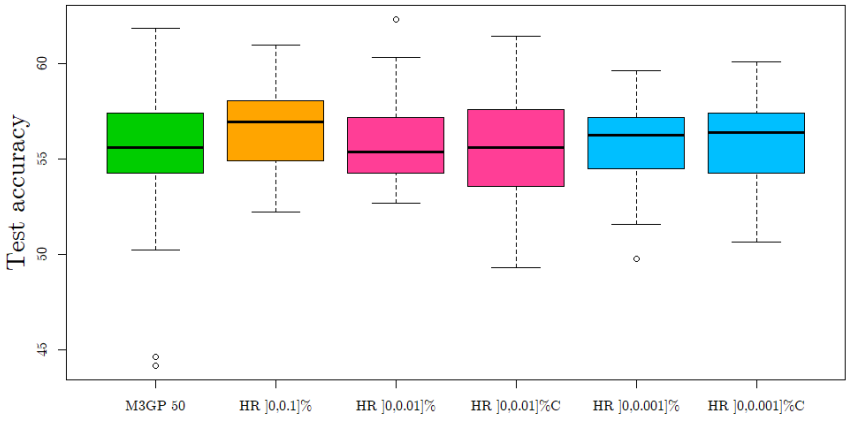
Figure 5.21: Comparison between last generation individuals' accuracies of M3GP 50 and the hyperrectangle variants on Heart, IM-3 and WAV datasets.



(a) SEG - Unseen.

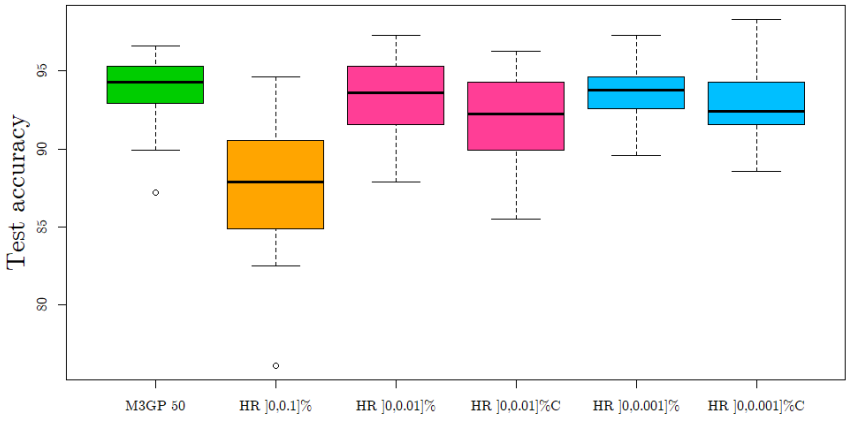


(b) IM-10 - Unseen.

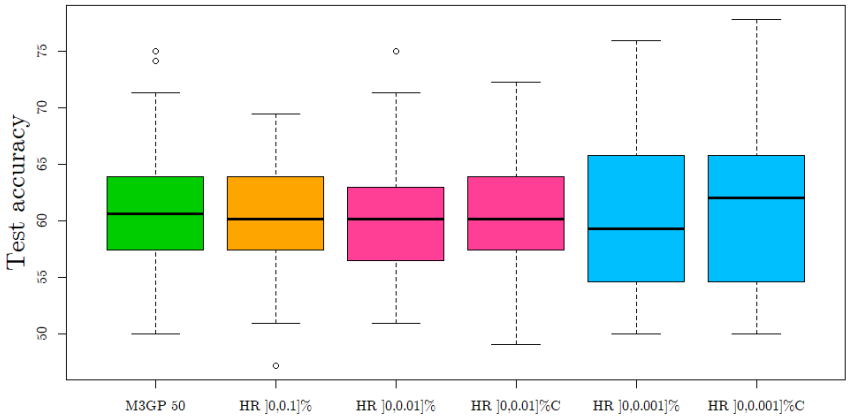


(c) YST - Training.

Figure 5.22: Comparison between last generation individuals' accuracies of M3GP 50 and the hyperrectangle variants on SEG, IM-10 and YST datasets.



(a) VOW - Unseen.



(b) MOVL - Unseen.

Figure 5.23: Comparison between last generation individuals' accuracies of M3GP 50 and the hyperrectangle variants on WAV and M-L datasets.

HR]0,0.01]% and HR]0,0.001]% are the approaches that achieve the highest values on IM-10 dataset (with HR]0,0.01]% having a smaller range of values than HR]0,0.001]%, but also not reaching has high values as HR]0,0.001]%, as expressed in figure 5.22b). Although HR]0,0.1]% has a higher median accuracy than M3GP, it does not reach as higher values as M3GP (see figure 5.22c). HR]0,0.001]% behaves quite similarly to M3GP on VOW (as in figure 5.22c). HR]0,0.001]% and HR]0,0.001]%C are the approaches achiving the highest test values on M-L, as it can be seen in figure 5.23b.

Although there is not a variant that is better than the others, HR]0,0.001]% seems to perform well and so, *pred*% is set to]0,0.001]% on the subsequent runs of the GSI-M3GP.

5.3.3 Hyperellipse

On the Hyperellipse variant (HE):

- the stopping criterion: 100 generations;
- $pred\% \in]0, 0.001]\%$

HE stands for Hyperellipse and $\chi^2_{\alpha\%}$ stands for the distance between the centroid and the points belonging to the hyperellipse.

Variant 1:

Table 5.24 stores the results of training accuracy, test accuracy and number of dimensions of M3GP, HE $\chi^2_{99\%}$, HE $\chi^2_{50\%}$ and HE $\chi^2_{25\%}$. Regarding training accuracy all the variants perform similarly (except on WAV, where the training accuracy of the hyperellipse variants is significantly better than M3GP's training accuracy). There is also no statistical difference between the test accuracy of M3GP and the test accuracy reached using the hyperellipse variants (except for HE $\chi^2_{25\%}$ on IM-10 and HE $\chi^2_{99\%}$ on VOW, where these hyperellipse variants perform worse than the hyperellipse variants and M3GP).

HE $\chi^2_{99\%}$ produces higher median values of test accuracy than M3GP on HRT, IM-3, IM-10 and YST. On IM-3, SEG, IM-10, YST and M-L HE $\chi^2_{50\%}$ reaches higher median values of test accuracy than M3GP. Finally, the HE $\chi^2_{25\%}$ hyperellipse variant of GSI-M3GP only reaches higher median test accuracy values on SEG and M-L datasets.

	HRT	IM-3	WAV	SEG	IM-10	YST	VOW	M-L
Training accuracy								
M3GP 50	**93.1**(3.2)	**98.7**(0.6)	89.6(0.6)	**96.4**(0.7)	**92.3**(0.8)	**66.0**(5.0)	**100**(0.0)	**100**(0.6)
HE $\chi^2_{99\%}$	**93.1**(1.6)	**98.7**(0.9)	**90.6**(0.6)	**96.5**(0.7)	**92.4**(0.8)	**65.7**(2.4)	**100**(0.0)	**100**(1.0)
HE $\chi^2_{50\%}$	**93.7**(2.7)	**98.7**(0.6)	**90.6**(0.4)	**96.6**(0.6)	**92.5**(0.6)	**65.6**(1.4)	**100**(0.0)	**100**(0.3)
HE $\chi^2_{25\%}$	**93.9**(2.2)	**99.1**(0.6)	**90.6**(0.5)	**96.6**(0.5)	**92.3**(0.8)	**65.2**(3.1)	**100**(0.0)	**100**(0.5)
Test accuracy								
M3GP 50	**77.8**(4.0)	**94.3**(2.6)	**84.7**(0.9)	**95.1**(1.0)	**91.1**(0.9)	**55.6**(3.8)	**94.3**(2.3)	**60.6**(5.9)
HE $\chi^2_{99\%}$	**79.6**(4.7)	**94.8**(2.3)	**84.1**(0.9)	**94.9**(2.6)	**91.5**(1.0)	**55.8**(2.2)	92.4(2.7)	**56.5**(6.9)
HE $\chi^2_{50\%}$	**77.2**(5.2)	**94.8**(2.6)	**84.4**(0.4)	**95.3**(0.9)	**91.3**(0.9)	**56.1**(2.1)	**93.9**(2.1)	**61.6**(6.8)
HE $\chi^2_{25\%}$	**77.8**(4.9)	**93.8**(2.6)	**84.1**(0.9)	**95.2**(6.8)	90.6(0.9)	**54.9**(2.5)	**93.6**(2.4)	**61.1**(4.7)
# of dimensions								
M3GP 50	12.5(1-20)	6(3-13)	27(22-32)	11(6-18)	18(9-23)	10(6-20)	20(15-24)	11(8-32)
HE $\chi^2_{99\%}$	16(9-21)	9(3-12)	32(26-39)	10(6-17)	15.5(8-20)	12.5(9-26)	22.5(18-26)	11(7-13)
HE $\chi^2_{50\%}$	15(4-24)	9(3-15)	32.5(23-38)	10(6-16)	16(9-21)	14(10-21)	22(18-26)	11(9-12)
HE $\chi^2_{25\%}$	15(2-21)	8.5(4-17)	32(23-38)	9(5-15)	16(8-22)	13(10-21)	22(18-26)	11(8-13)

Table 5.24: Comparison between M3GP, HE $\chi^2_{99\%}$, HE $\chi^2_{50\%}$ and HE $\chi^2_{25\%}$.

The median, minimum and maximum depth values found on the 30 runs of the algorithms can be seen in table 5.25. The median values are similar, implying that the usage of the mutation operator is roughly the same independently of the considered variant. It is interesting to see that, for the majority of the datasets, the median depth reached for HE $\chi^2_{50\%}$ is usually a value between the other two variants median depth values (except on HRT, IM-10 and M-L).

	HRT	IM-3	WAV	SEG	IM-10	YST	VOW	M-L
HE $\chi^2_{99\%}$	514.5 (435-594)	541.5 (356-643)	392 (316-449)	514.5 (408-583)	469.5 (328-545)	525 (370-587)	440 (296-568)	545 (405-615)
HE $\chi^2_{50\%}$	518.5 (438-685)	524 (399-636)	385 (336-463)	518 (441-581)	478.5 (307-524)	520.5 (489-573)	417 (336-597)	573 (364-636)
HE $\chi^2_{25\%}$	517.5 (462-629)	538 (334-636)	392 (301-469)	518.5 (476-583)	468 (351-594)	531 (345-587)	411 (345-497)	559.5 (392-636)

Table 5.25: Comparison between HE $\chi^2_{99\%}$, HE $\chi^2_{50\%}$ and HE $\chi^2_{25\%}$ depth values.

The *mutation percentage* is bigger than the other two operators' percentages, on all the variants. Also, the *mutation percentage* values do not vary a lot between variants, implying that the usage of the mutation operator is the same for all variants.

		HRT	IM-3	WAV	SEG	IM-10	YST	VOW	M-L
HE $\chi^2_{99\%}$	M	74%	76.933%	57%	75.7%	72%	77%	63%	76%
	A	21%	15.532%	41%	18.6%	24%	19%	30%	17%
	R	5%	7.535%	2%	5.7%	4%	4%	7%	7%
HE $\chi^2_{50\%}$	M	76%	77%	57%	77%	71%	77.39%	60%	77%
	A	20%	16%	41%	18%	24%	19.36%	31%	17%
	R	4%	7%	2%	5%	5%	3.25%	9%	6%
HE $\chi^2_{25\%}$	M	76%	75%	57%	78%	72%	77%	60.2%	76%
	A	20%	17%	40%	17%	24%	19%	31.5%	17%
	R	4%	8%	3%	5%	4%	4%	8.3%	7%

Table 5.26: Comparison between HE $\chi^2_{99\%}$, HE $\chi^2_{50\%}$ and HE $\chi^2_{25\%}$ operators' percentages.

The average number of operations is always above 90, implying that the algorithm improves from generation to generation (except on VOW and M-L for which the training accuracy gets to 100% very early) as expressed in table 5.27.

	HRT	IM-3	WAV	SEG	IM-10	YST	VOW	M-L
HE $\chi^2_{99\%}$	98.7 (96-100)	99.1 (95-100)	97.7 (93-100)	96.5 (91-100)	92.4 (83-99)	95.1 (82-100)	99.3 (96-100)	98.9 (93-100)
HE $\chi^2_{50\%}$	98.6 (96-100)	99.1 (95-100)	97.3 (95-100)	95.8 (90-100)	91.9 (79-98)	95.7 (90-100)	99.0 (98-100)	99.6 (97-100)
HE $\chi^2_{25\%}$	98.9 (96-100)	99.5 (97-100)	97.8 (94-100)	96.4 (92-100)	92.2 (85-98)	94.7 (87-100)	99.6 (97-100)	99.2 (96-100)

Table 5.27: Comparing the number of operations between HE $\chi^2_{99\%}$, HE $\chi^2_{50\%}$ and HE $\chi^2_{25\%}$.

Figures 5.24 and 5.25 show the evolution of training and test accuracies of M3GP and the hyperellipse variants, as the number of generations increases. Figures 5.24a,c,e,g and 5.25a,c,e,f show that M3GP and the hyperellipse variants perform similarly on the training set, on all datasets except on WAV. However, on the first generations the hyperellipse variants perform better. On WAV's training set, the hyperellipse variants perform better than M3GP (see figure 5.24e).

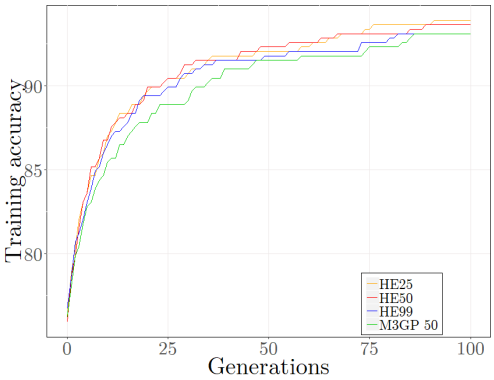
On the test set:

- HE $\chi^2_{50\%}$'s median accuracy is higher than the remaining algorithms' accuracy from generation 25 on, on HRT (see figure 5.24b);

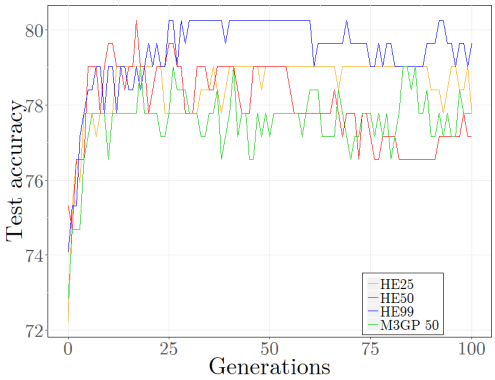
- On IM-3 and SEG there is no clear distinction between the median accuracies of the different algorithms (see figures 5.24d,h);
- On WAV, there is an almost invisible overfitting of the hyperellipse variants (there is a small decrease in the median accuracy from generation 50 on), and in the first 25 generations the median test accuracies of the hyperellipse variants are always bigger than M3GP's test accuracy (see figure 5.24f);
- On IM-10 and YST the median accuracy values of HE $\chi^2_{50\%}$ and HE $\chi^2_{99\%}$ are always above the median acuracy values of M3GP and HE $\chi^2_{25\%}$, from a certain generation on (see figures 5.27b,d);
- For VOW, in the first 30 generations, the median test accuracy of the hyperellipse variants is above the median test accuracy of M3GP, but then the median test accuracies of M3GP and HE $\chi^2_{25\%}$ become higher (5.27f);
- On M-L, all the algorithms overfit (see figures 5.25g,h).

The behaviour of the algorithms on the last generation can be seen in figures 5.26 and 5.27. Regarding last generation training accuracy, there is not a big difference between the algorithms' training accuracy on all datasets except for WAV, as it can be seen in figures 5.26a,c,e,f and 5.27a,c,e,f. This difference also exists on WAV's test set, where M3GP still reaches higher accuracy values (even though the difference is not statistically significant).

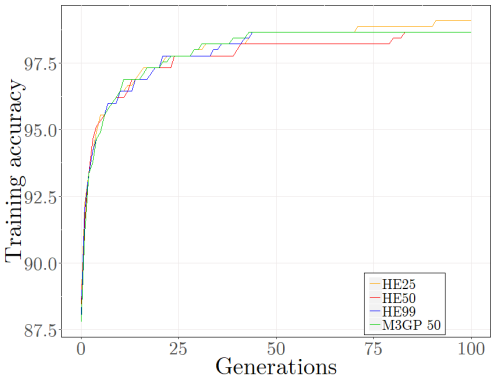
On HRT, ignoring outliers, HE $\chi^2_{25\%}$ is able to reach slightly higher values of test accuracy then M3GP, but also smaller values, as one can see in figure 5.26b. On IM-3, M3GP and HE $\chi^2_{50\%}$ reach the same range of test accuracy values, although HE $\chi^2_{50\%}$ has a higher median accuracy. On SEG, HE $\chi^2_{25\%}$ is able to reach higher values of test accuracy, when compared to M3GP (see figure 5.24h). On IM-10 and M-L, HE $\chi^2_{50\%}$ reaches higher values of median test accuracy but also lower values, in comparison to M3GP (see figures 5.25b,h). On YST, M3GP reaches higher values than all hyperellipse variants (see figure 5.25d). HE $\chi^2_{50\%}$ and M3GP act in the same manner on VOW, as one can conclude by looking at figure 5.25f).



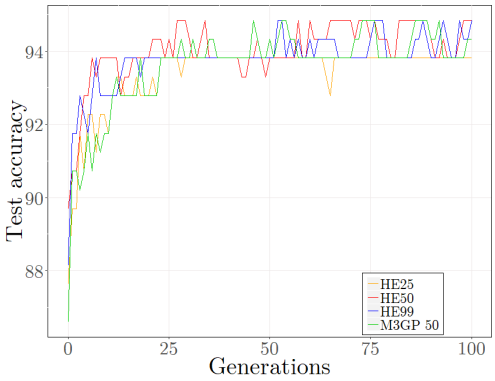
(a) HRT - Training.



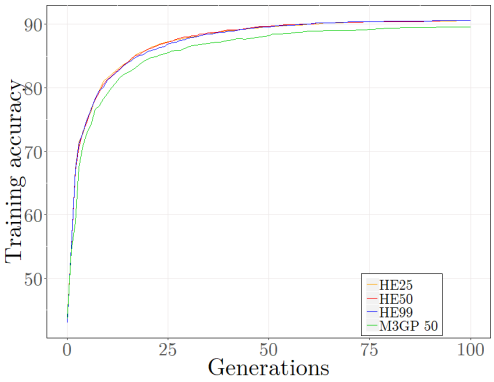
(b) HRT - Test.



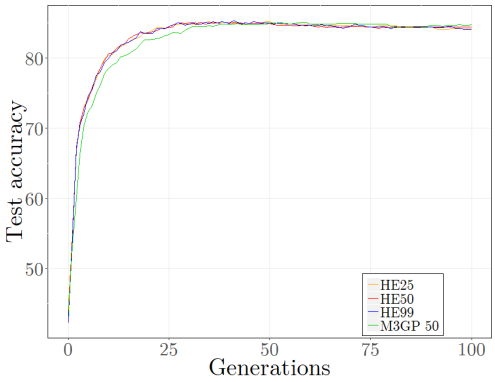
(c) IM-3 - Training.



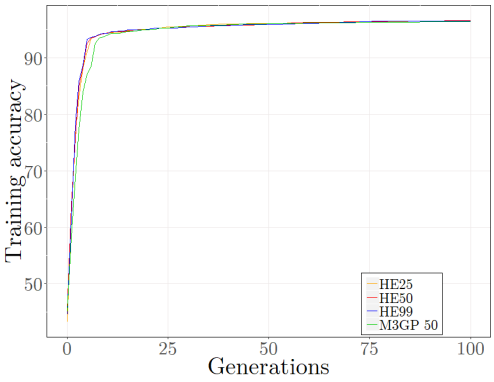
(d) IM-3 - Test.



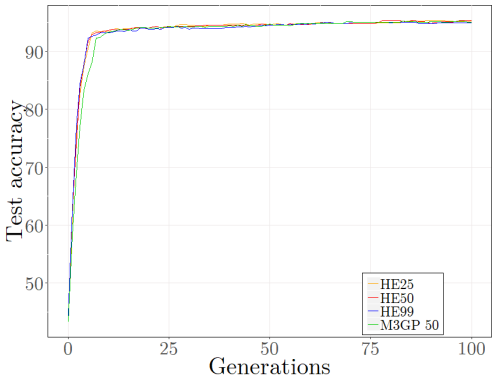
(e) WAV - Training.



(f) WAV - Test.

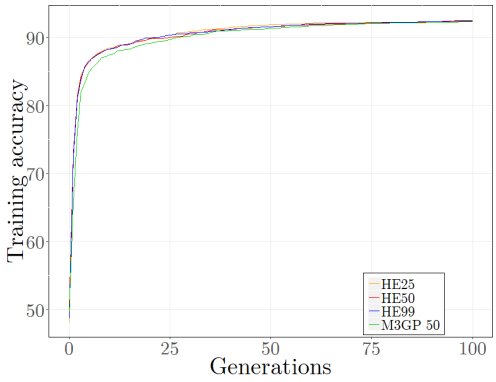


(g) SEG - Training.

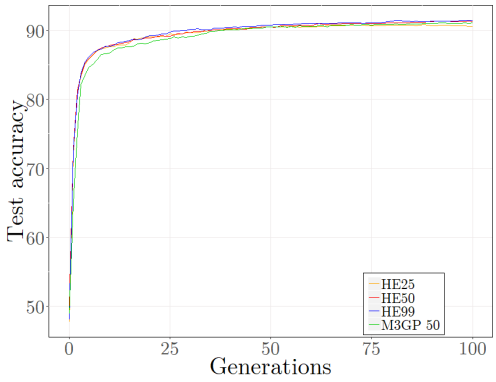


(h) SEG - Test.

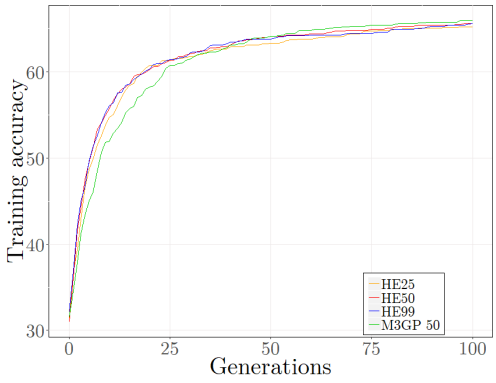
Figure 5.24: Comparison between median accuracies of M3GP 50, HE $\chi^2_{25\%}$, HE $\chi^2_{50\%}$ and HE $\chi^2_{99\%}$ algorithms on HRT, IM-3, WAV and SEG datasets.



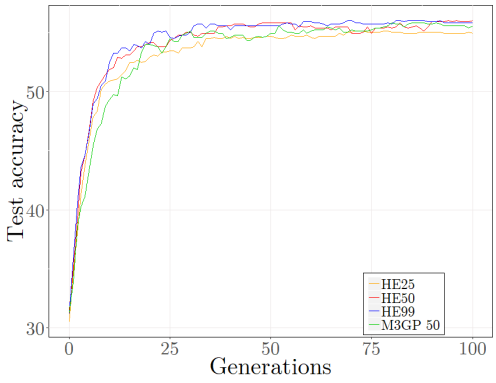
(a) IM-10 - Training.



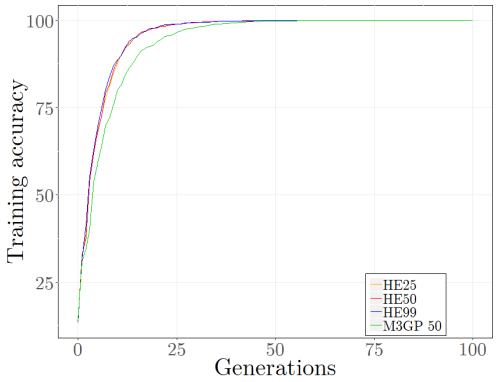
(b) IM-10 - Test.



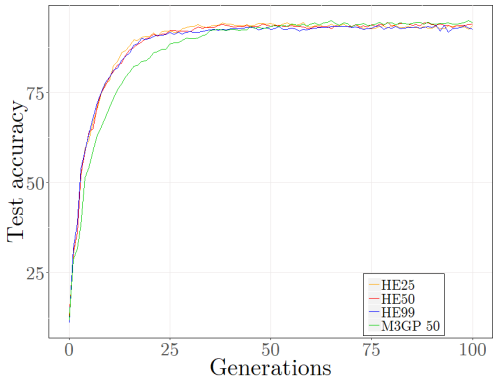
(c) YST - Training.



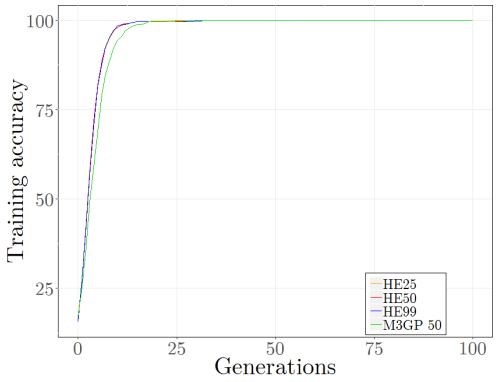
(d) YST - Test.



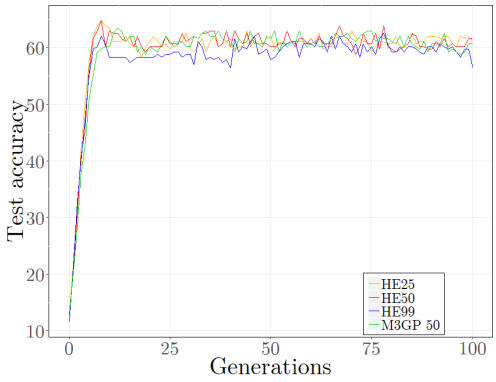
(e) VOW - Training.



(f) VOW - Test.

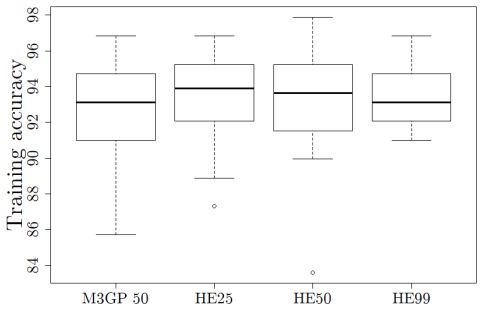


(g) M-L - Training.

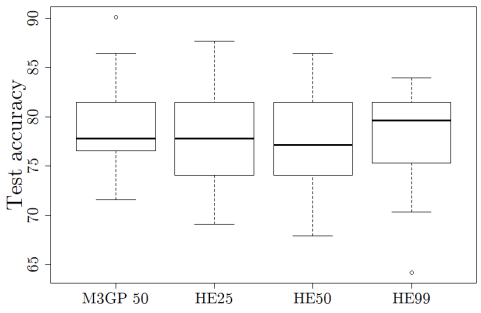


(h) M-L - Test.

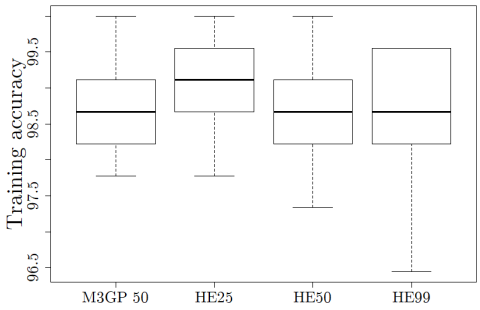
Figure 5.25: Comparison between median accuracies of M3GP 50, HE $\chi^2_{25\%}$, HE $\chi^2_{50\%}$ and HE $\chi^2_{99\%}$ algorithms on IM-10, YST, VOW and M-L datasets.



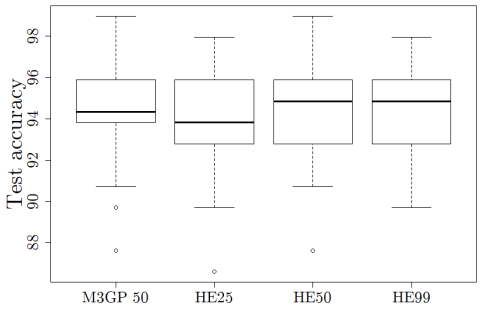
(a) HRT - Training.



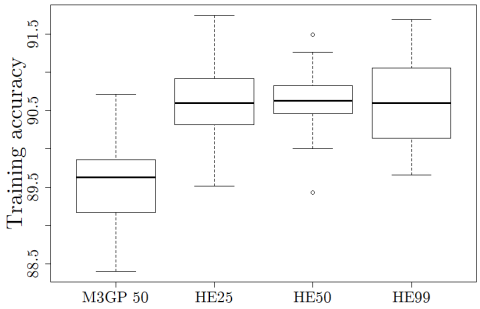
(b) HRT - Test.



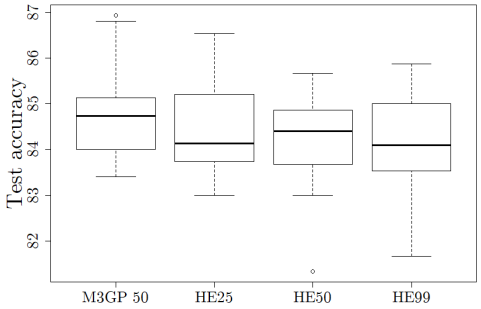
(c) IM-3 - Training.



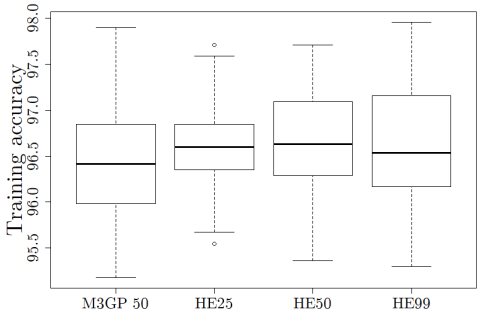
(d) IM-3 - Test.



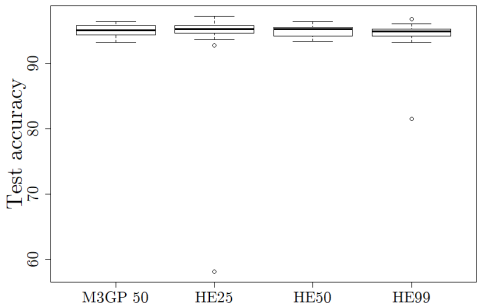
(e) WAV - Training.



(f) WAV - Test.

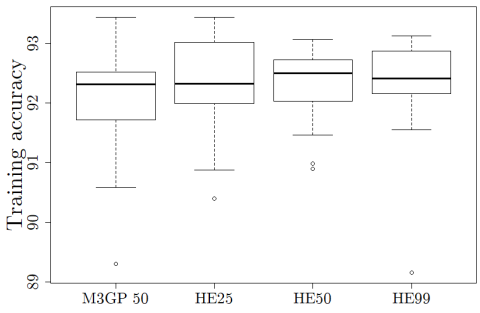


(g) SEG - Training.

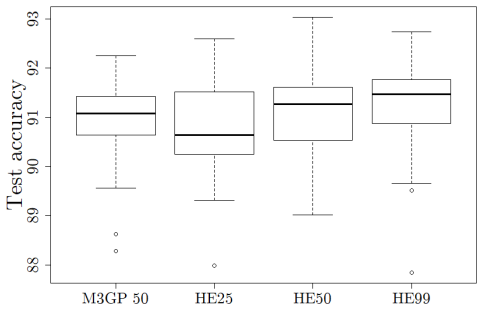


(h) SEG - Test.

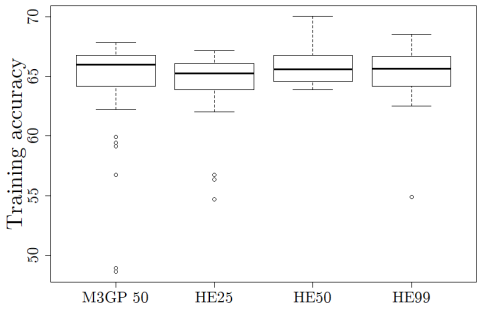
Figure 5.26: Comparison between last generation individuals' accuracies of M3GP 50, HE $\chi^2_{25\%}$, HE $\chi^2_{50\%}$ and HE $\chi^2_{99\%}$ algorithms on HRT, IM-3, WAV and SEG datasets.



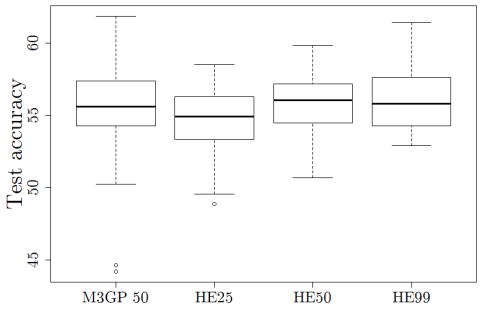
(a) IM-10 - Training.



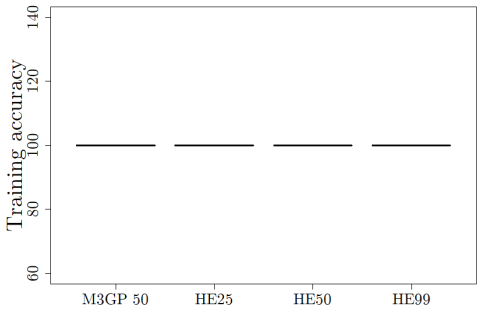
(b) IM-10 - Test.



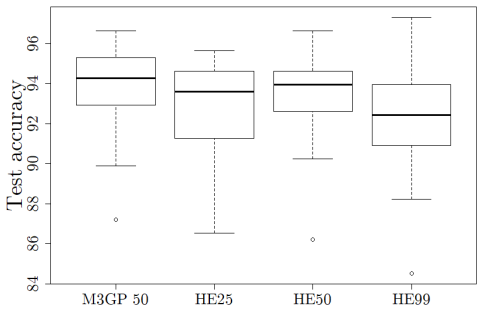
(c) YST - Training.



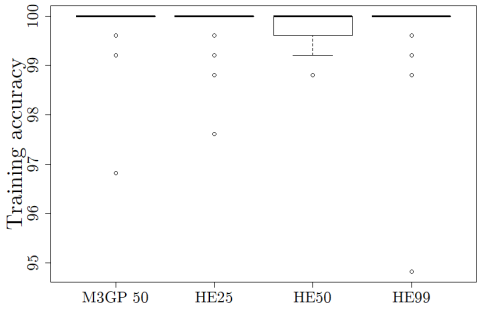
(d) YST - Test.



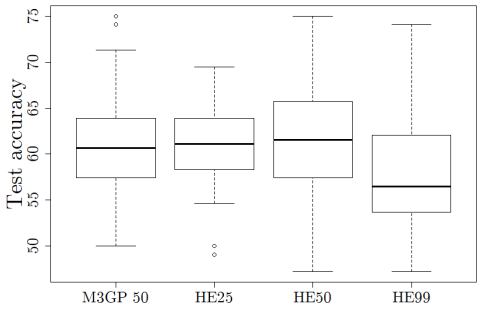
(e) VOW - Training.



(f) VOW - Test.



(g) M-L - Training.



(h) M-L - Test.

Figure 5.27: Comparison between last generation individuals' accuracies of M3GP 50, HE $\chi^2_{25\%}$, HE $\chi^2_{50\%}$ and HE $\chi^2_{99\%}$ algorithms on IM-10, YST, VOW and M-L datasets.

variant 2:

Table 5.28 shows a comparison between M3GP and the hyperellipse variants regarding training and test accuracy. Additionally, the number of dimensions is also shown.

There is no statistical difference between the three algorithms' median training accuracy (except for M3GP which is worse than the hyperellipse variants on WAV, and HE $\chi^2_{50\%}$ which is worse than the remaining algorithms on M-L). On the test set, the median test accuracy of HE $\chi^2_{50\%}$ on WAV and VOW is statistically worse. Also, the hyperellipse variants generate median test accuracies on M-L which are statistically better than the accuracy generated by M3GP.

	HRT	**IM-3**	**WAV**	**SEG**	**IM-10**	**YST**	**VOW**	**M-L**
Training accuracy								
M3GP 50	**93.1**(3.2)	**98.7**(0.6)	89.6(0.6)	**96.4**(0.7)	**92.3**(0.8)	**66.0**(5.0)	**100**(0.0)	**100**(0.6)
HE $\chi^2_{99\%}$	**93.4**(1.9)	**98.7**(1.0)	**90.8**(0.4)	**96.8**(0.6)	**92.5**(0.6)	**65.7**(2.4)	**100**(0.0)	**100**(0.4)
HE $\chi^2_{50\%}$	**93.7**(2.1)	**98.9**(0.6)	**90.9**(0.4)	**96.6**(0.6)	**92.5**(0.5)	**65.8**(1.6)	**100**(0.0)	99.8(0.6)
Test accuracy								
M3GP 50	**77.8**(4.0)	**94.3**(2.6)	**84.7**(0.9)	**95.1**(1.0)	**91.1**(0.9)	**55.6**(3.8)	**94.3**(2.3)	60.6(5.9)
HE $\chi^2_{99\%}$	**79.0**(4.4)	**93.8**(2.5)	**84.2**(0.8)	**95.2**(1.4)	**91.3**(0.7)	**55.4**(2.2)	**92.6**(2.6)	**63.9**(6.9)
HE $\chi^2_{50\%}$	**79.6**(5.4)	**93.8**(2.2)	83.9(0.9)	**95.2**(0.9)	**91.2**(4.4)	**55.4**(2.0)	92.8(2.5)	**62.5**(4.6)
# of dimensions								
M3GP 50	12.5(1-20)	6(3-13)	27(22-32)	11(6-18)	18(9-23)	10(6-20)	20(15-24)	11(8-32)
HE $\chi^2_{99\%}$	14(7-22)	9(3-17)	32(26-40)	10(5-16)	15.5(9-22)	14(7-17)	22(18-27)	11(8-139
HE $\chi^2_{50\%}$	16(6-23)	9(3-15)	34(25-39)	10.5(6-19)	16(12-19)	13(11-19)	22(16-24)	11(9-12)

Table 5.28: Comparison between M3GP and HE $\chi^2_{99\%}$ and $\chi^2_{50\%}$.

Table 5.29 stores the values of depth obtained with the hyperellipse variants of GSI-M3GP. The median depth values of HE $\chi^2_{50\%}$ are smaller than the median depth values of HE $\chi^2_{99\%}$ for HRT, IM-3, IM-10, VOW and M-L.

	HRT	**IM-3**	**WAV**	**SEG**	**IM-10**	**YST**	**VOW**	**M-L**
HE $\chi^2_{99\%}$	521 (406-587)	557 (370-636)	381.5 (336-448)	515 (434.609)	466 (340-559)	517.5 (437-587)	428 (331-554)	573 (357-629)
HE $\chi^2_{50\%}$	514 (420-622)	541.5 (444-643)	385 (322-427)	528.5 (454-595)	463.5 (384-538)	524 (440-601)	403.5 (323-518)	564 (454-636)

Table 5.29: Comparison between M3GP and HE $\chi^2_{99\%}$ and $\chi^2_{50\%}$ depth values.

The operators' percentages are stored in table 5.30. As for **variant 1**, the values are not distinct between variants and, most importantly, the *mutation percentage* is higher than the other operators' percentages.

		HRT	**IM-3**	**WAV**	**SEG**	**IM-10**	**YST**	**VOW**	**M-L**
HE $\chi^2_{99\%}$	*M*	74.6%	79%	56%	76.35%	70%	77%	62%	78%
	A	20.6%	15%	41%	18.19%	25%	19%	31%	16%
	R	4.8%	6%	3%	5.46%	5%	4%	7%	6%
HE $\chi^2_{50\%}$	*M*	74%	78%	55%	77.6%	70.7%	78%	59%	79.3%
	A	21%	15%	42%	15.6%	24.5%	19%	32%	15.2%
	R	5%	7%	3%	4.8%	4.8%	3%	9%	5.5%

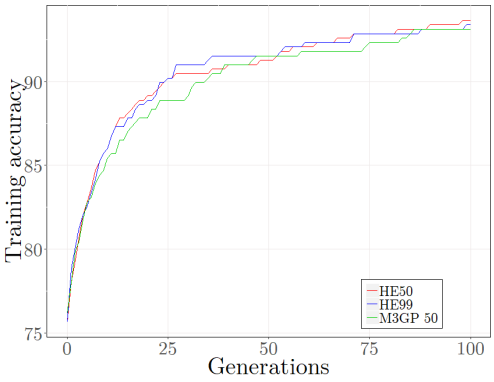
Table 5.30: Comparison between M3GP and HE $\chi^2_{99\%}$ and $\chi^2_{50\%}$ operators' percentages.

The average number of operations is higher than 90 on all problems and for both hyperellipse variants (see table 5.30).

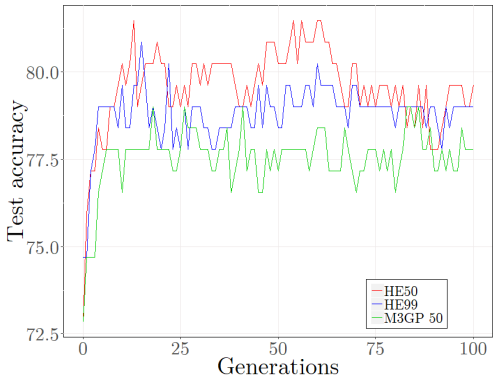
	HRT	**IM-3**	**WAV**	**SEG**	**IM-10**	**YST**	**VOW**	**M-L**
HE $\chi^2_{99\%}$	98.1 (94-100)	99.2 (97-100)	98.0 (91-100)	97.0 (89-100)	92.7 (84-99)	95.3 (90-99)	99.1 (96-100)	99.3 (94-100)
HE $\chi^2_{50\%}$	98.7 (97-100)	99.0 (96-100)	97.9 (95-100)	97.1 (93-100)	93.2 (83-100)	95.9 (83-100)	99.4 (97-100)	99.4 (97-100)

Table 5.31: Comparing the number of operations between M3GP and HE $\chi^2_{99\%}$ and $\chi^2_{50\%}$.

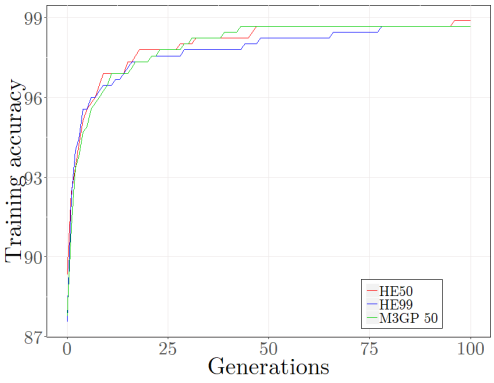
Figures 5.28 and 5.29 show the evolution of training and test accuracies of M3GP and the hyperellipse variants, as the number of generations increases. As for the previous hyperellipse variants, M3GP and the hyperellipse variants perform similarly on the training set, except on WAV (on WAV's training set, the hyperellipse variants perform better than M3GP). However, on the first generations the hyperellipse variants are able to produce higher values of training accuracy, when compared to M3GP. On the test set: HE $\chi^2_{50\%}$ achieves higher values of accuracy throughout the generations, on HRT; On IM-3, HE $\chi^2_{99\%}$ and M3GP seem to produce slightly bigger values on the last generations; on WAV, SEG and YST, the hyperellipses slightly overfit on the last generations; On VOW, M3GP produces higher values on the second half of the generations; All the approaches overfit on M-L, although HE $\chi^2_{99\%}$ produces higher values than the others throughout the generations.



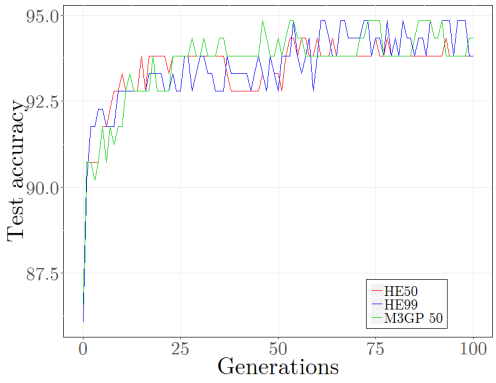
(a) HRT - Training.



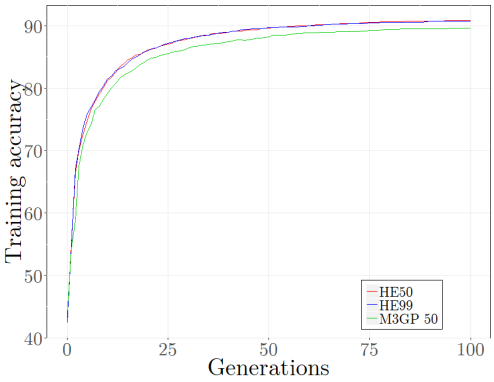
(b) HRT - Test.



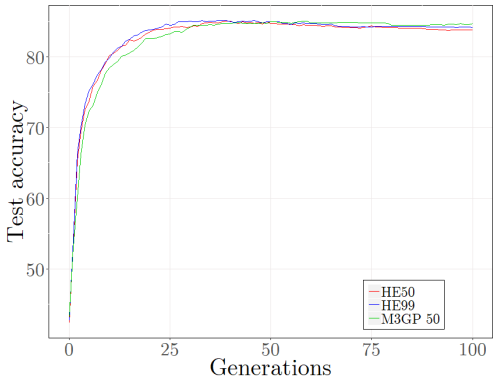
(c) IM-3 - Training.



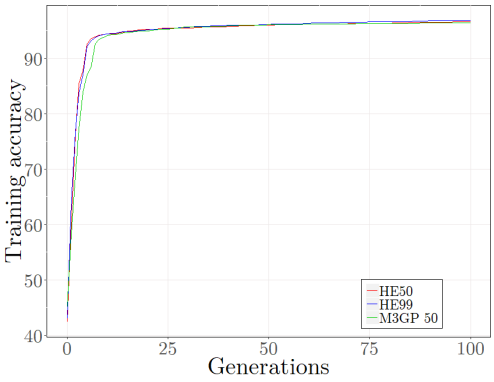
(d) IM-3 - Test.



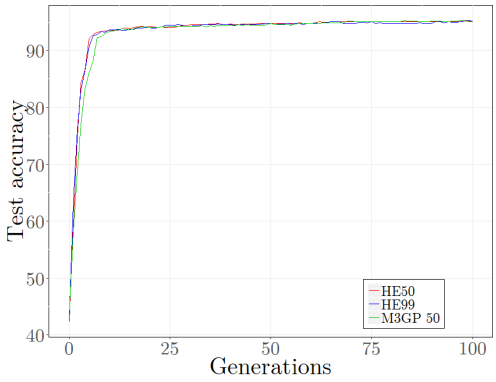
(e) WAV - Training.



(f) WAV - Test.

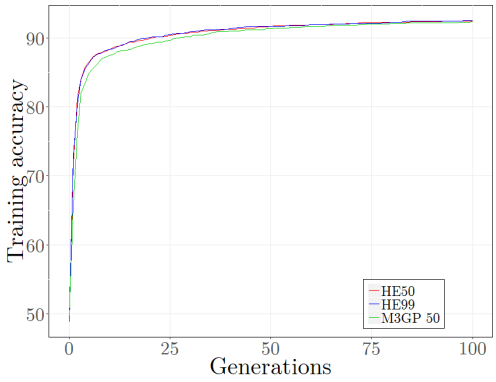


(g) SEG - Training.

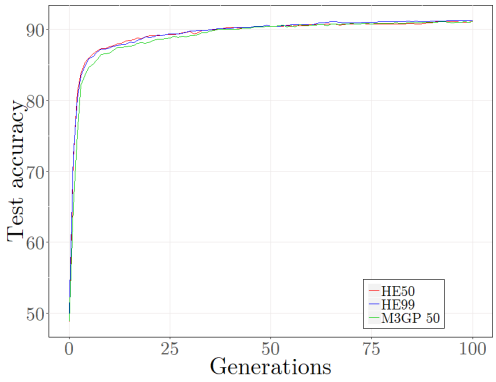


(h) SEG - Test.

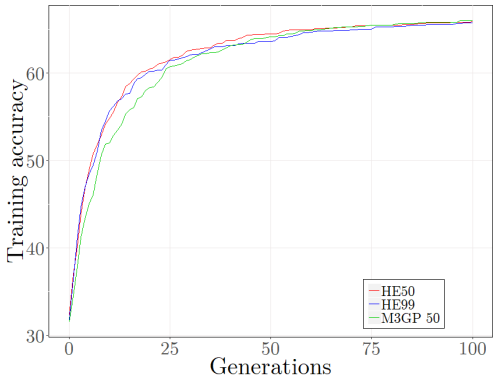
Figure 5.28: Comparison between median accuracies of M3GP 50, HE $\chi^2_{50\%}$ and HE $\chi^2_{99\%}$ algorithms on HRT, IM-3, WAV and SEG datasets.



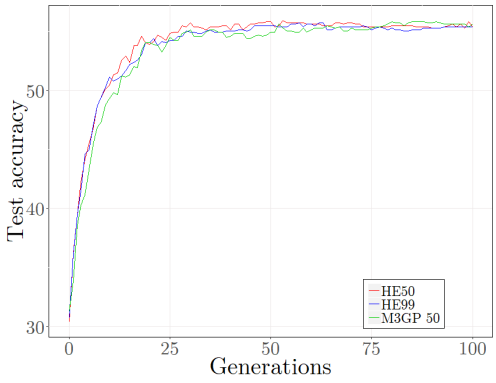
(a) IM-10 - Training.



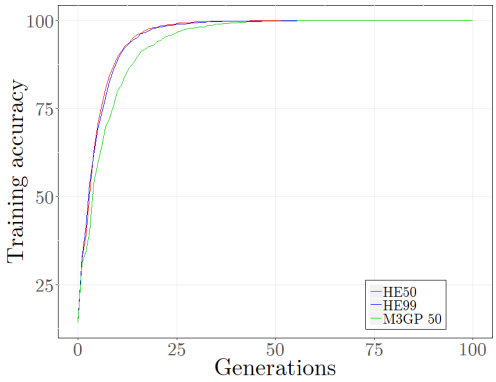
(b) IM-10 - Test.



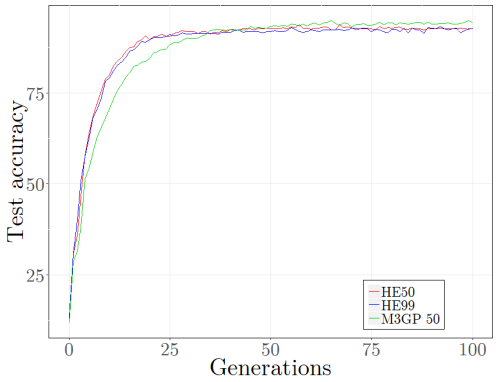
(c) YST - Training.



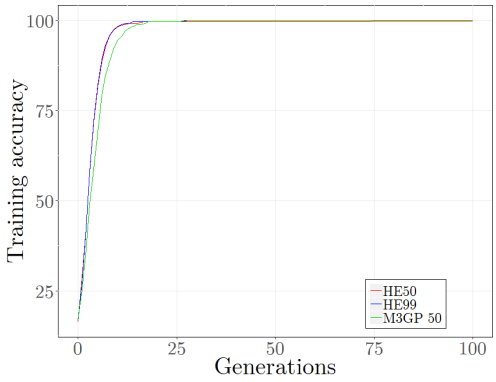
(d) YST - Test.



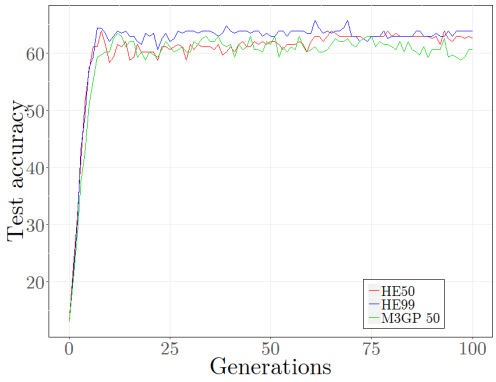
(e) VOW - Training.



(f) VOW - Test.

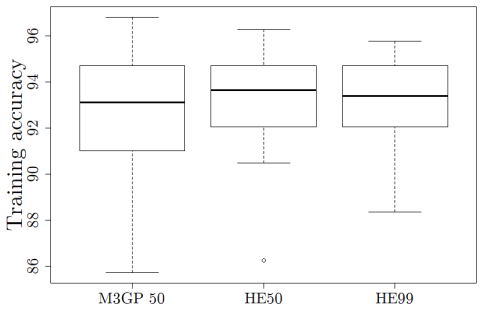


(g) M-L - Training.

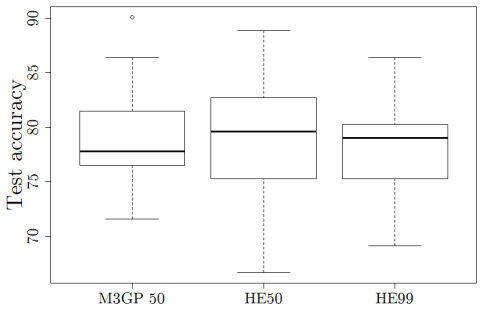


(h) M-L - Test.

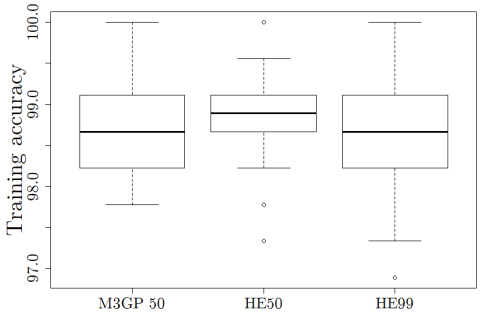
Figure 5.29: Comparison between median accuracies of M3GP 50, HE $\chi^2_{50\%}$ and HE $\chi^2_{99\%}$ algorithms on IM-10, YST, VOW and M-L datasets.



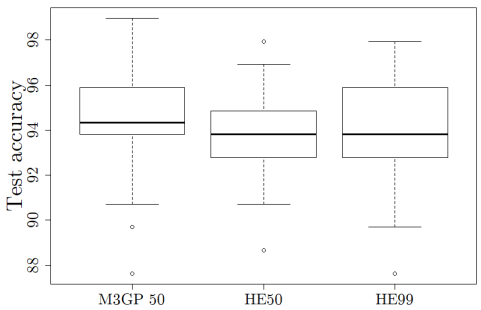
(a) HRT - Training.



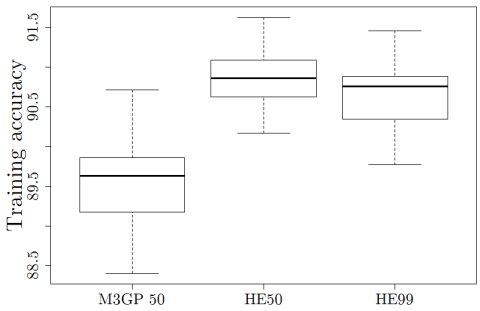
(b) HRT - Test.



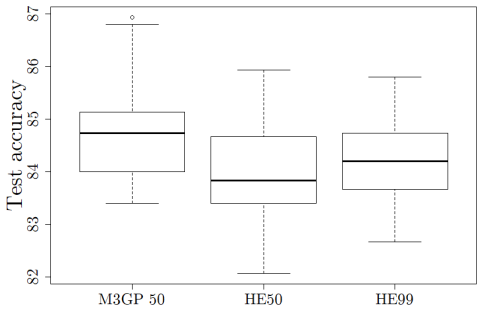
(c) IM-3 - Training.



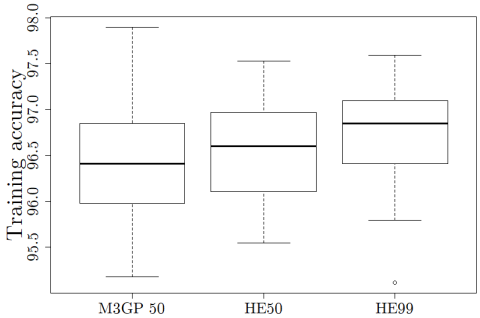
(d) IM-3 - Test.



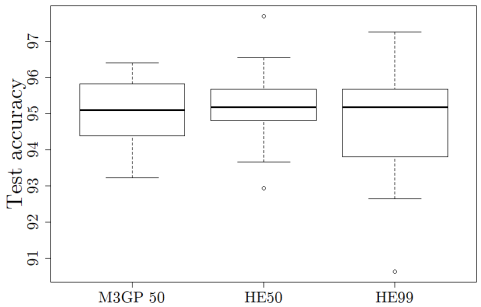
(e) WAV - Training.



(f) WAV - Test.

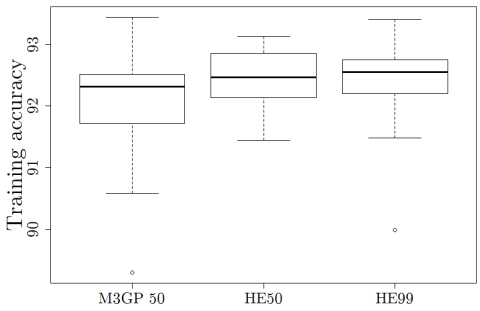


(g) SEG - Training.

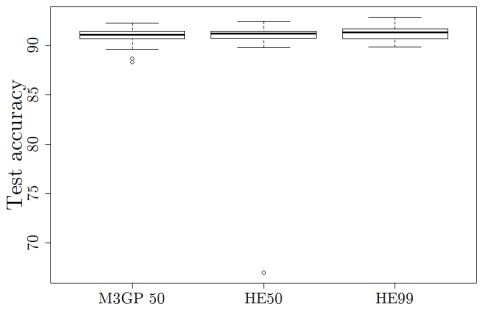


(h) SEG - Test.

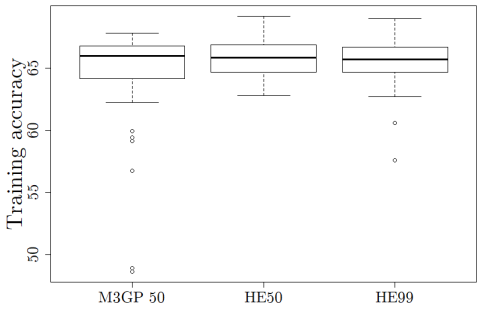
Figure 5.30: Comparison between last generation individuals' accuracies ofM3GP 50, HE $\chi^2_{50\%}$ and HE $\chi^2_{99\%}$ algorithms on HRT, IM-3, WAV and SEG datasets.



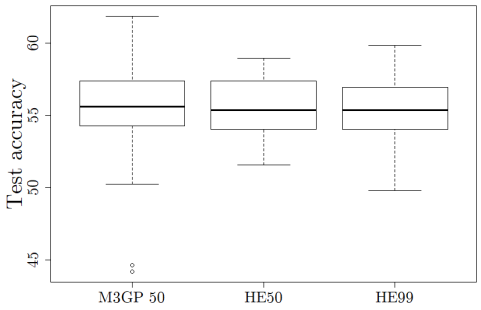
(a) IM-10 - Training.



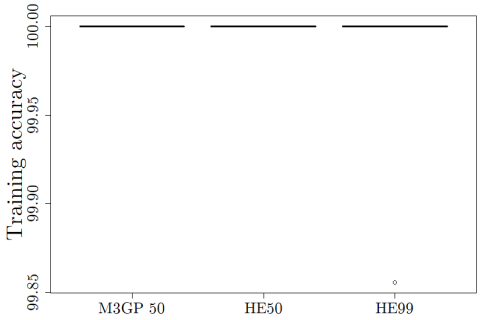
(b) IM-10 - Test.



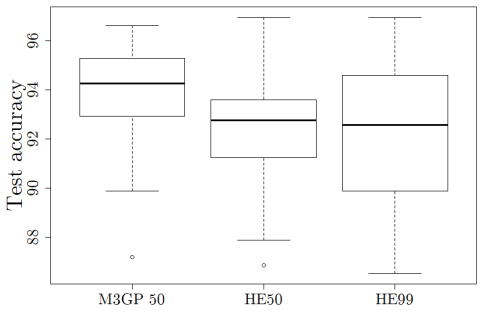
(c) YST - Training.



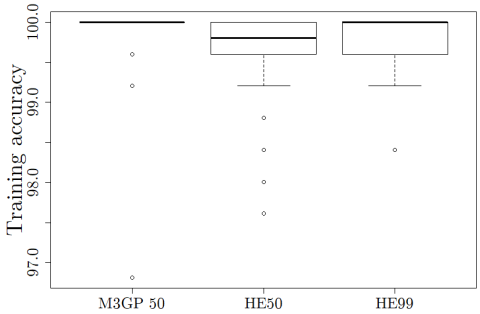
(d) YST - Test.



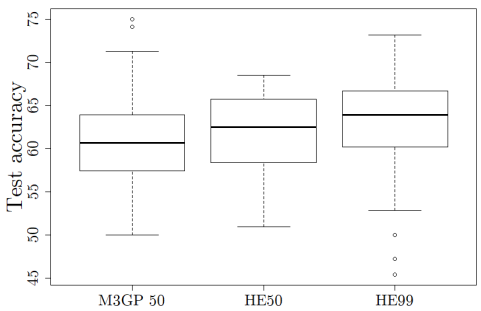
(e) VOW - Training.



(f) VOW - Test.



(g) M-L - Training.



(h) M-L - Test.

Figure 5.31: Comparison between last generation individuals' accuracies of M3GP 50, HE $\chi^2_{50\%}$, HE $\chi^2_{99\%}$ algorithms on IM-10, YST, VOW and M-L datasets.

The behaviour of the algorithms on the last generation can be seen in figures 5.30 and 5.31.

Summarizing:

- On HRT, the range of training values for M3GP is much wider than for the other two approaches, and so, M3GP is able to produce slightly higher values of training accuracy and much smaller values of training accuracy than the other two approaches. However, for the test set, HE $\chi^2_{50\%}$ is able to produce higher values than M3GP and HE $\chi^2_{99\%}$, but is does also produce lower values than the two remaining approaches;
- On IM-3, HE $\chi^2_{99\%}$ and M3GP perform similarly on train, but M3GP reaches higher values on test;
- On WAV and YST, the training accuracy values are higher for the hyperellipse variants, but the test accuracy values are higher for M3GP;
- On SEG, the range of training accuracy values is slightly wider for M3GP, but HE $\chi^2_{99\%}$ is able to reach higher values of test accuracy;
- The range of training accuracy values is less wide for the hyperellipse variants that for M3GP, and the hyperellipse variants are able to produce higher test accuracy values when compared to M3GP;
- For the training set, all the approaches perform similarly on VOW, but on the test set the hyperellipse approaches reach slightly bigger values of test accuracy but also much lower values than M3GP;
- On M-L, the three approaches perform similarly both on training and test sets.

5.3.4 Donut

On the Donut variant:

- stopping criteria: number of generations equal to 100 or training accuracy equal to 100%;
- $pred\% \in]0, 0.001]\%$

From table 5.32 one can conclude that the Donut variant of GSI-M3GP is competitive with M3GP. The median test values of the Donut variant are higher than the median values of M3GP on IM-3, SEG and IM-10, although the difference is not statistically significant.

	HRT	**IM-3**	**WAV**	**SEG**	**IM-10**	**YST**	**VOW**	**M-L**
Training accuracy								
M3GP 50	**93.1**(3.2)	**98.7**(0.6)	89.6(0.6)	**96.4**(0.7)	**92.3**(0.8)	**66.0**(5.0)	**100**(0.0)	**100**(0.6)
Donut	**93.7**(2.3)	**98.7**(0.7)	**90.9**(0.5)	**96.7**(0.6)	**92.5**(0.5)	**66.3**(1.5)	**100**(0.0)	**100**(0.3)
Test accuracy								
M3GP 50	**77.8**(4.0)	**94.3**(2.6)	**84.7**(0.9)	**95.1**(1.0)	**91.1**(0.9)	**55.6**(3.8)	**94.3**(2.3)	**60.6**(5.9)
Donut	**77.8**(4.2)	**94.8**(2.6)	**84.1**(0.8)	**95.2**(1.0)	**91.3**(11.5)	**54.7**(2.4)	**93.4**(2.8)	**55.6**(7.2)
# of dimensions								
M3GP 50	12.5(1-20)	6(3-13)	27(22-32)	11(6-18)	18(9-23)	10(6-20)	20(15-24)	11(8-32)
Donut	15(7-26)	9(3-15)	33(26-39)	11(7-18)	16(10-22)	14(10-19)	22(18-27)	11(9-13)

Table 5.32: Comparison between M3GP and the Donut variant of GSI-M3GP.

The individuals' median, minimum and maximum depth values are stored in table 5.33. These values are similar to the previous variants values.

	HRT	**IM-3**	**WAV**	**SEG**	**IM-10**	**YST**	**VOW**	**M-L**
Donut	524.5 (433-629)	538 (419-629)	392 (308-462)	504 (392-595)	469 (370-580)	526.5 (462-594)	422 (314-512)	566 (380-615)

Table 5.33: GSI-M3GP Donut variant depth values.

As for the hyperellipse variants, the mutation operator percentages of usage are always higher than the other operators' percentages, implying that the gsi-mutation operator is working well (see table 5.34).

		HRT	IM-3	WAV	SEG	IM-10	YST	VOW	M-L
Donut	*M*	75%	76.18%	56.4%	74%	72%	78%	60%	77.3%
	A	20%	16.37%	41.2%	20%	25%	19%	32%	16.4%
	R	5%	7.45%	2.4%	6%	4%	3%	8%	6.3

Table 5.34: GSI-M3GP Donut variant operators' percentages.

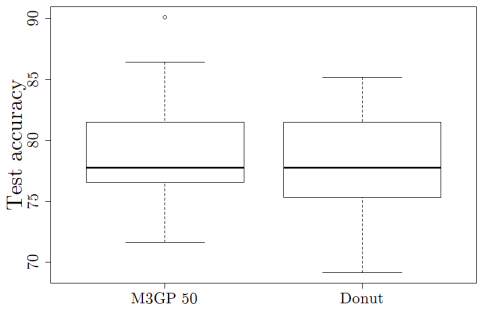
Table 5.35 shows the average, minimum and maximum number of operations. Since the average number of operations and minimum number of operations are always higher than 90 (except on IM-10), it means that the individuals created using the Donut variant of GSI-M3GP improve their training accuracy on almost every generation.

	HRT	IM-3	WAV	SEG	IM-10	YST	VOW	M-L
Donut	98.2	99.4	97.9	97.0	93.9	95.7	99.4	99.5
	(94-100)	(94-100)	(95-100)	(93-100)	(83-99)	(90-99)	(98-100)	(96-100)

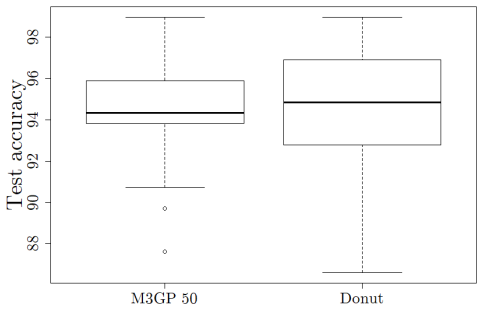
Table 5.35: GSI-M3GP Donut variant number of operations.

Figure 5.32 shows boxplots with last generation's test accuracy values. Summarizing:

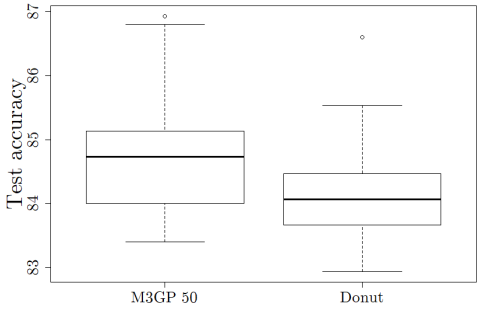
- M3GP test accuracy values are, approximately, in the same range as the test accuracy values of the Donut variant on HRT, IM-10 and M-L datasets;
- On IM-3, the minimum value of test accuracy of the Donut variant is smaller than the minimum value for M3GP. The maximum values are the same;
- On WAV and YST, the highest value of test accuracy for M3GP is higher than the highest test accuracy value for the Donut variant;
- the Donut variant is able to reach higher test accuracy values than M3GP on SEG and VOW. However, for VOW, the minimum value of test accuracy for the Donut variant is smaller than the minimum value of test accuracy for M3GP.



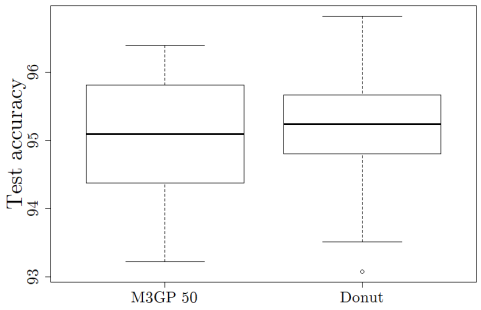
(a) HRT - Test.



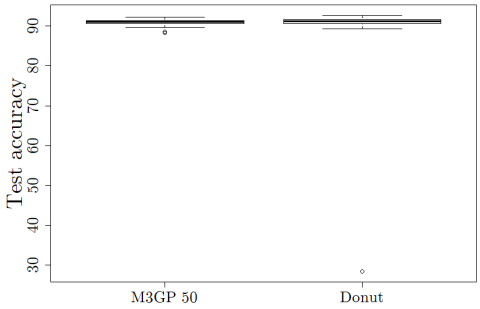
(b) IM-3 - Test.



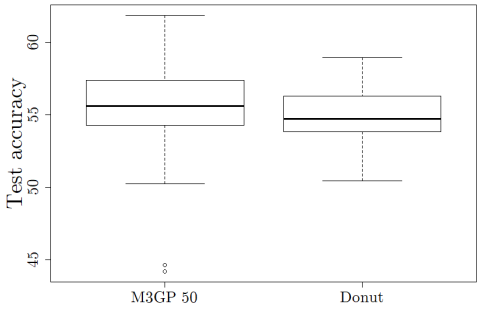
(c) WAV - Test.



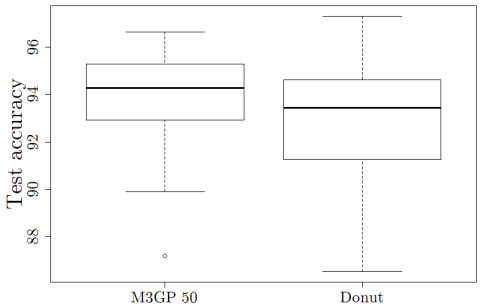
(d) SEG - Test.



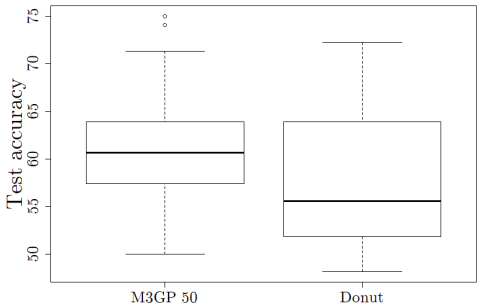
(e) IM-10 - Test.



(f) YST - Test.



(g) VOW - Test.



(h) M-L - Test.

Figure 5.32: Comparison between last generation individuals' median accuracies of M3GP 50 and Donut algorithms on HRT, IM-3 WAV, SEG, IM-10, YST, VOW and M-L datasets.

5.3.5 Misclassified

On the Misclassified variant:

- stopping criteria: number of generations equal to 100 or training accuracy equal to 100%;
- $pred\% \in]0, 0.001]\%$

	HRT	IM-3	WAV	SEG	IM-10	YST	VOW	M-L
Training accuracy								
M3GP 50	93.1(3.2)	98.7(0.6)	89.6(0.6)	**96.4**(0.7)	**92.3**(0.8)	66.0(5.0)	**100**(0.0)	**100**(0.6)
Misc	**100**(0.5)	**100**(0.0)	**91.7**(0.3)	**96.8**(0.4)	**92.8**(0.7)	**69.4**(2.9)	**100**(0.0)	**100**(0.0)
Test accuracy								
M3GP 50	**77.8**(4.0)	**94.3**(2.6)	**84.7**(0.9)	**95.1**(1.0)	**91.1**(0.9)	**55.6**(3.8)	**94.3**(2.3)	**60.6**(5.9)
Misc	51.9(9.5)	92.3(4.2)	83.9(0.9)	94.0(0.9)	90.0(1.2)	52.6(2.9)	12.8(15.7)	**59.7**(6.3)
# of dimensions								
M3GP 50	12.5(1-20)	6(3-13)	27(22-32)	11(6-18)	18(9-23)	10(6-20)	20(15-24)	11(8-32)
Misc	24(2-31)	5(2-10)	32(26-37)	10(4-19)	18(10-22)	16.5(4-14)	18(16-20)	11(7-12)

Table 5.36: Comparison between M3GP and the Misclassified variant of GSI-M3GP.

Table 5.36 shows that the Misclassified variant of M3GP is not competitive with M3GP.

5.3.6 Misclassified-hyperrectangle

On the Misclassified-hyperrectangle variant (MHR):

- stopping criteria: number of generations equal to 100 or training accuracy equal to 100%;
- $pred\% \in]0, 0.001]\%$

Table 5.37 shows that the MHR variant of GSI-M3GP is only competitive with M3GP on WAV, SEG, IM-10 and M-L datasets. For the other datasets, the test accuracy values of MHR are significantly worse than the test accuracy values of M3GP.

	HRT	IM-3	WAV	SEG	IM-10	YST	VOW	M-L
Training accuracy								
M3GP 50	**93.1**(3.2)	98.7(0.6)	89.6(0.6)	**96.4**(0.7)	**92.3**(0.8)	**66.0**(5.0)	**100**(0.0)	**100**(0.6)
MHR	**94.2**(2.7)	**100**(0.5)	**90.5**(0.4)	**96.5**(0.6)	**92.5**(0.7)	**66.3**(3.3)	**100**(0.0)	**100**(0.1)
Test accuracy								
M3GP 50	**77.8**(4.0)	**94.3**(2.6)	**84.7**(0.9)	**95.1**(1.0)	**91.1**(0.9)	**55.6**(3.8)	**94.3**(2.3)	**60.6**(5.9)
MHR	56.2(15.6)	92.8(4.2)	**84.2**(0.9)	**95.2**(6.6)	**91.5**(0.8)	53.0(7.6)	14.3(23.1)	**60.6**(6.3)
# of dimensions								
M3GP 50	12.5(1-20)	6(3-13)	27(22-32)	11(6-18)	18(9-23)	10(6-20)	20(15-24)	11(8-32)
MHR	15(7-24)	8.5(4-21)	32(23-38)	9.5(5-16)	16(5-22)	15(8-27)	18(16-22)	10(8-12)

Table 5.37: Comparison between M3GP and the MHR variant of GSI-M3GP.

The median, minimum and maximum depth values of depth found for MHR are stored in table 5.38. Interestingly, for IM-3, IM-10, YST, VOW and M-L the median depth values are much smaller than on previous variants (comparing, for example, with table 5.33).

	HRT	IM-3	WAV	SEG	IM-10	YST	VOW	M-L
MHR	564	218	392	509.5	436.5	461	95	41
	(217-615)	(48-608)	(343-469)	(364-601)	(290-552)	(393-608)	(47-191)	(6-524)

Table 5.38: GSI-M3GP MHR variant depth values.

For HRT, IM-3, YST and VOW (the datasets for which MHR is not competitive with M3GP) the *mutation percentages* are smaller when compared, for example, with the Donut variant (see tables 5.39 and 5.34), although they are still higher than the *add branch percentages* (except for VOW). It is obvious that the mutation operator is not working well on this variant of GSI-M3GP.

		HRT	IM-3	WAV	SEG	IM-10	YST	VOW	M-L
MHR	M	69%	61%	58%	76%	70.79%	70%	40%	35%
	A	24%	30%	40%	18%	26.65%	24%	57%	21%
	R	7%	8%	2%	6%	4.56%	6%	3%	4%

Table 5.39: GSI-M3GP MHR variant operators' percentages.

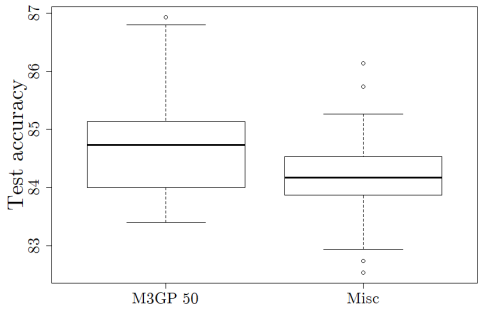
As expected, the average and minimum number of operations for MHR (see table 5.40) are smaller than 90 for some datasets - it means that, on some runs of the algorithm, from a certain

generation on, there is no improvement.

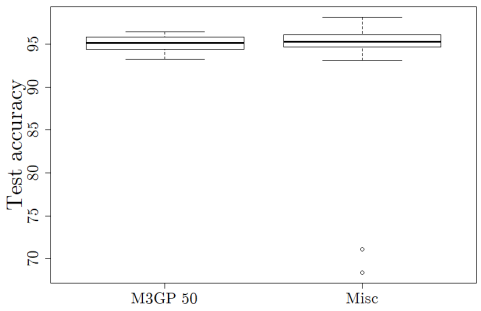
	HRT	IM-3	WAV	SEG	IM-10	YST	VOW	M-L
MHR	94.9 (65-100)	58.3 (17-100)	97.6 (92-100)	94.2 (74-99)	89.5 (74-97)	92.5 (74-100)	35.3 (29-51)	22.0 (10-94)

Table 5.40: GSI-M3GP MHR variant number of operations.

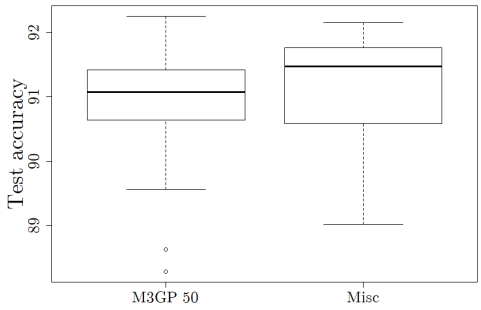
Figure 5.33 shows boxplots with last generation median accuracies of M3GP and MHR on the datasets for which MHR is competitive with M3GP. On WAV, although the median test accuracy values are similiar, the maximum value of test accuracy reached with MHR is much smaller than the maximum value of test accuracy for M3GP (ignoring outliers), as it can be seen in figure 5.33a. Ignoring outliers, on SEG, MHR is able to reach higher test accuracy values than M3GP (see figure 5.33b). On IM-10 and M-L the range of test accuracy values is approximately the same for M3GP and MHR (as expressed in figures 5.33c,d).



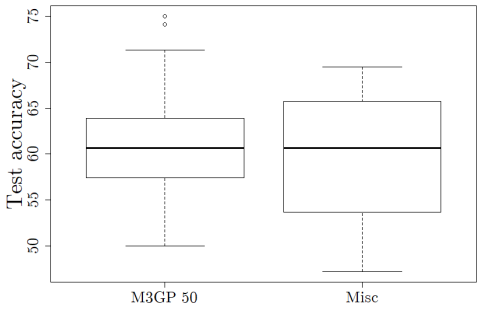
(a) WAV - Test.



(b) SEG - Test.



(c) IM-10 - Test.



(d) M-L - Test.

Figure 5.33: Comparison between last generation individuals' accuracies of M3GP 50 and MHR algorithms on WAV, SEG, IM-10 and M-L datasets.

5.3.7 Misclassified-hyperellipse

On the Misclassified-hyperellipse variant (MHE):

- stopping criteria: number of generations equal to 100 or training accuracy equal to 100%;
- $pred\% \in]0, 0.001]\%$.

Both of the MHE variants of GSI-M3GP are only competitive with M3GP for IM-3, SEG, IM-10, YST and M-L datasets (see table 5.41).

	HRT	**IM-3**	**WAV**	**SEG**	**IM-10**	**YST**	**VOW**	**M-L**
Training accuracy								
M3GP 50	**93.1**(3.2)	**98.7**(0.6)	89.6(0.6)	**96.4**(0.7)	92.3(0.8)	**66.0**(5.0)	**100**(0.0)	**100**(0.6)
MHE $\chi^2_{99\%}$	**93.4**(2.2)	**99.1**(0.7)	90.5(0.4)	**96.4**(0.6)	**92.7**(0.9)	**65.4**(3.2)	**100**(0.0)	**100**(0.3)
MHE $\chi^2_{10\%}$	**93.9**(2.1)	**99.1**(0.8)	**90.8**(0.4)	**96.9**(0.6)	**92.3**(0.8)	**65.3**(3.4)	**100**(0.0)	**100**(0.7)
Test accuracy								
M3GP 50	**77.8**(4.0)	**94.3**(2.6)	**84.7**(0.9)	**95.1**(1.0)	**91.1**(0.9)	**55.6**(3.8)	**94.3**(2.3)	**60.6**(5.9)
MHE $\chi^2_{99\%}$	51.2(17.2)	**94.3**(2.7)	**84.3**(0.8)	**94.6**(5.0)	**91.2**(0.9)	**52.9**(3.0)	63.5(23.3)	**59.3**(6.9)
MHE $\chi^2_{10\%}$	51.2(15.6)	**94.8**(2.7)	83.9(1.0)	**95.0**(5.6)	**91.2**(0.9)	**54.8**(3.0)	62.8(24.3)	**60**.2(6.9)
# of dimensions								
M3GP 50	12.5(1-20)	6(3-13)	27(22-32)	11(6-18)	18(9-23)	10(6-20)	20(15-24)	11(8-32)
MHE $\chi^2_{99\%}$	15(5-26)	9(4-15)	32(28-39)	9(5-14)	15.5(6-24)	14.5(11-29)	20(17-22)	11(9-13)
MHE $\chi^2_{10\%}$	16(10-27)	8(4-15)	33(26-40)	10(6-15)	15(8-20)	14(11-22)	19(17-23)	11(8-13)

Table 5.41: Comparison between M3GP, MHE $\chi^2_{99\%}$ and MHE $\chi^2_{10\%}$.

The median depth values for HRT, VOW and M-L (see table 5.42) are smaller than the depth values found for competitive variants of GSI-M3GP (see, for example, 5.33).

	HRT	**IM-3**	**WAV**	**SEG**	**IM-10**	**YST**	**VOW**	**M-L**
MHE $\chi^2_{99\%}$	482.5 (321-650)	520.5 (91-671)	388.5 (301-434)	525 (462-608)	444 (377-573)	503 (251-587)	113 (63-462)	83 (6-622)
MHE $\chi^2_{10\%}$	434.5 (328-587)	545 (69-629)	389 (322-448)	508.5 (387-625)	461 (350-587)	496 (357-579)	130 (56-533)	55 (6-636)

Table 5.42: Comparison between MHE $\chi^2_{99\%}$ and MHE $\chi^2_{10\%}$ depth values.

The same happens with the operators' percentages - the *mutation* percentages are smaller than the values found for competitive variants of GSI-M3GP. Between variants, the values do not vary a lot.

		HRT	IM-3	WAV	SEG	IM-10	YST	VOW	M-L
MHE $\chi^2_{99\%}$	*M*	71%	72%	56%	77.5%	70%	74%	44%	43%
	A	23%	20%	41%	16.9%	25%	22%	52%	52%
	R	6%	8%	3%	5.6%	5%	4%	4%	5%
MHE $\chi^2_{10\%}$	*M*	66%	74%	57%	75%	72%	73%	45%	41%
	A	26%	19%	41%	19%	23%	22%	52%	54%
	R	8%	7%	2%	6%	5%	5%	3%	5%

Table 5.43: Comparison between MHE $\chi^2_{99\%}$ and MHE $\chi^2_{10\%}$ operators' percentages.

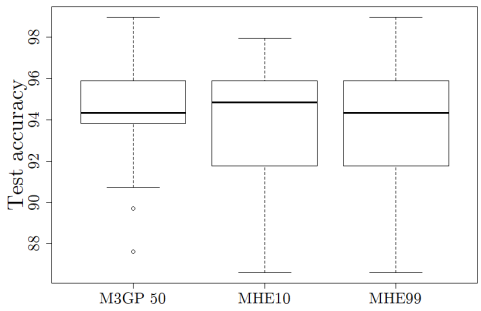
Table 5.44 shows the number of operations, which are also smaller than wanted.

	HRT	IM-3	WAV	SEG	IM-10	YST	VOW	M-L
MHE $\chi^2_{99\%}$	97.0 (85-100)	87.0 (26-100)	97.4 (93-100)	97.1 (85-100)	91.1 (83-99)	93.9 (83-100)	46.0 (31-100)	46.4 (10-100)
MHE $\chi^2_{10\%}$	94.1 (72-100)	85.8 (23-100)	97.8 (94-100)	96.3 (90-100)	91.4 (80-98)	93.7 (77-100)	45.0 (28-99)	42.6 (11-100)

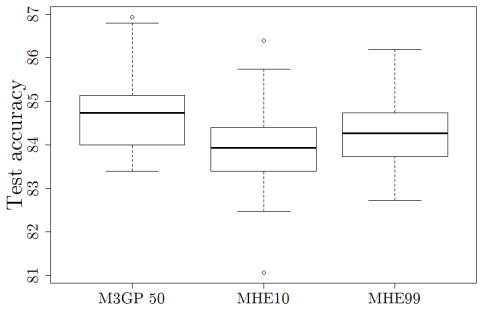
Table 5.44: Comparing the number of operations between MHE $\chi^2_{99\%}$ and MHE $\chi^2_{10\%}$.

Regarding last generation individuals' test accuracies (see figure 5.34):

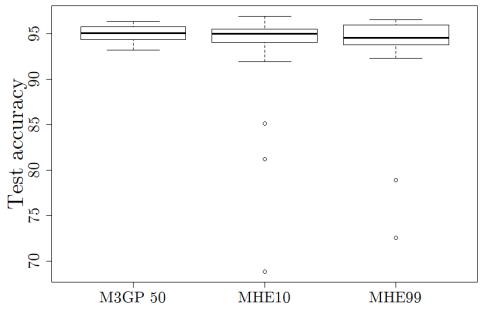
- they do not differ a lot between variants for IM-3 (although the MHE variants reach lower values than M3GP) and SEG;
- On WAV, M3GP reaches higher test accuracy values;
- MHE $\chi^2_{99\%}$ can produce higher test accuracy values than the other two algorithms on IM-10;
- MHE $\chi^2_{10\%}$ can produce higher test accuracy values than the other two algorithms on M-L.



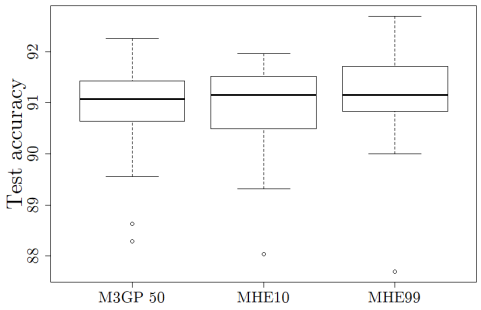
(a) IM-3 - Test.



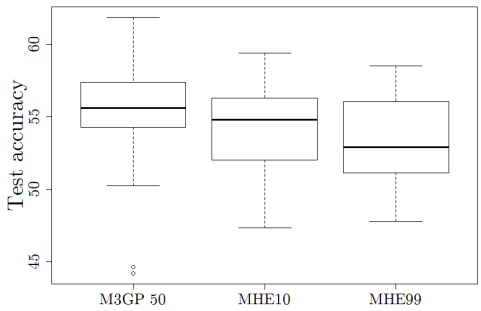
(b) WAV - Test.



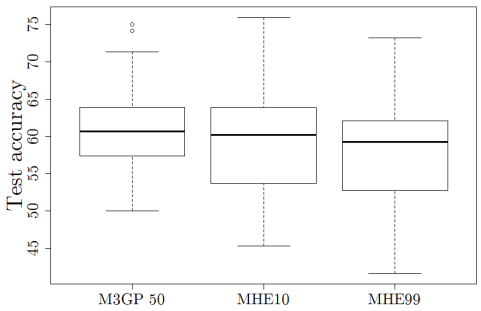
(c) SEG - Test.



(d) IM-10 - Test.



(e) YST - Test.



(f) M-L - Test.

Figure 5.34: Comparison between last generation individuals' test accuracies of M3GP 50, MHE $\chi^2_{99\%}$ and MHE $\chi^2_{10\%}$ algorithms on IM-3, WAV, SEG, IM-10, YST and M-L datasets.

5.3.8 Around Misclassified

On the Around-Misclassified variant (AM):

- stopping criteria: number of generations equal to 100 or training accuracy equal to 100%;
- $pred\% \in]0, 0.001]\%$

σ-variant

Table 5.45 shows a comparison between the AM-*sigma* variants of GSI-M3GP on training accuracy, test accuracy and number of nodes. As expressed by the values in bold, there is no statistical difference between the algorithms on training accuracy (except for M3GP, on WAV, which performs worse than the others) and test accuracy (except for AM2σ on WAV, performing worse than the other three).

	HRT	**IM-3**	**WAV**	**SEG**	**IM-10**	**YST**	**VOW**	**M-L**
Training accuracy								
M3GP 50	**93.1**(3.2)	**98.7**(0.6)	89.6(0.6)	**96.4**(0.7)	**92.3**(0.8)	**66.0**(5.0)	**100**(0.0)	**100**(0.6)
AM0.5σ	**94.2**(1.6)	**99.1**(0.8)	**90.7**(0.4)	**96.5**(0.7)	**92.5**(0.7)	**66.0**(2.9)	**100**(0.0)	**100**(0.4)
AM1σ	**92.7**(1.5)	**98.7**(0.8)	**90.6**(0.5)	**96.7**(0.6)	**92.4**(0.7)	**65.8**(2.7)	**100**(0.0)	**100**(0.6)
AM2σ	**94.2**(1.6)	**98.9**(0.8)	**90.9**(0.5)	**96.5**(05)	**92.3**(0.5)	**65.9**(4.2)	**100**(0.0)	**100**(0.8)
Test accuracy								
M3GP 50	**77.8**(4.0)	**94.3**(2.6)	**84.7**(0.9)	**95.1**(1.0)	**91.1**(0.9)	**55.6**(3.8)	**94.3**(2.3)	**60.6**(5.9)
AM0.5σ	**79.0**(3.9)	**94.3**(2.7)	**84.3**(1.0)	**94.9**(0.4)	**91.1**(0.9)	**55.9**(2.3)	**92.9**(1.8)	**60.6**(7.2)
AM1σ	**79.0**(4.7)	**95.9**(2.7)	**84.3**(1.0)	**95.1**(0.9)	**91.1**(0.8)	**56.3**(2.0)	**93.8**(2.5)	**63.9**(6.2)
AM2σ	**77.8**(3.4)	**92.8**(3.2)	83.9(1.0)	**94.9**(1.1)	**91.1**(0.7)	**54.9**(2.7)	**94.3**(2.6)	**60.6**(7.9)
# of dimensions								
M3GP 50	12.5(1-20)	6(3-13)	27(22-32)	11(6-18)	18(9-23)	10(6-20)	20(15-24)	11(8-32)
AM0.5σ	16.5(13-24)	9(3-18)	32.5(27-37)	10(5-16)	15(8-20)	14(7-21)	20(17-22)	11(9-12)
AM1σ	17.5(10-22)	8(4-14)	32(26-37)	10(6-17)	15(9-21)	13(9-22)	20(17-24)	11(9-12)
AM2σ	16(7-24)	9(2-14)	35(27-39)	9.5(6-18)	15(11-22)	15(4-28)	19(16-25)	11(9-12)

Table 5.45: Comparison between M3GP, AM0.5σ, AM1σ and AM2σ.

Table 5.46 stores the depth values of the AM-σ variants. The values are similar between variants, for a specific dataset, except on the M-L dataset for which the median depth values of AM1σ and AM2σ are much smaller than the value for AM0.5σ. More than that, when comparing the values of AM1σ with HR]0,0.001]% (see table 5.21, the depth values for VOW and M-L are much smaller, while the number of dimensions remains the same (see tables 5.45 and 5.20). Although this does not necessarily imply that the number of nodes of the AM1σ individuals is less than HR]0,0.001]% individuals' number of nodes, it is good to have trees with much lower depth values that produce competitive test and train accuracy values.

	HRT	IM-3	WAV	SEG	IM-10	YST	VOW	M-L
AM0.5σ	507	569.5	396.6	528.5	475	524	151.5	209
	(407-581)	(393-664)	(322-455)	(420-629)	(377-573)	(433-601)	(56-309)	(6-629)
AM1σ	501.5	552	378	515	482.5	517	155	69
	(420-580)	(318-650)	(308-457)	(421-609)	(391-573)	(246-580)	(57-371)	(6-629)
AM2σ	514.5	552	371	552.5	475	501	168.5	45
	(448-608)	(181-657)	(287-434)	(448-632)	(390-559)	(369-601)	(32-336)	(6-615)

Table 5.46: Comparison between AM0.5σ, AM1σ and AM2σ depth values.

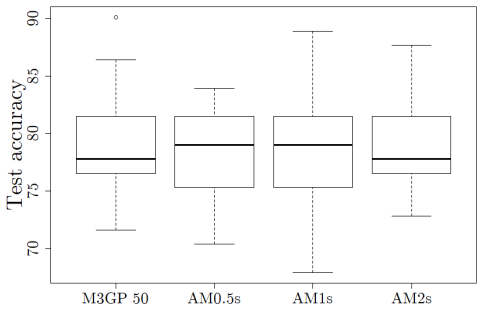
Table 5.47 shows operators' percentages for the three AMσ variants. There is no big distinction between variants, but when comparing these variants to the HR]0,0.001]% (see table 5.22) the *mutation percentages* are much lower for the AMσ variants and almost equal to the *add branch percentages*. Again, as the results of accuracy show that the AMσ variants are competitive with M3GP, for some datasets it might not be bad to have a lower value of mutation percentage.

		HRT	IM-3	WAV	SEG	IM-10	YST	VOW	M-L
AM0.5σ	M	72.6%	80%	57.1%	77%	74%	77%	49%	53.2%
	A	22.6%	14%	40.5%	18%	22%	20%	48%	42.5%
	R	4.8%	6%	2.4%	5%	4%	3%	3%	4.3%
AM1σ	M	72.9%	78.7%	57%	76%	73%	75%	47%	45%
	A	22.5%	14.6%	40%	19%	23%	21%	49%	51%
	R	4.6%	6.7%	3%	5%	4%	4%	4%	4%
AM2σ	M	74.1%	75.75%	54%	79%	71%	74%	47%	38%
	A	21.5%	16.73%	43%	16%	24%	22%	48%	58%
	R	4.4%	7.52%	3%	5%	5%	4%	5%	4%

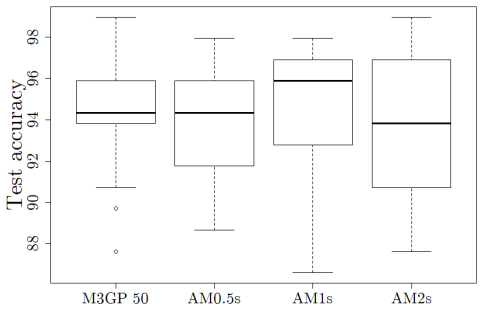
Table 5.47: Comparison between AM0.5σ, AM1σ and AM2σ operators' percentages.

	HRT	IM-3	WAV	SEG	IM-10	YST	VOW	M-L
AM0.5σ	98.7	99.4	97.5	97.4	92.6	96.2	47	55.9
	(96-100)	(94-100)	(94-100)	(94-100)	(84-98)	(93-100)	(31-89)	(10-100)
AM1σ	98.5	97.7	97.6	97.7	92.8	96.1	48.4	46.9
	(94-100)	(62-100)	(93-100)	(92-100)	(84-99)	(89-100)	(30-76)	(11-100)
AM2σ	98.9	97.1	97.2	98.1	93.6	96.1	51.1	37.1
	(90-100)	(39-100)	(93-100)	(95-100)	(86-97)	(91-100)	(27-80)	(11-100)

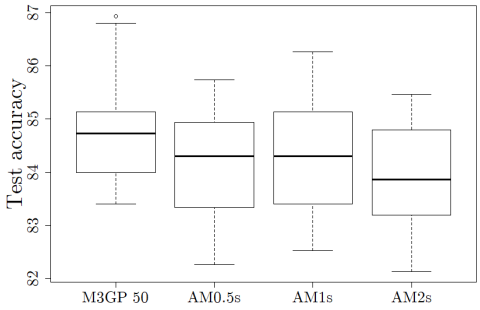
Table 5.48: Comparing the number of operations between AM0.5σ, AM1σ and AM2σ.



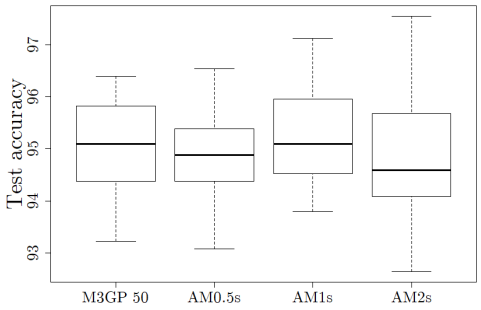
(a) HRT - Test.



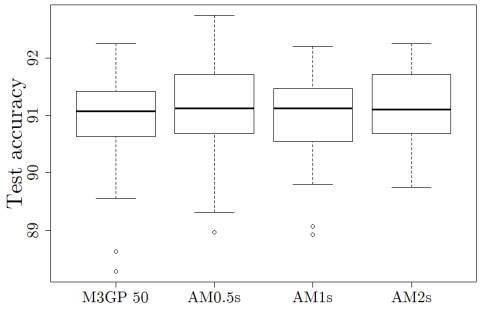
(b) IM-3 - Test.



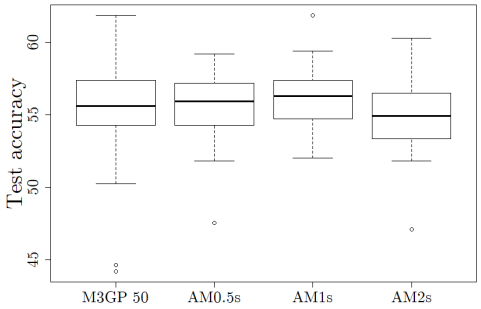
(c) WAV - Test.



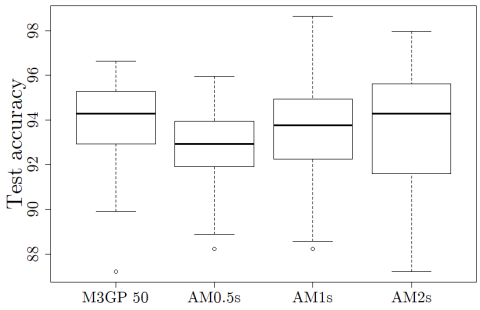
(d) SEG - Test.



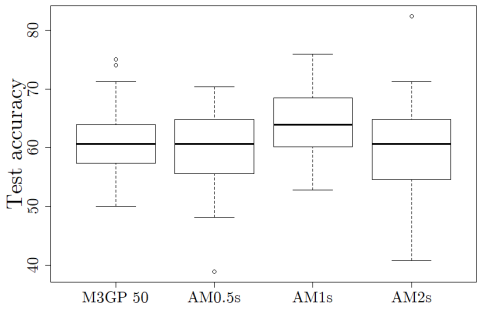
(e) IM-10 - Test.



(f) YST - Test.



(g) VOW - Test.



(h) M-L - Test.

Figure 5.35: Comparison between last generation individuals' accuracies of M3GP 50, AM0.5σ, AM1σ and AM2σ algorithms on HRT, IM-3, WAV, SEG, IM-10, YST, VOW and M-L datasets.

Table 5.48 shows that the average number of operations is bigger than 90 for HRT, IM-3, WAV, SEG, IM-10 and YST. For VOW, the average number of operations is around 50, and the maximum number of operations does not even reach 90. The median training accuracy on VOW is 100% and it is reached very early, and so it means that after a number of generations whatever operation applied to the individuals decreases the training accuracy. A similar reasoning can be thought for M-L.

Figure 5.35 shows boxplots with the last generation test accuracy values. Summarizing:

- On HRT, VOW and M-L, AM1σ is the approach reaching to higher values of test accuracy, but for the two first datasets it is also able to reach lower values of test accuracy (when comparing to M3GP);
- AM2σ is able to reach high values like M3GP on IM-3, but has a wider range of values, also reaching lower values of accuracy;
- The best approach on WAV and YST is M3GP, achieving higher values of test accuracy and not achieving as lower values as the AMσ variants.
- AM1σ and AM2σ are the approaches reaching the higher values of test accuracy on SEG;
- On IM-10, M3GP, AM1σ and AM2σ perform similarly on the test set, but AM0.5σ is able to reach slightly higher values and also slightly lower values.

%-variant

Table 5.49 shows the comparison between M3GP and the AM% variants training and test accuracy values. Additionally, the number of dimensions is presented.

Looking at the values in bold on training accuracy, AM25% is better or equal to all the other variants of every dataset. Regarding test accuracy, all the approaches perform without significant distinction.

	HRT	IM-3	WAV	SEG	IM-10	YST	VOW	M-L
Training accuracy								
M3GP 50	**93.1**(3.2)	**98.7**(0.6)	89.6(0.6)	96.4(0.7)	92.3(0.8)	**66.0**(5.0)	**100**(0.0)	**100**(0.6)
AM10%	**93.7**(2.7)	**98.7**(0.9)	**90.7**(0.3)	96.4(0.6)	**92.7**(0.8)	**65.1**(1.9)	**100**(0.0)	**100**(0.5)
AM25%	**93.7**(1.9)	**98.7**(0.7)	**90.6**(0.5)	**96.9**(0.5)	**92.6**(0.8)	**65.3**(3.7)	**100**(0.0)	**100**(0.6)
Test accuracy								
M3GP 50	**77.8**(4.0)	**94.3**(2.6)	**84.7**(0.9)	**95.1**(1.0)	**91.1**(0.9)	**55.6**(3.8)	**94.3**(2.3)	**60.6**(5.9)
AM10%	**77.8**(5.1)	**94.8**(3.6)	**84.2**(0.7)	**94.9**(0.9)	**91.1**(0.9)	**56.1**(2.2)	**93.4**(2.5)	**61.1**(6.4)
AM25%	**77.8**(3.1)	**94.8**(2.1)	**84.3**(0.9)	**94.9**(0.8)	**91.0**(0.9)	**55.2**(2.4)	**93.3**(2.4)	**60.6**(6.8)
# of dimensions								
M3GP 50	12.5(1-20)	6(3-13)	27(22-32)	11(6-18)	18(9-23)	10(6-20)	20(15-24)	11(8-32)
AM10%	16(1-22)	9(4-13)	33.5(27-38)	10(5-13)	16(7-2)	13.5(8-19)	20(17-23)	11(8-13)
AM25%	15(7-22)	9(3-16)	31.5(24-36)	11(6-19)	16.5(11-23)	15(10-30)	20(16-26)	11(9-13)

Table 5.49: Comparison between M3GP, AM10% and AM25%.

Looking at table 5.50, one can see that the median depth values are, except on WAV and IM-10, smaller for AM25% than for AM10%.

	HRT	IM-3	WAV	SEG	IM-10	YST	VOW	M-L
AM10%	524.5	545	371	535.5	471.5	531	154.5	177.5
	(444-692)	(160-643)	(301-455)	(420-610)	(364-587)	(461-622)	(72-378)	(6-609)
AM25%	511	538	395.5	515	475.5	519.5	133	72.5
	(427-587)	(419-608)	(322-483)	(421-617)	(365-573)	(336-587)	(58-539)	(6-622)

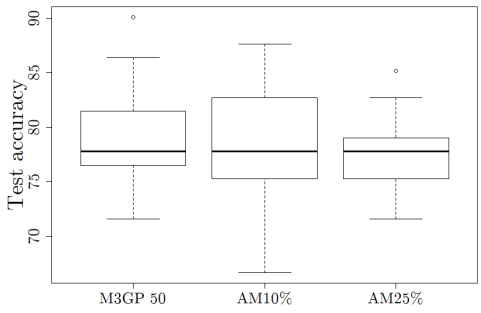
Table 5.50: Comparison AM10% and AM25% depth values.

		HRT	IM-3	WAV	SEG	IM-10	YST	VOW	M-L
AM10%	M	76%	79%	55%	78%	71.5%	78%	48%	50%
	A	20%	15%	42%	17%	24.6%	19%	49%	44%
	R	4%	6%	3%	5%	3.9%	3%	3%	6%
AM25%	M	73%	77%	57.6%	74%	70%	72.6%	45.5%	43%
	A	22%	16%	39.9%	20%	25%	22.8%	50.9%	52%
	R	5%	7%	2.5%	6%	5%	4.6%	3.6%	5%

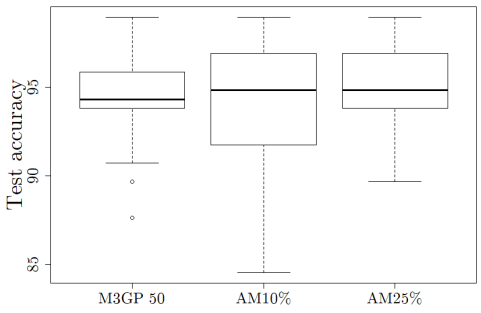
Table 5.51: Comparison AM10% and AM25% operators' percentages.

	HRT	IM-3	WAV	SEG	IM-10	YST	VOW	M-L
AM10%	99.0	96.7	96.9	96.9	92.5	96.3	48.6	54.4
	(96-100)	(42-100)	(92-100)	(92-100)	(85-98)	(91-100)	(31-98)	(11-100)
AM25%	98.5	99.2	97.5	97.3	94.3	96.8	48.6	47.3
	(96-100)	(96-100)	(93-100)	(92-100)	(84-100)	(92-100)	(28-100)	(10-100)

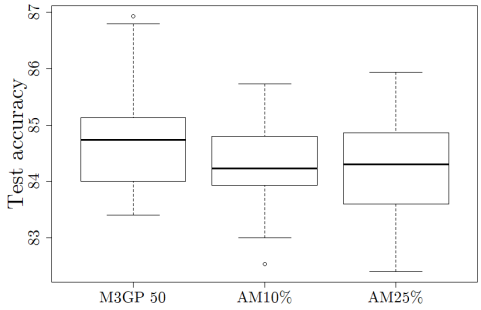
Table 5.52: Comparing the number of operations between AM10% and AM25% depth values.



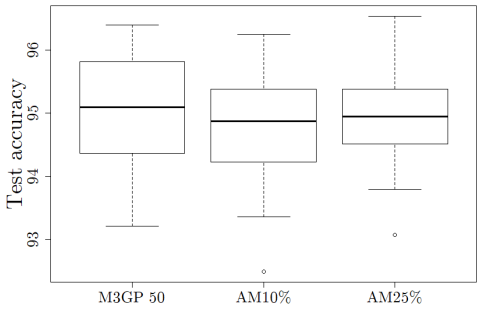
(a) HRT - Test.



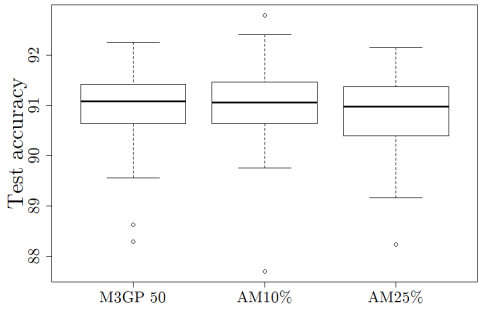
(b) IM-3 - Test.



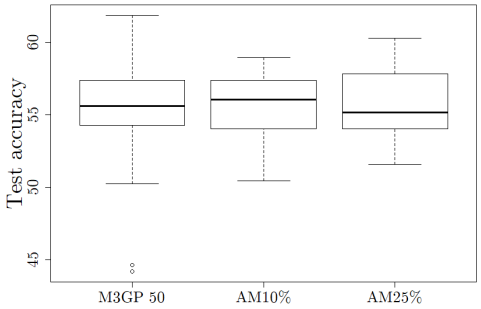
(c) WAV - Test.



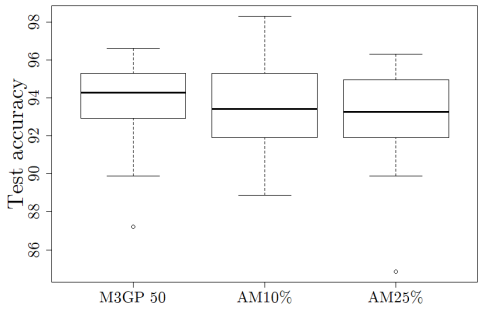
(d) SEG - Test.



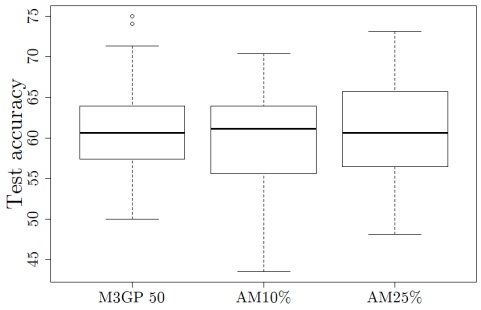
(e) IM-10 - Test.



(f) YST - Test.



(g) VOW - Test.



(h) M-L - Test.

Figure 5.36: Comparison between last generation individuals' accuracies of M3GP 50, AM10% and AM25% algorithms on HRT, IM-3, WAV, SEG, IM-10, YST and M-L datasets.

Table 5.51 shows that the *mutation percentage* values are slightly higher for AM10% than for AM25% (except on WAV).

As for the σ-variant, table 5.52 shows that the average number of operations is bigger than 90 for HRT, IM-3, WAV, SEG, IM-10 and YST, but not for VOW na M-L.

Last generation test accuracies can be seen in the boxplots of figure 5.36, and the following conclusions can be taken:

- AM10% can reach higher test accuracy values than M3GP, but it can also reach lower values on HRT and VOW (ignoring outliers);
- On IM-3, all the approaches reach the same maximum test accuracy values but AM10% also reaches much lower values than the other two approaches;
- M3GP is the best approach on WAV and YST;
- All the approaches perform similarly on SEG and IM-10;
- Although, on M-L, the median test accuracy of AM10% is higher than for the other two, AM10% also reaches lower values of accuracy than the others, and it is not able to achieve the same high values as the others; AM25% performs similarly to M3GP.

5.3.9 Correct-Misclassified distances

On the Correct-Misclassified distances variant (CMD):

- stopping criteria: number of generations equal to 100 or training accuracy equal to 100%;
- $pred\% \in]0, 0.001]\%$

Variant 1:

Table 5.53 shows a comparison between M3GP and the CMD-1 variants of GSI-M3GP. Regarding training accuracy, there is no significant difference between CMD AVG-A (where AVG stands for average and A stands for All), CDM AVG-C (where C stands for Class), CMD MED-A (where MED stands for median) and CMD MED-C on all datasets (except CMD AVG-C on YST). M3GP, CMD AVG-C, CMD MED-A and CMD MED-C produce test accuracy values without statistical difference.

	HRT	**IM-3**	**WAV**	**SEG**	**IM-10**	**YST**	**VOW**	**M-L**
Training accuracy								
M3GP 50	**93.1**(3.2)	**98.7**(0.6)	89.6(0.6)	**96.4**(0.7)	**92.3**(0.8)	**66.0**(5.0)	**100**(0.0)	**100**(0.6)
CMD AVG-A	**93.9**(2.9)	**99.1**(0.7)	**90.7**(0.5)	**96.9**(0.6)	**92.4**(0.6)	**66.6**(2.3)	**100**(0.0)	**100**(0.2)
CMD AVG-C	**93.7**(2.3)	**98.7**(0.7)	**90.6**(0.5)	**96.8**(0.4)	**92.5**(0.7)	65.3(3.0)	**100**(0.0)	**100**(0.8)
CMD MED-A	**93.7**(2.0)	**98.7**(0.7)	**90.7**(0.5)	**96.8**(0.6)	**92.5**(0.7)	**65.3**(3.4)	**100**(0.0)	**100**(0.3)
CMD MED-C	**93.9**(2.2)	**98.7**(2.5)	**90.7**(0.4)	**96.6**(0.5)	**92.2**(0.5)	**66.1**(1.6)	**100**(0.0)	**100**(0.4)
Test accuracy								
M3GP 50	**77.8**(4.0)	**94.3**(2.6)	**84.7**(0.9)	**95.1**(1.0)	**91.1**(0.9)	**55.6**(3.8)	**94.3**(2.3)	**60.6**(5.9)
CMD AVG-A	**77.8**(4.5)	**92.8**(2.6)	**84.4**(0.9)	**95.1**(5.0)	**91.3**(0.7)	54.1(2.2)	**93.6**(2.6)	**63.0**(6.8)
CMD AVG-C	**77.8**(4.8)	**94.8**(3.1)	**84.5**(0.9)	**95.2**(8.9)	**91.0**(0.9)	**55.4**(3.4)	**92.6**(2.2)	**61.1**(7.3)
CMD MED-A	**76.5**(4.2)	**93.8**(2.2)	**84.3**(1.0)	**95.2**(0.9)	**91.0**(0.8)	**56.1**(1.8)	**93.8**(2.4)	**60.2**(5.9)
CMD MED-C	**77.4**(4.7)	**93.3**(2.5)	**84.6**(1.0)	**95.0**(0.9)	**91.0**(13.2)	**55.6**(2.4)	**92.9**(1.8)	**61.6**(5.9)
# of dimensions								
M3GP 50	12.5(1-20)	6(3-13)	27(22-32)	11(6-18)	18(9-23)	10(6-20)	20(15-24)	11(8-32)
CMD AVG-A	18(5-23)	9(3-13)	32(20-38)	10(4-21)	16.5(10-21)	14.5(11-20)	21(16-26)	11(9-13)
CMD AVG-C	16(2-22)	9(4-15)	32(25-38)	9(7-15)	16(8-21)	13(4-21)	22.5(18-27)	11(8-12)
CMD MED-A	15.5(8-25)	8(4-13)	32.5(23-38)	10.5(5-16)	17(9-20)	14(9-24)	22(16-27)	11(9-13)
CMD MED-C	15(8-22)	7(3-12)	21(25-38)	10.5(5-17)	14(10-22)	14(10-21)	21(16-26)	11(9-13)

Table 5.53: Comparison between M3GP, CMD AVG-A, CMD AVG-C, CMD MED-A and CMD MED-C.

The individuals created with the different variants of CMD-1 have a similar number of median depth values, as expressed in table 5.54.

	HRT	**IM-3**	**WAV**	**SEG**	**IM-10**	**YST**	**VOW**	**M-L**
CMD AVG-A	499.5 (379-650)	552 (412-650)	378.5 (322-490)	505 (385-637)	475 (386-559)	512 (364-573)	441 (322-533)	538.5 (399-636)
CMD AVG-C	503 (436-643)	555.5 (377-650)	385 (308-442)	524.5 (406-606)	466 (300-587)	534.5 (436-629)	414 (336-511)	567.5 (287-636)
CMD MED-A	515.5 (434-650)	548.5 (446-635)	389.5 (336-497)	518.5 (420-610)	462 (347-573)	510 (399-594)	420.5 (343-532)	545 (370-629)
CMD MED-C	522 (434-608)	567 (472.629)	385 (322-476)	514.5 (428-609)	479.5 (321-559)	506.5 (426-608)	413 (332-525)	528.5 (405-643)

Table 5.54: Comparison between CMD AVG-A, CMD AVG-C, CMD MED-A and CMD MED-C depth values.

Also, the operators' percentages are not very distinct between variants, and the *mutation percentages* are always bigger than the *add branch percentages*.

		HRT	IM-3	WAV	SEG	IM-10	YST	VOW	M-L
CMD AVG-A	*M*	73%	78%	57%	75.65%	72%	74.33%	62%	75.3%
	A	22%	15%	41%	18.84%	24%	21.34%	30%	17.5%
	R	5%	7%	2%	5.51%	4%	4.33%	8%	7.2%
CMD AVG-C	*M*	73.63%	79%	57%	78%	72%	79%	60%	76%
	A	21.65%	15%	41%	17%	24%	18%	32%	17%
	R	4.72%	6%	2%	5%	4%	3%	8%	7%
CMD MED-A	*M*	74%	79%	57.5%	76%	71%	76%	60%	75.3%
	A	21%	14%	40.4%	18%	24%	20%	31%	17.4%
	R	5%	7%	2.1%	6%	5%	4%	9%	7.3%
CMD MED-C	*M*	75%	80%	57%	75%	73.7%	76%	61%	74%
	A	20%	14%	41%	19%	22.8%	20%	21%	18%
	R	5%	6%	2%	6%	4.5%	4%	8%	8%

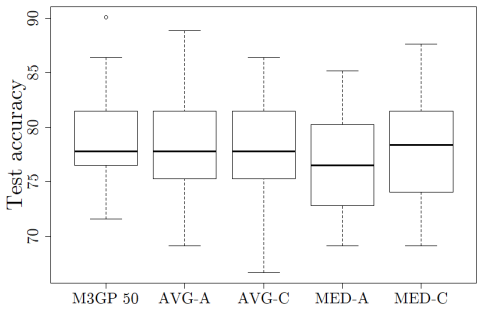
Table 5.55: Comparison between CMD AVG-A, CMD AVG-C, CMD MED-A and CMD MED-C operators' percentages.

From table 5.56, one concludes that there is improvement in almost every generation of these CMD-1 variants of GSI-M3GP.

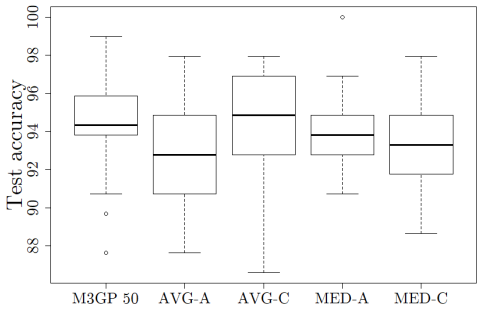
	HRT	IM-3	WAV	SEG	IM-10	YST	VOW	M-L
CMD AVG-A	98.8 (96-100)	99.1 (97-100)	97.2 (92-100)	95.9 (82-100)	93.3 (86-98)	95.0 (81-100)	99.1 (95-100)	99.4 (96-100)
CMD AVG-C V1	98.9 (96-100)	99.3 (97-100)	97.6 (94-100)	96.2 (90-100)	91.5 (74-98)	94.9 (87-99)	99.1 (95-100)	99.3 (96-100)
CMD MED-A	98.4 (94-100)	99.4 (98-100)	97.9 (94-100)	96.6 (92-100)	93.3 (82-99)	94.9 (84-100)	99.5 (97-100)	99.5 (98-100)
CMD MED-C	98.9 (96-100)	99.1 (96-100)	97.5 (95-100)	96.7 (93-100)	92.0 (72-99)	95.0 (85-100)	99.2 (97-100)	99.3 (96-100)

Table 5.56: Comparing the number of operations between CMD AVG-A, CMD AVG-C, CMD MED-A and CMD MED-C.

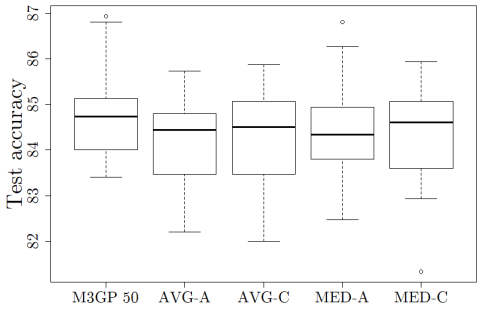
Last generation test accuracies are presented in the boxplots of figure 5.37. M3GP seems to be the best approach on WAV. On HRT, AVG-A is able to reach higher values than M3GP (ignoring outliers), but also slightly lower values. On IM-3, M3GP is the best approach. AVG-A and AVG-C perform similarly on SEG, and are able to reach higher values than M3GP (excluding outliers).



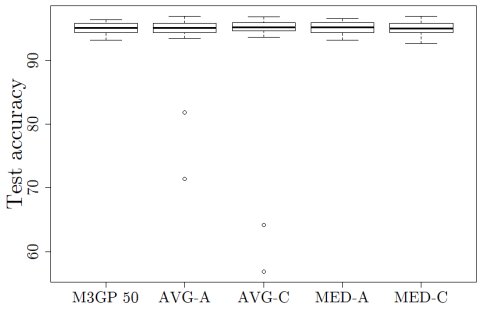
(a) HRT - Test.



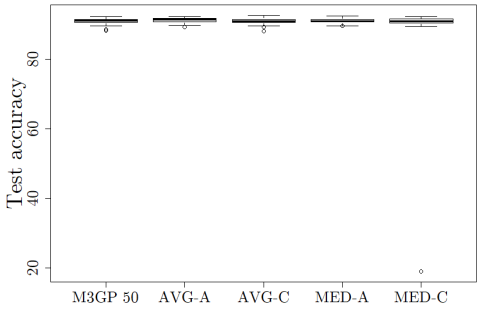
(b) IM-3 - Test.



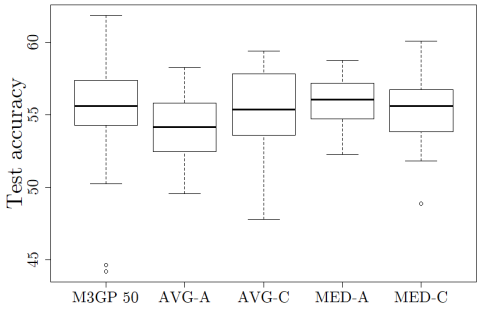
(c) WAV - Test.



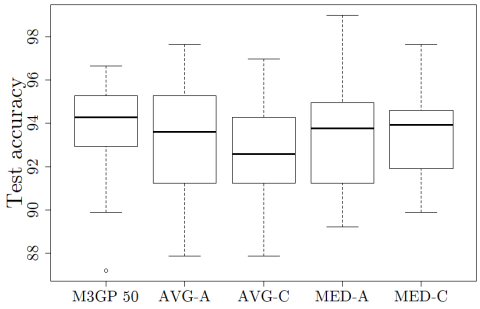
(d) SEG - Test.



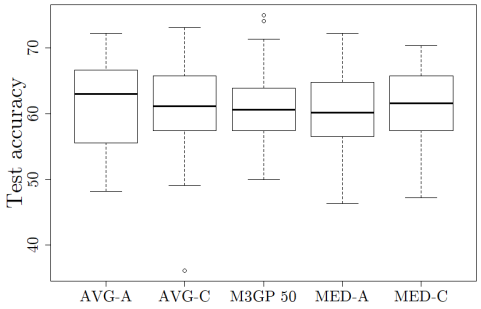
(e) IM-10 - Test.



(f) YST - Test.



(g) VOW - Test.



(h) M-L - Test.

Figure 5.37: Comparison between last generation individuals' accuracies of M3GP 50, CMD AVG-A, CMD AVG-C, CMD MED-A and CMD MED-C algorithms on HRT, IM-3, WAV, SEG, IM-10, YST and M-L datasets.

AVG-C is able to reach higher values of test accuracy than M3GP, on IM-10. M3GP is the approach reaching higher values on YST. MED-A is the approach reaching higher values of test accuracy on VOW. All the approaches perform similarly on M-L.

Variant 2:

The results of CDM-2 are stored in table 5.57. The values of training and test accuracy show that both of the new approaches are competitive with M3GP.

	HRT	**IM-3**	**WAV**	**SEG**	**IM-10**	**YST**	**VOW**	**M-L**
Training accuracy								
M3GP 50	**93.1**(3.2)	**98.7**(0.6)	89.6(0.6)	**96.4**(0.7)	**92.3**(0.8)	**66.0**(5.0)	**100**(0.0)	**100**(0.6)
CMD MED-A 25%	**93.7**(1.7)	**99.1**(0.6)	**90.5**(0.6)	**96.8**(0.8)	**92.6**(0.8)	**66.4**(1.8)	**100**(0.0)	**100**(0.3)
CMD MED-A 100%	**93.7**(2.4)	**98.7**(0.8)	**90.7**(0.5)	**96.6**(0.7)	**92.5**(0.5)	**65.3**(1.6)	**100**(0.0)	**100**(0.7)
Test accuracy								
M3GP 50	**77.8**(4.0)	**94.3**(2.6)	**84.7**(0.9)	**95.1**(1.0)	**91.1**(0.9)	**55.6**(3.8)	**94.3**(2.3)	**60.6**(5.9)
CMD MED-A 25%	**77.8**(1.7)	**94.8**(2.8)	**84.3**(1.0)	**95.4**(1.2)	**91.1**(1.2)	**55.2**(2.6)	**92.6**(2.1)	**61.6**(5.5)
CMD MED-A 100%	**76.5**(3.8)	**93.8**(3.4)	**84.2**(0.9)	**95.2**(5.7)	**91.1**(2.1)	**55.5**(2.4)	**93.1**(2.0)	**61.1**(5.4)
# of dimensions								
M3GP 50	12.5(1-20)	6(3-13)	27(22-32)	11(6-18)	18(9-23)	10(6-20)	20(15-24)	11(8-32)
CMD MED-A 25%	16(10-23)	8.5(4-14)	32(20-38)	10(6-17)	17(10-23)	14(10-20)	19.5(17-23)	11(9-12)
CMD MED-A 100%	15(8-21)	8(4-13)	33(24-36)	9.5(7-18)	14(11-21)	13(8-21)	20(16-23)	10.5(8-13)

Table 5.57: Comparison between M3GP, CMD MED-A 25% and CMD MED-A 100%.

The median depth of the individuals' trees is similar between variants. The biggest discrepancy is for M-L dataset, for which the median depth of the trees created with CMD MED-A 100% is more than twice the size of the ones created with CMD MED-A 25%.

	HRT	**IM-3**	**WAV**	**SEG**	**IM-10**	**YST**	**VOW**	**M-L**
CMD MED-A 25%	516	559	382	528.5	450.5	517	169	97.5
	(440-598)	(167-629)	(309-525)	(384-637)	(323-594)	(411-594)	(63-455)	(6-629)
CMD MED-A 100%	530	555.5	389	505	448	531	165	213.5
	(461-615)	(181-643)	(287-490)	(398-632)	(288-566)	(461-595)	(58-518)	(6-629)

Table 5.58: Comparison between CMD MED-A 25% and CMD MED-A 100% depth values.

The operators' percentages are also similar between variants, and for VOW and M-L the *mutation percentages* are almost equal to the *add branch percentages* (see table 5.59).

		HRT	IM-3	WAV	SEG	IM-10	YST	VOW	M-L
CMD MED-A 25%	*M*	74%	79%	58%	78%	69.9%	78%	49%	47%
	A	21%	15%	40%	17%	25.5%	19%	48%	49%
	R	5%	6%	2%	5%	4.6%	3%	3%	4%
CMD MED-A 100%	*M*	76%	79%	57.1%	75%	71%	78.5%	50%	50%
	A	20%	15%	40.5%	19%	25%	18.3%	46%	45%
	R	4%	6%	2.4%	6%	4%	3.2%	4%	5%

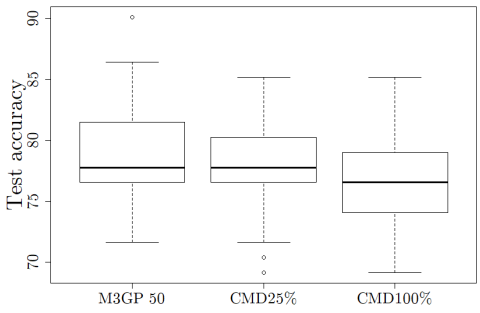
Table 5.59: Comparison between CMD MED-A 25% and CMD MED-A 100% operators' percentages.

The average number of operations is stored in table 5.60. The values are above 90 for both variants on HRT, IM-3, WAV, SEG IM-10, YST and below it on VOW and M-L. As such, for VOW and M-L, on every run, from a certain generation on there is no improvement by applying mutation or adding/removing branches from the tree.

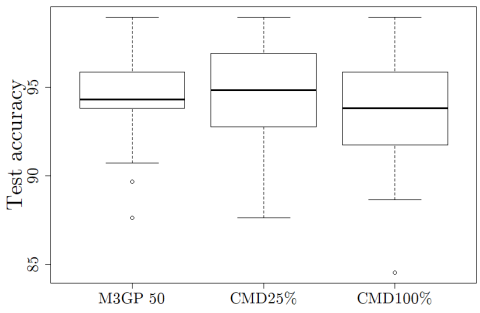
	HRT	IM-3	WAV	SEG	IM-10	YST	VOW	M-L
CMD MED-A 25%	98.7 (94-100)	95.2 (38-100)	97.1 (93-100)	95.6 (84-100)	90.6 (83-97)	94.1 (83-99)	49.8 (31-98)	43.9 (10-100)
CMD MED-A 100%	98.6 (96-100)	97.1 (48-100)	97.7 (94-100)	95.5 (81-100)	89.3 (89-97)	95.2 (86-99)	52.7 (30-99)	54.9 (10-100)

Table 5.60: Comparing the number of operations between CMD MED-A 25% and CMD MED-A 100%.

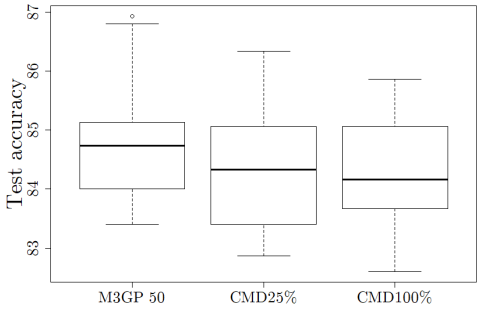
The test accuracy on the last generation (which might not always be generation 100) is now analysed according to the boxplots in figure 5.38. All the approaches perform similarly on HRT, although M3GP is able to reach slightly higher values than the other two. On IM-3, all the approaches reach the same maximum value of test accuracy. As for almost all the variants, M3GP is the best approach on WAV. CMD MED-A 25% reaches higher values of test accuracy than the other two approaches, on SEG. All the approaches perform similarly on IM-10, VOW and M-L (excluding outliers). On YST, CMD10% and M3GP reach the highest values of test accuracy and the minimum test value reached on CMD10% is higher than the minimum for M3GP.



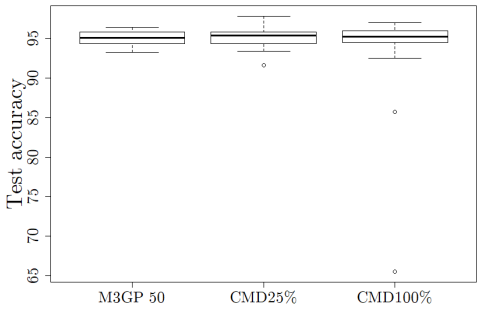
(a) HRT - Test.



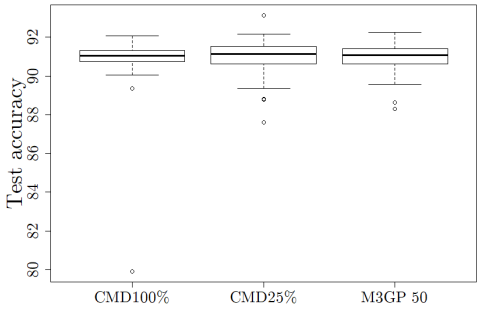
(b) IM-3 - Test.



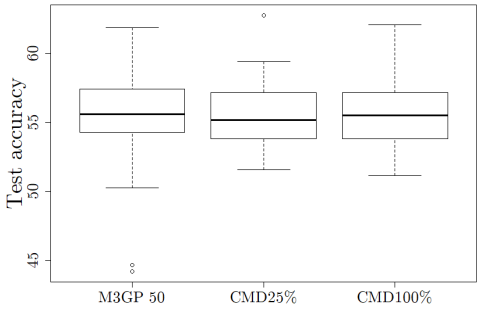
(c) WAV - Test.



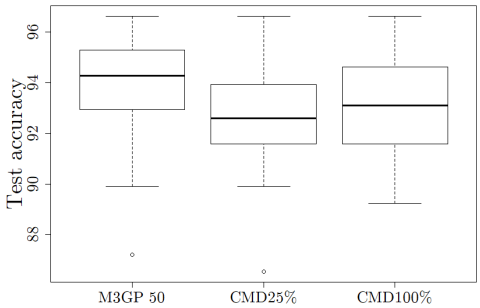
(d) SEG - Test.



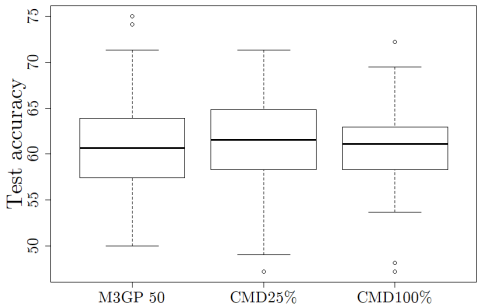
(e) IM-10 - Test.



(f) YST - Test.



(g) VOW - Test.



(h) M-L - Test.

Figure 5.38: Comparison between last generation individuals' accuracies of M3GP 50, CMD MED-A 25% and CMD MED-A 100% algorithms on HRT, IM-3, WAV, SEG, IM-10, YST and M-L datasets.

After analysing the results of all the different variants of *the moving point* and the different values of *pred*%, one can conclude that choosing the right value of *pred*% and the right variant of the *moving point* is crucial for GSI-M3GP. More thoroughly:

- low values of *pred*% (lower than 0.1%) work better than higher values;
- for certain values of *pred*%, if the *moving point* is chosen according to the variants HR, HE1 $\chi^2_{50\%}$, HE2 $\chi^2_{99\%}$, Donut, AM0.5σ, AM1σ, AM10%, AM25%, CMD AVG-C, CMD MED-A, CMD MED-C, CMD MED-A 25% and CMD MED-A 100%, then GSI-M3GP is competitive with M3GP;

It is also important to notice that, for WAV and YST, it is very difficult to find a variant of GSI-M3GP that can produce as high values of test accuracy as the ones that M3GP can reach.

For some variants of GSI-M3GP, the algorithm is able to achieve much higher values of training accuracy than M3GP. However, on those cases, the test accuracy of GSI-M3GP is typically lower than the test accuracy of M3GP (and the difference is statistically significant).

Chapter 6

GSI-M3GP-XO: *adding crossover*

This chapter introduces a new version of **GSI-M3GP**, referred in this chapter as **GSI-M3GP-XO**. This version of the algorithm considers one extra operator, a crossover operator, the operator *swap dimensions* (introduced in [15] and also explained in subsection 2.3.2). As such, **GSI-M3GP-XO** works just as **GSI-M3GP**, except that there is one more operator. Hence, four operators are considered to create offspring: *gsi-mutation*, *add new branch*, *remove existing branch* and *swap dimensions*[1]. Given these four operators, a question can be posed: *what is the probability of applying each one of the operators?*

To answer this question, two versions are considered:

- **EP**: all operators with the same probability of 0.25;
- **X & M**: 0.5 probability of crossover and 0.5 probability of mutation (and so, each one of the mutation operators - *gsi-mutation*, *add branch* and re*move branch* - have a probability of $0.5 \times \frac{1}{3}$ of being chosen).

The idea of introducing the *swap dimensions* operator is to, hopefully, increase the variability of the offspring population.

[1] Remember that *dimensions* and *branches* refer to the same thing.

6.1 Experimental Setup and Results

The running parameters of **GSI-M3GP-XO** are stored in table 6.1. These are very similar to the ones in 5.7. In fact, there is only one difference: the *predefined percentage* is set to be a random value of the set]0,0.001]%. The datasets used for the experimental analysis are the same as in chapter 3.

Runs	30
Population size	50 individuals
Stopping criteria	100 generations (or $accuracy_{Train} = 100\%$)
Initialization	6-depth Full Initialization[11]
Function set	+, -, ×, ÷ protected as in [11]
Terminal set	Ephemeral random constants [0,1]
Bloat control	No depth limit
Selection	Tournament of size 5
Elitism	Keep best individual
Predefined percentage, pred%	Random value in]0,0.001]%

Table 6.1: Running parameters of GSI-M3GP-XO.

Two different versions of the *moving point* were used to make the comparison between **GSI-M3GP** and **GSI-M3GP-XO**: the hyperrectangle version (HR) and the 1σ-version of around misclassified (AM1σ). Two versions were chosen, in order to make a fair comparison between **M3GP** and **GSI-M3GP-XO**. The previous referred versions (the chosen ones for the comparison) are versions satisfying the following:

- by using it, **GSI-M3GP** is competitive with **M3GP**;
- the *moving point* is chosen in different ways, and so, using different stopping criteria;
- the median depth values of the **GSI-M3GP** individuals' trees are very different.

Remembering the results in section 5.3:

- HR]0,0.001]% and AM1σ versions of **GSI-M3GP** are competitive with M3GP, on all problems;

- in HR]0,0.001]% the moving point is chosen as a random point inside the hyperrectangle that includes all the mapped training samples (the stopping criterion is 100 generations reached, and in AM1σ the moving point is chosen around a misclassified mapped training sample (more specifically one standard deviation way from the misclassified mapped training sample) and the stopping criteria are reaching 100 generations or the highest training accuracy equal to 100%;

- finally, looking at tables 5.20 and 5.46, the median depth values on VOW and M-L are much smaller on AM1σ than on HR]0,0.001]%.

As for the results presented in section 5.3, the **GSI-M3GP-XO** results are divided in 4 tables: **accuracy and dimension tables**, **depth tables**, **improvement tables** and **mutation-add-remove-crossover percentage tables** (see page 71 to review the explanation). Here, the percentage tables are mutation-add-remove-crossover percentage tables instead of mutation-add-remove percentage tables since the swap dimensions operator is also considered.

	HRT	**IM-3**	**WAV**	**SEG**	**IM-10**	**YST**	**VOW**	**M-L**
Training accuracy								
M3GP 50	**93.1**(3.2)	**98.7**(0.6)	89.6(0.6)	**96.4**(0.7)	**92.3**(0.8)	***66.0***(5.0)	***100***(0.0)	***100***(0.6)
HR	***93.9***(2.1)	**98.9**(0.9)	***90.6***(0.5)	**96.4**(0.6)	***92.5***(0.8)	**65.2**(2.3)	***100***(0.0)	***100***(1.0)
HR-XO EP	**93.7**(1.7)	**98.7**(1.2)	***90.6***(0.5)	**96.3**(0.5)	**92.3**(0.6)	**64.8**(1.5)	***100***(0.0)	***100***(0.4)
HR-XO X&M	92.6(2.9)	**98.7**(1.0)	90.1(0.4)	95.9(0.6)	91.8(0.7)	**64.6**(2.5)	***100***(0.2)	***100***(0.7)
AM1σ	**93.7**(1.5)	**98.7**(0.8)	**90.6**(0.5)	***96.7***(0.6)	**92.4**(0.7)	**65.8**(2.7)	***100***(0.0)	***100***(0.6)
AM1σ-XO EP	**92.9**(2.4)	***99.1***(0.8)	**90.4**(0.3)	**96.6**(0.6)	**92.1**(0.5)	**65.4**(3.2)	***100***(0.0)	***100***(0.5)
AM1σ-XO X&M	92.6(2.6)	**98.7**(0.9)	90.0(0.4)	95.9(1.0)	91.7(0.4)	**64.7**(2.8)	***100***(0.1)	***100***(0.5)
Test accuracy								
M3GP 50	**77.8**(4.0)	**94.3**(2.6)	***84.7***(0.9)	**95.1**(1.0)	***91.1***(0.9)	**55.6**(3.8)	***94.3***(2.3)	**60.6**(5.9)
HR	***79.0***(4.1)	**94.3**(2.8)	**84.5**(0.8)	**95.2**(1.1)	***91.1***(0.9)	***56.3***(2.4)	**93.8**(1.7)	**59.3**(6.6)
HR-XO EP	**77.2**(5.9)	**94.3**(2.6)	**84.4**(1.0)	**94.9**(0.8)	***91.1***(0.7)	**54.8**(1.6)	**93.9**(1.9)	**61.6**(6.1)
HR-XO X&M	***79.0***(4.1)	**93.8**(2.9)	**84.4**(1.0)	**94.9**(0.9)	**90.6**(0.8)	***56.3***(1.8)	**93.9**(2.2)	**63.0**(6.5)
AM1σ	***79.0***(4.7)	***95.9***(2.7)	**84.3**(1.0)	**95.1**(0.9)	***91.1***(0.8)	***56.3***(2.0)	**93.8**(2.5)	***63.9***(6.2)
AM1σ-XO EP	***79.0***(4.9)	**94.8**(2.9)	**84.5**(0.8)	***95.4***(1.0)	**90.9**(2.8)	**56.2**(2.2)	**93.3**(2.1)	59.7(7.2)
AM1σ-X&M	**78.4**(4.1)	**93.8**(3.5)	**84.6**(1.0)	**94.7**(1.0)	**90.9**(0.7)	**55.0**(2.2)	**93.1**(2.3)	59.7(7.3)
# of dimensions								
M3GP 50	12.5(1-20)	6(3-13)	27(22-32)	11(6-18)	18(9-23)	10(6-20)	20(15-24)	11(8-32)
HR	16.5(7-21)	9(4-15)	32(25-39)	10(7-16)	16(10-22)	14(9-21)	22(18-28)	11(7-13)
HR-XO EP	15(10-21)	8.5(3-16)	31.5(26-36)	8(4-16)	16(11-20)	12(10-21)	22(17-25)	11(9-12)
HR-XO X&M	14(2-18)	7.5(3-12)	28(23-33)	8(5-13)	14(9-22)	13.5(8-20)	20(15-24)	11(9-12)
AM1σ	17.5(10-22)	8(4-14)	32(26-37)	10(6-17)	15(9-21)	13(9-22)	20(17-24)	11(9-12)
AM1σ-XO EP	14.5(4-23)	8(4-13)	30.5(24-36)	10(6-16)	14(11-22)	14(4-19)	19(17-23)	11(9-13)
AM1σ-XO X&M	14(3-22)	8(3-16)	28(22-33)	8(5-16)	13.5(10-20)	13(10-19)	19(17-23)	11(9-13)

Table 6.2: Comparison between M3GP, HR, HR-XO EP, HR-XO X&M, AM1σ, AM1σ-XO EP and AM1σ-XO X&M.

Table 6.2 shows a comparison between the previously referred approaches on training and test accuracy. Regarding training accuracy, all the approaches perform equally on all the datasets (the difference is not statistically significant) except for M3GP on WAV, and AM1σ-XO X&M and HR-XO X&M on HRT, WAV, SEG and IM-10. As for test accuracy AM1σ-XO EP and AM1σ-XO X&M are not competitive with the others on M-L dataset.

The approaches which produce the highest median training and test accuracy, on each dataset are marked in *italic*.

As expected, the median depth values for HR-XO X&M and AM1σ-XO X&M are lower than the ones for HR-XO EP and AM1σ-XO EP, and the depth values for all of the previous are lower than the depth values for HR and AM1σ (there is only one exception on M-L).

	HRT	**IM-3**	**WAV**	**SEG**	**IM-10**	**YST**	**VOW**	**M-L**
HR	497.5 (435-608)	539 (426-671)	392 (336-434)	545.5 (455-602)	477 (399-587)	517 (419-594)	410 (325-535)	560.5 (356-643)
HR-XO EP	396.5 (105-573)	412 (92-561)	322 (63-399)	347 (23-518)	336.5 (88-440)	372 (120-510)	308.5 (71-455)	441 (98-539)
HR-XO X&M	269.5 (21-440)	252 (84-406)	199.5 (14-315)	274.5 (21-371)	174 (39-366)	201 (42-394)	206.5 (35-378)	245 (42-406)
AM1σ	501.5 (420-580)	552 (318-650)	378 (308-457)	515 (421-609)	482.5 (391-573)	517 (246-580)	155 (57-371)	69 (6-629)
AM1σ-XO EP	436.5 (189-560)	403 (67-546)	301.5 (36-406)	332.5 (35-476)	321 (31-483)	388.5 (109-513)	182.5 (62-455)	98 (7-581)
AM1σ-XO X&M	249 (56-385)	255.5 (35-274)	210 (84-343)	238.5 (29-392)	217.5 (35-343)	231 (21-336)	209.5 (8-343)	52 (6-343)

Table 6.3: Comparison between HR, HR-XO EP, HR-XO X&M, AM1σ, AM1σ-XO EP and AM1σ-XO X&M depth values.

Table 6.4 shows that operators' percentages change according to the operators probabilities. Finally, table 6.5 shows a comparison between the average number of operations according to approach and dataset. For each dataset, the average number of operations does not vary much (except on VOW and M-L, since there is a clear difference between choosing the moving point in the hyperrectangle or around a misclassified point), but the minimum values tend to vary more.

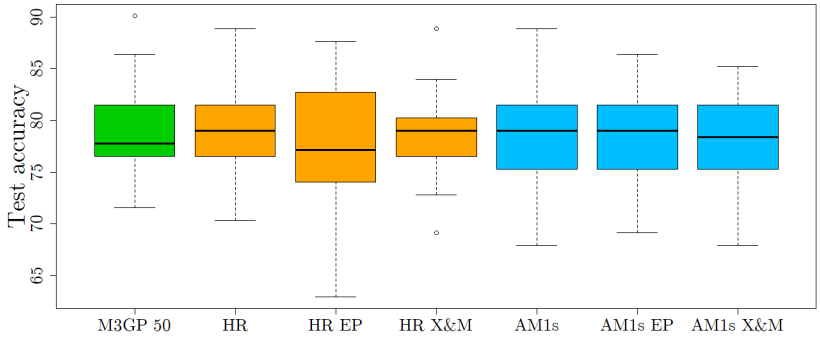
		HRT	IM-3	WAV	SEG	IM-10	YST	VOW	M-L
HR	*M*	73%	77%	57%	78%	72%	76%	60.3%	78%
	A	22%	16%	41%	17%	24%	20%	31.4%	16%
	R	5%	7%	2%	5%	4%	4%	8.3%	6%
HR-XO EP	*M*	63%	67%	51%	66%	62%	65.7%	55%	70%
	A	23%	14%	41%	16%	25%	20.9%	31%	15%
	R	6%	7%	3%	6%	5%	4.9%	8%	6%
	XO	8%	12%	5%	12%	8%	8.5%	6%	9%
HR-XO X&M	*M*	49%	49%	40%	49%	48%	49%	46%	49%
	A	22%	14%	42%	17%	25%	22%	30%	18%
	R	6%	6%	4%	6%	6%	5%	7%	8%
	XO	23%	31%	14%	28%	21%	24%	17%	25%
	M	72.9%	78.7%	57%	76%	73%	75%	47%	45%
AM1σ	*A*	22.5%	14.6%	40%	19%	23%	21%	49%	51%
	R	4.6%	6.7%	3%	5%	4%	4%	4%	4%
AM1σ-XO EP	*M*	66.8%	67%	51%	65%	60%	66%	49%	48%
	A	20.5%	15%	41%	19%	26%	20%	41%	40%
	R	5.2%	6%	3%	6%	6%	4%	4%	5%
	XO	7.5%	12%	5%	10%	8%	10%	6%	7%
AM1σ-XO X&M	*M*	48.59%	48%	41.0%	50%	45%	49%	42%	31.7%
	A	23.75%	16%	41.4%	17%	26%	23%	34%	39.5%
	R	6.68%	7%	3.3%	6%	6%	6%	6%	4.5%
	XO	20.98%	29%	14.3%	27%	23%	22%	18%	24.3%

Table 6.4: Comparison between HR, HR-XO EP, HR-XO X&, AM1σ, AM1σ-XO EP and AM1σ-XO X&M operators' percentages.

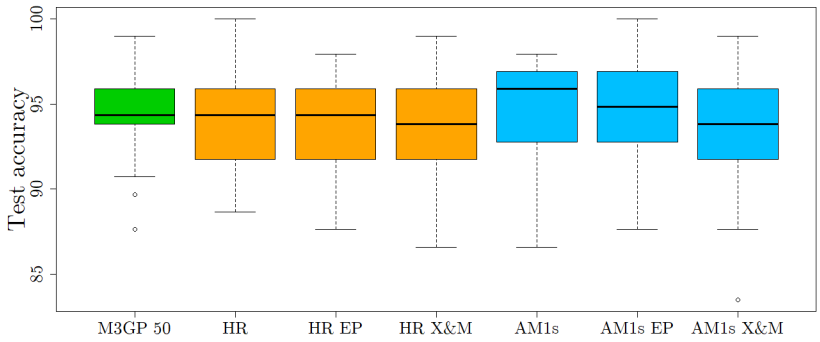
	HRT	IM-3	WAV	SEG	IM-10	YST	VOW	M-L
HR	98.4	99.2	97.2	97.8	94.7	97.2	99.3	99.6
	(95-100)	(96-100)	(93-100)	(93-100)	(90-99)	(93-100)	(97-100)	(96-100)
HR-XO EP	97.4	98.5	95.4	96.9	91.4	95.4	98.4	98.2
	(92-100)	(94-100)	(87-99)	(92-100)	(84-97)	(86-100)	(96-100)	(93-100)
HR-XO X&M	95.1	97.8	91.1	93.7	90	90.8	93.7	93.8
	(89-100)	(92-100)	(82-96)	(82-100)	(77-100)	(81-98)	(89-100)	(78-100)
AM1σ	98.5	97.7	97.6	97.7	92.8	96.1	48.4	46.9
	(94-100)	(62-100)	(93-100)	(92-100)	(84-99)	(89-100)	(30-76)	(11-100)
AM1σ-XO EP	98.2	96.7	95.2	95.0	90.6	95.5	61.1	58.5
	(93-100)	(39-100)	(89-99)	(88-100)	(74-98)	(85-99)	(27-100)	(12-100)
AM1σ-XO X&M	93.8	95.2	91.2	93.1	87.5	89.9	75.6	52.9
	(86-100)	(56-100)	(90-99)	(86-99)	(75-96)	(81-100)	(43-97)	(13-98)

Table 6.5: Comparing the number of operations between HR, HR-XO EP, HR-XO X&M, AM1σ, AM1σ-XO EP and AM1σ-XO X&M.

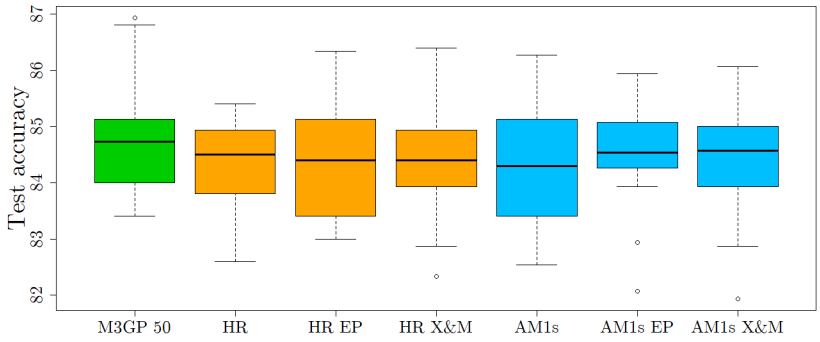
Figures 6.1 and 6.2 show a comparison between last generation test accuracies for the different approaches, by dataset.



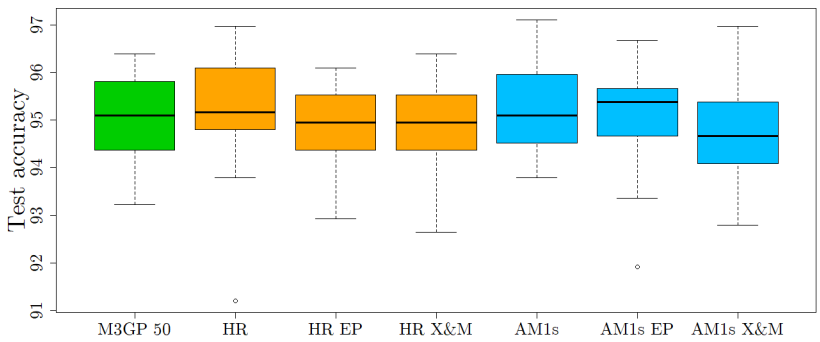
(a) HRT - Test.



(b) IM-3 - Test.

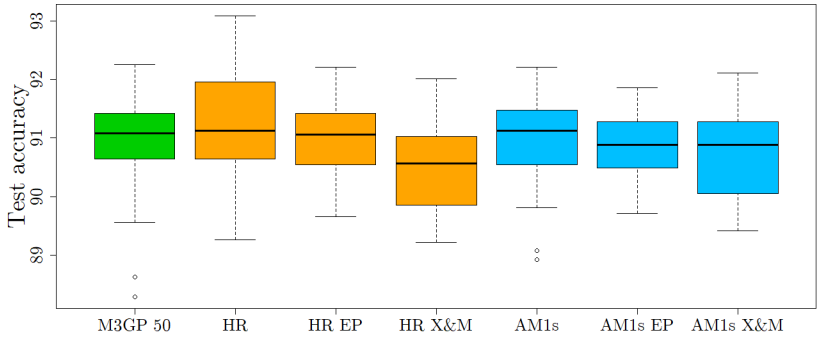


(c) WAV - Test.

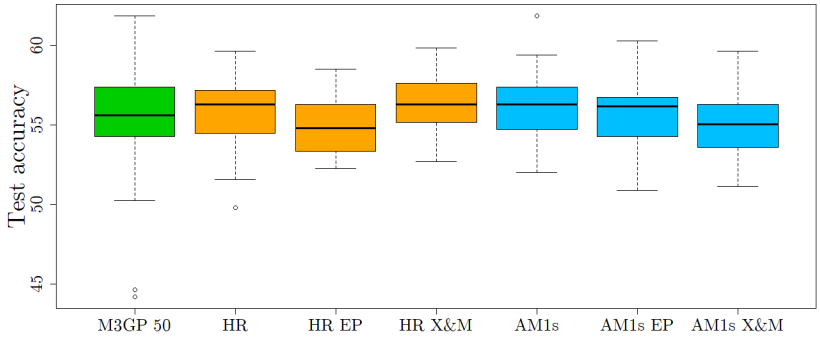


(d) SEG - Test.

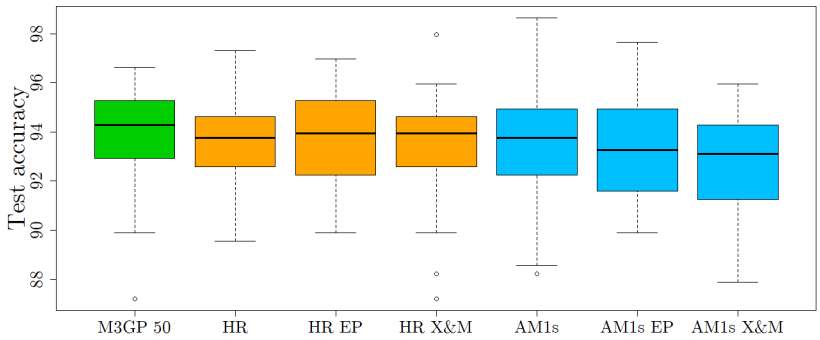
Figure 6.1: Comparison between last generation individuals' accuracies of M3GP 50, HR, HR-XO EP, HR-XO X&M, AM1σ, AM1σ-XO EP and AM1σ-XO X&M algorithms on HRT, IM-3, WAV and SEG datasets.



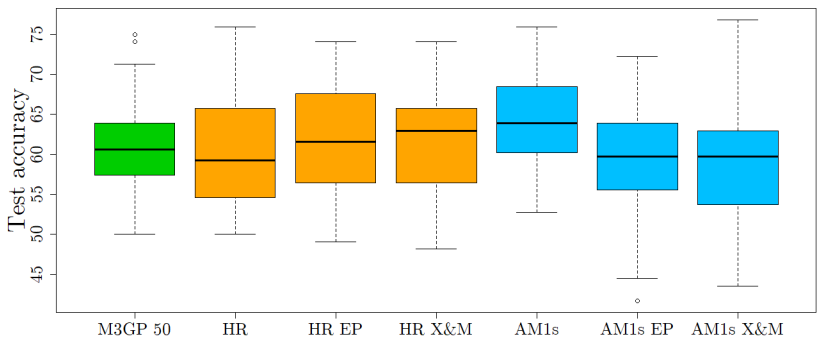
(a) IM-10 - Test.



(b) YST - Test.



(c) VOW - Test.



(d) M-L - Test.

Figure 6.2: Comparison between last generation individuals' accuracies of M3GP 50, HR, HR-XO EP, HR-XO X&M, AM1σ, AM1σ-XO EP and AM1σ-XO X&M algorithms on SEG, IM-10, YST and M-L datasets.

Summarizing, on the last generation (which might not be generation 100 for AM1σ versions):

- On HRT and SEG, HR and AM1σ are the approaches reaching the highest values;
- HR and AM1σ produce the highest test accuracy values on IM-3;
- As expected, on WAV and YST, M3GP is the best approach;
- On IM-10, the approach reaching the highest values of test accuracy is HR;
- AM1σ is the approach reaching the highest test accuracy values on VOW;
- Finally, on M-L, HR, AM1σ and AM1σ X&M are the approaches reaching the highest values of test accuracy on the last generation, but AM1σ X&M also reaches the lowest values between all the approaches.

One can conclude that by adding the *swap dimensions* crossover operator, there is no improvement in the accuracy of the individuals.

Chapter 7

Implementation issues of M3GP

The aim of this section is to refer some details of the implementation which are quite important, highlight some issues, and also answer some questions that the reader might have once trying to implement the algorithm.

7.1 Training set and test set

When shuffling and "randomly" dividing the dataset instances between the training set and test set, one needs to be careful so that at least 2 instances belonging to each class are in the training set - otherwise, the algorithm does not work. Each sample covariance matrix needs at least 2 instances to work, since the formula uses a division by $N - 1$, where N is the number of training instances assigned with that class. Also, every class needs to be represented in the training set.

This is a concern, for example, when considering YST dataset. It has 10 classes and 1484 instances. However, only 5 instances are assigned to class 10, corresponding to approximately 0.3% of the instances in the dataset.

7.2 Covariance matrices

In M3GP, an individual/solution is characterized not only by its tree, but also by its cluster centroids and covariance matrices (as many as the number of classes present in the dataset).

7.2.1 What if a covariance matrix is not invertible?

If a covariance matrix is *not invertible* (or if it is *singular*), then the Mahalanobis distance cannot be computed, since the inverse of the covariance matrix is required to compute it.

When this happens, one can choose between a set of options such as:

- use a different distance measure, which does not requires the covariance matrix to be invertible;
- replace the inverse matrix by a pseudo-inverse matrix, such as the Moore-Penrose pseudo-inverse matrix.

In this implementation, a different distance measure was considered: the Euclidean distance measure. Hence, if the covariance matrix for a given class is not invertible, the Euclidean distance between the centroid and the instances is computed instead of the Mahalanobis distance.

7.2.2 What matrices are considered to be invertible?

Mathematically, a matrix M is *invertible* if and only if $det(M) \neq 0$. But what if, in a computer program, there is a matrix A such that, for instance, $det(A) = 10^{-40}$? Is A invertible? Although $det(A)$ is not 0, one cannot trust A to be invertible. Considering matrix

$$B = \begin{pmatrix} -2.000000009904309 & 2.0 \\ 2.0 & -1.9999999900956875 \end{pmatrix}$$

According to JAVA, $det(B) = -7.105427357601002 \times 10^{-15}$, but we know that B is not invertible.

So how can one check if a matrix is invertible, when running a computer program?

Computationally, one should only consider a matrix M to be invertible if and only if $|det(M)| > \epsilon$ (where ϵ is a very small number) instead of $det(M) \neq 0$. But how small should ϵ be? The value of ϵ is an arbitrary choice of the person implementing the code.

However, different values of ϵ might give very different outputs. This is the case of the Baseline 1% version of the GSI-M3GP algorithm (presented and explained in chapter 5), used for the M-L dataset. The training and test accuracy values are significantly different when considering $\epsilon = 10^{-4}$ and $\epsilon = 10^{-10}$ (the difference between the two is almost 30% in training accuracy and 10% for unseen accuracy). This is due to the fact that the covariance matrices' determinants were always very small (smaller than 10^{-4}), and the Euclidean distance was used most of the times when considering $\epsilon = 10^{-4}$ - probably because $\epsilon = 10^{-4}$ is still too big to be considered as being equal to 0.

On top of this problem, another one was arising from using the determinant to decide whether a matrix is singular or not: some of the distances between points and centroids returned a result equal to NaN. The algorithm works even with NaN distances, but that is not correct. *But where do these NaN distances come from?*

Remembering the formula of the Mahalanobis distance:

$$d_M(x) = \sqrt{(x-\mu)^T C^{-1} (x-\mu)},$$

where μ is the centroid, C is the covariance matrix and x is a point in space to which the distance from it to the distribution described my μ and C is being calculated.

A covariance matrix C is always a *positive-semidefinite* matrix. However, the Mahalanobis distance can be defined if and only if C is a *positive-definite* matrix. A *positive-semidefinite* matrix is *positive-definite* if and only if C is invertible[7]. Then, if C is invertible:

$$(x - \mu)^T C^{-1}(x - \mu) \geq 0,$$

i.e. the squared Mahalanobis distance is positive, and so the mahalanobis distance can be computed.

In JAVA, if one tries to calculate the square root of a negative value, the output is NaN. The NaN distance values were appearing because of this problem - singular matrices were being "inverted" causing squared Mahalanobis distances to, sometimes, be negative.

Hence, I looked at other ways to check singularity of matrices. Another way to verify if a matrix is nonsingular is to compute its rank. Considering a $n \times n$ matrix M, M is nonsingular if and only if $rank(M) = n$. If $rank(M) < n$, then M is singular. Using the rank to check matrix singularity, NaN values appear only very sporadically. In order to completely eliminate this issue, when for a specific point x, we have $rank(C) = n$ but $(x - \mu)^T C^{-1}(x - \mu) < 0$, the euclidean distance is computed instead.

Notice that: my first implementation of the determinant's computation was done using *Gauss Elimination with Partial Pivoting* (not the Laplace's formula). After that, I used function *det*() from *JAMA* (A Java Matrix Package, see https://math.nist.gov/javanumerics/jama/). A lot of NaN values appeared using both implementations. The function used to compute matrices' rank is *rank*() function from *JAMA*.

7.3 Dimensions of a tree

A question that can be posed is *what to do when a dimension is constant?* A constant dimension is a dimension consisting only of one numeric constant node, or a function of numeric constant nodes. The constant dimensions do not add any "knowledge" to the tree, *i.e.* it is not helpful for clustering. One can easily accept this, my imagining the points in space.

Mathematically, what happens?

Let $w_k(x) = a \in \mathbb{R}$, *i.e.* a constant dimension. Then $cov(w_k, w_i) = 0$, $\forall i \in \{1, .., d\}$. Let C be the covariance matrix, which is $d \times d$, when considering the set of all dimension variables (including the constant dimension). Then, the k^{th} column and line of C are full of zeros. If a square matrix has a line or a column of zeros, then $rank(C) < d$, and so, C is singular. If C is singular, its inverse cannot be computed and so, the Mahalanobis distance cannot be considered. As such, when there is a constant dimension, the Mahalanobis distance is never used.

Let C' be the covariance matrix that results from discarding the line and column of zeros, *i.e.* by discarding the constant dimension. Then, C' might be nonsingular, unlike C. If that is the case, then C' can be inverted and the Mahalanobis distance can be calculated.

The possible difference in accuracy of the two individuals (I and I', where I is the original individual and I' is the individual for whom the tree does not have the constant dimension) comes only from the fact that different distance measures might be considered. Thus, one cannot say that, by taking out the constant dimensions, the accuracy of an individual always improves or always deteriorates. However, the accuracy should not change. However, by considering constant dimensions, that might happen.

By the way the pruning procedure is implemented, these constant dimensions only disappear after pruning if the training accuracy improves without that dimension. By applying pruning to the trees with constant dimensions, the individuals get a "truthful" training accuracy - but one could be transforming a spuriously very good individual into a not so good one in terms of accuracy. The pruning procedure is only applied to the best individual in each population and so, very good individuals could be considered worse because of these constant dimensions.

One concludes that it does not make sense to have a dimension consisting only of one terminal node that is a numeric constant (or a composition of numeric constants). However, when the Java code was implemented and ran, these conclusions had not been taken yet, and so, the results presented in this document consider trees with constant dimensions.

Chapter 8

Conclusions

8.1 Summary of Contributions

A new version of Multidimensional Multiclass Classification Genetic Programming with Multidimensional Populations (M3GP), called Geometric Semantic Inspired M3GP (GSI-M3GP), was presented in this document. As explained in the previous chapters, GSI-M3GP works as M3GP, but it only uses three operators instead of the five operators presented in M3GP. GSI-M3GP operators are: *add branch to the tree*, *remove existing branch from the tree* (introduced by M3GP's authors in [15]), and the new mutation operator: *geometric semantic inspired mutation*.

As the name of the operator implies, it is inspired in geometric semantic operators. And like the geometric semantic operators, it suffers from the drawback of creating *offspring* with number of nodes much larger than their parents. For GSI-M3GP, the number of nodes problem is even more critical since the trees created by GSI-M3GP are multidimensional. Hence, one of the *cons* of GSI-M3GP is the creation of large trees, that might not be able to be written by a computer.

GSI-M3GP shows to be competitive with M3GP on benchmark problems, achieving training and test accuracies which are not statistically different from the ones attained by M3GP. Thus, with only three operators, none of them being a crossover operator, GSI-M3GP is able to achieve the same results as the 5-operator M3GP.

8.2 Future Work

However, there are still aspects that can be improved. The pruning procedure has shown to increase the accuracy of the best individual in each population, yet it can still be improved. The way it is acting now, the pruning procedure does not take into consideration all the combinations of dimensions. If this could be done in a way that the algorithm would not slow down, then the pruning procedure would be optimal.

As referred in chapter 7, the implementation developed in the context of this thesis allows for numeric constant dimensions (*i.e.* dimensions consisting of one terminal or a function of numeric constants), and it should not allow for constant dimensions. This issue can easily be tackled, and it should be tackled in subsequent implementations of both M3GP and GSI-M3GP.

Finally, and most importantly, each time the geometric semantic inspired mutation is applied, at maximum only one instance gets differently classified (with some exceptions), while for GSGP's geometric semantic operators, the semantics of the individual can change in more than one entry (and so, for more than one instance). Hence, the geometric semantic inspired mutation does not have the same impact on datasets with very different number of samples - it will have a higher impact on low number of samples datasets. Future work considers the creation of a geometric semantic inspired mutation operator that might be able to re-classify multiple instances at the same time, or even a "true" geometric semantic mutation operator. The previously referred exceptions come from the fact that some mapped samples might change place if one of their entries is equal to an entry of the moving point, and correctly classified samples might get misclassified, which is also a *con* of the algorithm.

Bibliography

[1] U.S. geological survey (USGS) earth resources observation systems (EROS) data center (EDC). http://glovis.usgs.gov/.

[2] J. Alcalá-Fdez, A. Fernández, J. Luengo, J. Derrac, and S. García. Keel data-mining software tool: Data set repository, integration of algorithms and experimental analysis framework. *Multiple-Valued Logic and Soft Computing*, 17(2-3):255–287, 2011.

[3] K. Bache and M. Lichman. UCI Machine Learning Repository. http://archive.ics.uci.edu/ml, 2013.

[4] M. Castelli, S. Silva, L. Vanneschi, A. Cabral, M. J. P. de Vasconcelos, L. Catarino, and J. M. B. Carreiras. Land Cover/Land Use Multiclass Classification using GP with Geometric Semantic Operators. In *Applications of Evolutionary Computation - 16th European Conference, EvoApplications 2013, Vienna, Austria, April 3-5, 2013. Proceedings*, pages 334–343, 2013.

[5] C. Darwin. *On the Origin of Species by Means of Natural Selection.* Murray, London, 1859. or the Preservation of Favored Races in the Struggle for Life.

[6] I. Gonçalves, S. Silva, and C. M. Fonseca. On the Generalization Ability of Geometric Semantic Genetic Programming. In *EuroGP*, 2015.

[7] R. A. Horn and C. R. Johnson, editors. *Matrix Analysis.* Cambridge University Press, New York, NY, USA, 1986.

[8] V. Ingalalli, S. Silva, M. Castelli, and L. Vanneschi. A Multi-dimensional Genetic Programming Approach for Multi-class Classification Problems. In *Revised Selected Papers of the 17th European Conference on Genetic Programming - Volume 8599*, EuroGP 2014, pages 48–60, New York, NY, USA, 2014. Springer-Verlag New York, Inc.

[9] R. A. Johnson and D. W. Wichern, editors. *Applied Multivariate Statistical Analysis*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1988.

[10] I. Jolliffe. *Principal Component Analysis*. Springer-Verlag, New York, 2002.

[11] J. R. Koza. *Genetic Programming - on the programming of computers by means of natural selection*. Complex adaptive systems. MIT Press, 1992.

[12] S. Luke and L. Panait. Lexicographic Parsimony Pressure. In W. B. Langdon, E. Cantú-Paz, K. Mathias, R. Roy, D. Davis, R. Poli, K. Balakrishnan, V. Honavar, G. Rudolph, J. Wegener, L. Bull, M. A. Potter, A. C. Schultz, J. F. Miller, E. Burke, and N. Jonoska, editors, *GECCO 2002: Proceedings of the Genetic and Evolutionary Computation Conference*, pages 829–836, New York, 9-13 July 2002. Morgan Kaufmann Publishers.

[13] T. M. Mitchell. *Machine learning*. McGraw Hill series in computer science. McGraw-Hill, 1997.

[14] A. Moraglio, K. Krawiec, and C. G. Johnson. Geometric Semantic Genetic Programming. In *PPSN*, 2012.

[15] L. Munoz, S. Silva, and L. Trujillo. M3GP: Multiclass Classification with GP. In P. Machado, M. I. Heywood, J. McDermott, M. Castelli, P. Garcia-Sanchez, P. Burelli, S. Risi, and K. Sim, editors, *18th European Conference on Genetic Programming*, volume 9025 of *LNCS*, pages 78–91, Copenhagen, 8-10 Apr. 2015. Springer.

[16] R. Poli, W. B. Langdon, and N. F. McPhee. *A field guide to genetic programming*. Published via `http://lulu.com` and freely available at `http://www.gp-field-guide.org.uk`, 2008. (With contributions by J. R. Koza).

[17] S. Silva and J. Almeida. GPLAB - A Genetic Programming Toolbox for MATLAB. In *Proc. of the Nordic MATLAB Conference, NMC-2003*, pages 273–278, 2005.

[18] L. Vanneschi. An Introduction to Geometric Semantic Genetic Programming. In *NEO*, 2015.

[19] L. Vanneschi, M. Castelli, L. Manzoni, and S. Silva. A New Implementation of Geometric Semantic GP and Its Application to Problems in Pharmacokinetics. In *Proceedings of the 16th European Conference on Genetic Programming*, EuroGP'13, pages 205–216, Berlin, Heidelberg, 2013. Springer-Verlag.

[20] A. J. Wilson. Volume of n-dimensional ellipsoid. *Sciencia Acta Xaveriana (SAX)*, 1(1):101–106, 2010.