NOVA IMS Information Management School

MAA

Mestrado em Métodos Analíticos Avançados
Master Program in Advanced Analytics

Progressive Insular Cooperative Genetic Programming Algorithm for Multiclass Classification

Karina Brotto Rebuli

Dissertation presented as the partial requirement for obtaining a Master's degree in Data Science and Advanced Analytics

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

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

PROGRESSIVE INSULAR COOPERATIVE GENETIC PROGRAMMING ALGORITHM FOR MULTICLASS CLASSIFICATION

Karina Brotto Rebuli

Dissertation presented as the partial requirement for obtaining a Master's degree in Data Science and Advanced Analytics

Advisor: Professor Dr. Leonardo Vanneschi

November 2019

Progressive Insular Cooperative Genetic Programming Algorithm for Multiclass Classification

Acknowledgements

My heartfelt thanks go to:

All my professors of this master degree. Especially Professor Leonardo Vanneschi, my always supportive supervisor, a great professor who is excellent in teaching and who encourages students' ideas so they can truly develop. *Grazie mille!*

All my classmates. It was a pleasure to learn with you. It is amazing when, beyond fellow students/professionals, we can find beautiful human beings on such a journey!

The NOVA IMS staff, who are always helpful and friendly.

My professors from Brazil, namely Professor Paulo Afonso Bracarense Costa and Professor Paulo Ribeiro Justiniano Jr. They were of major importance in my scientific trajectory and I will always be grateful to them, wherever I am studying or working.

Professor DeRose and my DeROSE Method monitor, Valéria Vidal, with whom I learn day by day to find the strength in who I am. This helps me overcome every challenge with a smile on my face and makes my life an amazing experience, more and more conscious and full of possibilities.

My partner João, with whom I have the pleasure of sharing my life, the love and the certainty that we can be the better world with which we dream about. Thank you also for revising this thesis and for the love and affection in these last days of big effort! I would also like to thank João's mother, Maria Clara, who was also there for me in these last days with her care and attention to help me finish this work.

My brother Átila, who is always there for me and who supports me whenever and wherever I need. You are always part of my achievements!

Last but not least, my father and mother, who gave me life and who taught me to love. You are insuperable in many ways, but I hope that the *XO* that created this *offspring* has been able to *optimise* some other characteristics, so that you can always feel proud.

Abstract

In contrast to other types of optimisation algorithms, Genetic Programming (GP) simultaneously optimises a group of solutions for a given problem. This group is named population, the algorithm iterations are named generations and the optimisation is named evolution as a reference to the algorithm's inspiration in Darwin's theory on the evolution of species.

When a GP algorithm uses a one-vs-all class comparison for a multiclass classification (MCC) task, the classifiers for each target class (specialists) are evolved in a subpopulation and the final solution of the GP is a team composed of one specialist classifier of each class. In this scenario, an important question arises: should these subpopulations interact during the evolution process or should they evolve separately?

The current thesis presents the Progressively Insular Cooperative (PIC) GP, a MCC GP in which the level of interaction between specialists for different classes changes through the evolution process. In the first generations, the different specialists can interact more, but as the algorithm evolves, this level of interaction decreases. At a later point in the evolution process, controlled through algorithm parameterisation, these interactions can be eliminated. Thus, in the beginning of the algorithm there is more cooperation among specialists of different classes, favouring search space exploration. With elimination of cooperation, search space exploitation is favoured.

In this work, different parameters of the proposed algorithm were tested using the Iris dataset from the UCI Machine Learning Repository. The results showed that cooperation among specialists of different classes helps the improvement of classifiers specialised in classes that are more difficult to discriminate. Moreover, the independent evolution of specialist subpopulations further benefits the classifiers when they already achieved good performance. A combination of the two approaches seems to be beneficial when starting with subpopulations of differently performing classifiers.

The PIC GP also presented great performance for the more complex Thyroid and Yeast datasets of the same repository, achieving similar accuracy to the best values found in literature for other MCC models.

Key-words: Multiclass classification (MCC), Genetic Programming (GP), Team GP.

Resumo

Diferente de outros algoritmos de otimização computacional, o algoritmo de Programação Genética (PG) otimiza simultaneamente um grupo de soluções para um determinado problema. Este grupo de soluções é chamado população, as iterações do algoritmo são chamadas de gerações e a otimização é chamada de evolução em alusão `a inspiração do algoritmo na teoria da evolução das espécies de Darwin.

Quando o algoritmo GP utiliza a abordagem de comparação de classes um-vs-todos para uma classificação multiclasses (CMC), os classificadores específicos para cada classe (especialistas) são evoluídos em subpopulações e a solução final do PG é uma equipe composta por um especialista de cada classe. Neste cenário, surge uma importante questão: estas subpopulações devem interagir durante o processo evolutivo ou devem evoluir separadamente?

A presente tese apresenta o algoritmo Cooperação Progressivamente Insular (CPI) PG, um PG CMC em que o grau de interação entre especialistas em diferentes classes varia ao longo do processo evolutivo. Nas gerações iniciais, os especialistas de diferentes classes interagem mais. Com a evolução do algoritmo, estas interações diminuem e mais tarde, dependendo da parametrização do algoritmo, elas podem ser eliminadas. Assim, no início do processo evolutivo há mais cooperação entre os especialistas de diferentes classes, o que favorece uma exploração mais ampla do espaço de busca. Com a eliminação da cooperação, favorece-se uma exploração mais local e detalhada deste espaço.

Foram testados diferentes parâmetros do PG CPI utilizando o conjunto de dados iris do *UCI Machine Learning Repository*. Os resultados mostraram que a cooperação entre especialistas de diferentes classes ajudou na melhoria dos classificadores de classes mais difíceis de modelar. Além disso, que a evolução sem a interação entre as classes de diferentes especialidades beneficiou os classificadores quando eles já apresentam boa performance. Uma combinação destes dois modos pode ser benéfica quando o algoritmo começa com classificadores que apresentam qualidades diferentes.

O PG CPI também apresentou ótimos resultados para outros dois conjuntos de dados mais complexos, o thyroid e o yeast, do mesmo repositório, alcançando acurácia similar aos melhores valores encontrados na literatura para outros modelos de CMC.

Palavras-chave: Classificação multiclasse, Programação genética, Programação genética com equipes.

Table of contents

List of tables

List of figures

List of abbreviations

AB	AdaBoost
CIR	Cooperation Intensity Rate
DL	Deep Learning
EA	Evolutionary algorithms
ELM	Extreme learning machine
GDBT	Stochastic Gradient Boosting Decision Trees
GP	Genetic Programming
GSGP	Geometric Semantic Genetic Programing
IRS	Iris
KNN	K-Nearest Neighbours
LGP	Linear GP
LR	Logistic Regression
MCC	Multiclass classification
ML	Machine Learning
NB	Naive Bayes
NN	Neural Networks
PIC	Progressively Insular Cooperative
RF	Random Forests
RHH	Ramped Half-and-Half
ROC	Receiver Operating Characteristic
SCR	Sparse Representation based Classification
STGP	Strong Typed GP
SVM	Support Vector Machine
THY	Thyroid
YST	Yeast

1. Introduction

Evolutionary algorithms (EA), including genetic programming (GP), are a class of machine learning algorithms that optimises a group of solutions instead of working with a single solution at a time. This group of solutions is called population as reference to its biological inspiration in Darwin's theory of evolution of the species. Essentially, at each step of the learning phase of the algorithm, the individuals are modified to generate new individuals and the best are selected for the next generation. This selection step emulates the natural selection of Darwin's theory and introduces to EA the ecological relationship of competition. In supervised learning tasks, individuals compete to give the best algorithm solution for a regression or a classification problem. The former problem has a continuous outcome and the latter, a discrete outcome that is the predicted class of an observation.

Besides EA, there is a profusion of supervised algorithms to solve multiclass classification (MCC) problems, such as K-Nearest Neighbors (KNN), Naive Bayes (NB), Neural Networks (NN), among others. When dealing with classification of three or more target classes, a crucial question arises: how to compare the classes among them, all at once or in pairs? In practical terms, addressing this question means having either a single classifier to hold the entire classification task or to have as many classifiers as the number of classes to be modeled. In the latter, the final prediction will be a combination of all classifiers. In GP context, both approaches can be taken. If classifying all classes at once, a single solution will have to distinguish all target classes. If classifying in pairs, GP should be changed from its basic design to generate and evolve more than one solution, since it will need one for each target class. These one-class classifiers are called specialised individuals. They are grouped in teams, an up-level solution that combines specialised individuals of each class to give the algorithm prediction. Consequently, in this approach, the GP evolution works in an upgraded two-level design, one level for the specialised individuals and another for the teams.

In addition to the competition relationship, which is always present in GP algorithms, the two-level design of teams-based GP presents an opportunity for the introduction of cooperation between individuals of different specialisations. Cooperation is a mutually beneficial interaction between species (Boucher, 2016) that contrasts with intragroup competition, in which individuals work against each other. Even if not present in GP standard applications, these mutually beneficial interactions among species are ubiquitous in nature and have played a pivotal role for the evolution of life on Earth (Preussger et al., 2020). The cooperation is present in a team-based GP only if specialised individuals are

allowed to interact over the evolution process. The team operation by itself is not a cooperative, but a collaborative action, since the specialised individuals just work together but do not benefit from this.

This work presents the development of the Progressively Insular Cooperative (PIC) GP, a cooperative team-based GP algorithm for MCC in which different classifiers can evolve with different levels of interaction and specialised individuals compose the team to make the final algorithm prediction. The main reasoning behind this algorithm is to change the rate of cooperation among individuals of different specialisations during GP evolution in order to keep the balance between learning from specialists of other classes and from other specialists of the same class. Specialists start all in the same GP population but, as the GP evolution moves forward, the subpopulations of specialists can be progressively separated or completely detached, then working as islands. Thus, the specialised individuals start learning from individuals of any specialisation but become restricted to learn only from individuals of the same specialisation over the GP evolution.

This can help GP because when specialists of different classes interact, they are helping each other to explore the search space and when they become restricted to interact only with individuals of the same specialisation, the exploitation of the search space is being favoured. It is expected that in the beginning of the GP evolution, the search space exploration will be more beneficial and as the algorithm evolves, the exploitation will become more important. This is because in the beginning of the algorithm a more intense exploration will allow solutions to look more widely for good regions of the search space and, then, once these good regions are found, it is more advantageous to intensify the exploitation, *i.e.* a more detailed look in these good search space regions.

In a standard GP, some parameters of the algorithm can help to control the search space exploration and exploitation balance, like the initialisation method, the crossover and mutation rates, etc. In PIC GP, in addition to them, the level of interaction between individuals of different specialisations is also used to interfere in this balance. The control of the level of interaction between individuals of different specialisations is done by three parameters: the cooperation intensity rate (CIR), the rate of CIR decrease and the generation in which the algorithm should separate specialised subpopulations previously allowed to interact (demes) into islands. The selection method is changed from the standard algorithm to work with two parents at a time. The first is chosen to balance the number of individuals among the specialisation classes. The second is chosen according to the CIR parameter and it will control the level of interaction between class specialisations. The CIR is a parameter in

the $[0, 1] \in \Re$ interval that weights the individuals fitnesses according to their specialisation to change the probability of selection of the second parent. If the individuals are in the same specialisation of the first parent, their fitnesses are not changed, otherwise they are weighted by the CIR. That is, the bigger the CIR, the less the fitness of other specialisation classes will be decreased and it will allow more interactions among individuals of different specialisation classes. Thus, PIC GP offers a team-based GP in which it is possible to control the intensity of cooperation among different class specialised groups over the algorithm evolution.

As in any classification GP, the fitness measure of the specialists is important for differentiating the individuals, since it is this fitness that guides the specialists' evolution. In addition to some largely used classification assessment measures like the accuracy, the area under the receiver operating characteristic curve and the *f*-score, the present work introduces a new measure called fuzzy accuracy. To calculate accuracy, individuals' real number outcomes are discretised into 0 or 1 by a threshold, with 0 being attributed to one class and 1 to the other. Then, each correct prediction is given a value of 1 and the sum of correct answers is divided by the total number of predictions. In fuzzy accuracy, each correct prediction is given a value corresponding to the distance between the real number outcome and the threshold used to discretise the prediction. Thus, instead of summing up 1 for each correct prediction in the numerator as in accuracy, the distances between the real number outcomes of correct classifications and the classification cutoff are summed. Therefore, the farthest from the threshold correct outcomes are, the higher the fitness will be. In PIC GP this can be important because the team prediction is made based on the real number outcomes and thus more information can be used for the algorithm prediction.

The evolution of specialists alone is not enough to produce good teams, which are the entities responsible for the final algorithm classification. It is therefore important to evolve also the teams and in PIC GP, teams evolve in a completely separate process from the individuals' evolution. Teams are made by combining probabilistically the best specialists for each class. These teams then participate in crossover and mutation operations. In the team crossover operation, parent teams will exchange entire specialist individuals of the same specialisation class. In the team mutation operation, new genetic material can come from the specialists population or it can be a new random tree. Moreover, the individual to be replaced can be chosen randomly or probabilistically, favouring the replacement of the weaker specialists in the latter. The team prediction is based on the probability of its members to give a positive prediction. The class with highest probability is the one taken by the team. As an alternative, a weighted version of the team prediction is also presented. In this case, the real

number outcomes from the team's members are weighted by the respective individuals' fitnesses, giving higher importance to the outcome of more qualified individuals.

To assess the effects of the PIC GP parameters in the evolution of specialists and teams as well as in final GP accuracy, the current work presents experiments with different selection algorithms, different specialists' fitnesses, different CIR values, different CIR decrease rates, different generations in which the algorithm changes from demes to islands, different team prediction methods and different team mutation operators.

The next sections of this thesis are organised as follows: chapter 2 provides a review of GP, with its main topics and the state of art in MCC GP; chapter 3 presents the proposed algorithm, including a discussion of its main features; chapter 4 presents results obtained with the proposed algorithm for multiple datasets; and chapter 5 closes the work with the conclusions and recommended future work.

2. Theoretical background

2.1. Genetic programming

Genetic Programming is a very flexible evolutionary Machine Learning (ML) algorithm that can be used for regression or classification in a wide variety of problems. As any evolutionary algorithm (EA), GP corresponds to a metaheuristic optimisation that works on the concept of population, that is a set of candidate solutions that evolves through individuals structure variations and fitness-based selection. In fact, GP can work even as a hyper-heuristic optimisation procedure, that instead of operating directly on the problem search space, operates on the heuristic search space, searching for the heuristics to be used to solve the target problem (Poli et al., 2008). In analogy to biological systems, the solutions are individuals, their structures are their genotype and their fitnesses, their phenotype. The genotype reflects in phenotype, that determines the probability of the individual to survive and to generate new individuals. Simply put, the phenotype drives the genotype perpetuation, but not its modification, since the genotype transformations are made blindly. Moraglio et al. (2012) presented the Geometric Semantic GP (GSGP) in which the operators that produce modifications in the individuals' structure are not completely blind, but reflect modifications in the solution output (its semantic) and the search is based in the error space (the space of the distances between solutions' semantics and the target). This space is unimodal, which ensures that the algorithm will not be trapped in local optima, representing a possible new state-of-the-art machine learning methodology (Vanneschi & Poli, 2012). Other examples of variations of GP widely used are Linear GP (LGP), that evolves computer programs written as linear sequences of instructions (Brameier & Banzhaf, 2007) and Strong Typed GP (STGP), that enforces data type constraints when forming solutions (Montana, 1995).

2.1.1. Tree-based GP

Part of GP flexibility is due to the flexibility of its solutions and the present work refers to tree-based GP. Trees are hierarchical, variable-size structures that represent computer programs with nested nodes and terminals. Nodes are elements that combine other elements (one up to many) and terminals are final branch elements. The combination of nodes and terminals allows the solutions to have variable (adaptable) size and to be of different kinds, like computer programs, decision trees, mathematical expressions (called discriminant

functions in GP context) or combined objects. Figure 1 shows an example of a mathematical expression represented as a tree.

While evolving, GP changes randomly the elements of the trees and this results in changing also their shapes. Therefore, solutions with different shapes interact in GP evolution. Trees satisfy two necessary conditions to be able to interplay with other trees with different shapes: sufficiency and closure. Sufficiency states that the terminals and nodes (in combination) must be capable of representing a solution to the problem. Closure requires that each function of the node set should be able to handle all values it might receive as input (Espejo et al., 2010). For example, the arithmetic division is usually applied in a protected form to handle with zeros in its denominator. Without these properties the random generated trees could produce a non-valid solution.

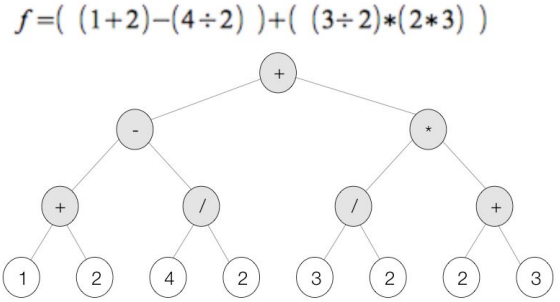

Figure 1: A mathematical expression represented as a tree. Gray objects are nodes and white objects are terminals.

2.1.2. Initial population

To build the first population, the maximum initial depth and the generation method of its individuals should be chosen. The depth is defined by the number of levels of nested nodes and terminals in a tree. The method defines if the tree will be full, having as many elements as possible for its maximum initial depth, or if it will have any size (the number of elements of a tree) and depth as long as it does not exceed the initial depth limit. The former method is called full and the latter, growth. In practice, a very common GP population initialisation is the ramped half-and-half (RHH) (Koza, 1992) , that creates half of the solutions with the full method and the other half with the growth method. This initialisation also has its limitations because it tends to produce a diversity bias, favouring full trees (Burke et al., 2003). Moreover, the creation of this initial random population is, in effect, a blind random search of the search space of the problem (Koza, 1994), and many alternatives are proposed in literature. For example, for GSGP (Vanneschi et al., 2017) proposed an initialisation algorithm in which the initial population individuals are the best individuals of different

populations (initialised with RHH method) that had already evolved for some generations, thus increasing the variability of individuals in the first population.

2.1.3. Evolution

The basic GP design performs two steps in each evolution iteration: the selection of its individuals and their modification to forge the next generation population. Both steps are always performed in a probabilistic way. Thus, the algorithm evolves creating new solutions from previous ones and favouring the selection of the better ones to the next generations until the end of its evolution. The random nature of EA is a key factor differentiating this family of algorithms from others. In contrast to other ML algorithms, like neural networks for example, the randomness of EA algorithms is present not only in the algorithm initialisation, but over all the learning phase. The evolution can take different directions in each run and thus reach different solutions. Besides that, the best individuals have higher probability of being kept for the next generation even if bad individuals can also survive. However, due to the algorithm's stochastic nature, it can also happen that it loses the best solution of a generation. That is why it's so common to run GP with elitism, a deterministic operator that keeps the best individual for the next generation.

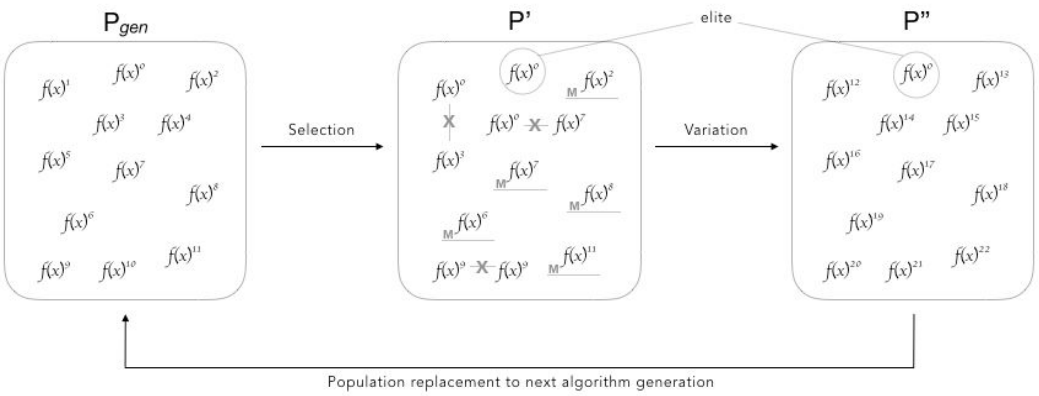

Figure 2: GP evolution cycle. Individuals in the population of a given generation (P_{gen}) are probabilistically selected according to some evolution criteria to create a temporary population P'. Next, the selected individuals are changed with crossover (X) or mutation (M) to generate new individuals in P''. The elite individual (circled) is kept unchanged between P' and P''. Later, the population P_{gen} is replaced by P'' when it goes to the next generation. Source: the author.

The random modifications of the solutions are made with a conservative and/or with an innovative variation operator: the crossover and/or mutation, respectively. The former exchanges the genetic material (the structure) between two solutions while the latter produces a random change in the genetic material of one single solution. Figure 2 shows a diagram of the GP evolution. In this example, the only preserved solution is the elite, which is

reproduced deterministically. Other individuals can also be reproduced depending on algorithm settings. The GP implementation can avoid the use of temporary populations to reduce its computational cost but the basic idea remains the same. Algorithm 1 shows a GP basic implementation.

Algorithm 1: *Genetic Programming.*

```
1: Set problem, terminal set and nodes set.
2: Set population size N, initialisation method, selection and elitism
   methods, crossover and mutation probabilities and termination
   condition.
3: Create N individuals for the initial population P.
4: repeat:
5:     Set individuals fitnesses.
6:     Starts P'
7:     repeat:
8:         Select parent 1.
9:         if making crossover:
10:            Select partner 2 and apply the crossover operator.
11:            Add offspring to P'.
12:        else if making mutation:
13:            Apply mutation operator to parent 1
14:            Add offspring to P'.
15:        else:
16:            Add parent 1 to P'.
17:    until P' has N offspring individuals.
18:    if using elitism: Apply elitism operator to offspring.
19:    Replace P with P'.
20:until maximum number of generations is reached or the best solution
   is acceptable.
21:return the best-so-far individual.
```

2.1.4. Selection algorithm

Selection algorithms can control specific characteristics of the evolution process, like the solutions size, the population diversity, etc. The most commonly used selection algorithms are:

- Fitness proportional selection: The probability of one individual to be selected is directly proportional to its fitness in maximisation problems or to the inverse of its fitness in minimisation problems. This is also called roulette wheel selection because it can be seen as a roulette wheel where each individual occupies a space proportional to the quality of its fitness. A random event chooses a point in the wheel, selecting the individual whose space contains that point. If the fitness values are too different, it tends to select only the best individuals, reducing the

chance of weaker individuals to be selected. This can lead the algorithm to a premature convergence, disabling it to properly explore the search space.

- Ranking selection: Instead of using the fitness value, the individuals are ranked according to the quality of their fitnesses. The ranking is used to define the probability of the individuals to be chosen in a fitness proportional selection.
- Tournament selection: A group of n individuals (tournament) is randomly selected from the population, with or without replacement, and the best is chosen amongst them. Since the competition is no longer among all individuals but among a few chosen by chance, this selection reduces significantly the selection pressure, helping the algorithm to avoid premature convergence. Evidently, the bigger the tournament, the higher the selection pressure.

Countless other selection algorithms have been proposed. One maybe important to mention is the lexicographic parsimony pressure (Luke & Panait, 2006) that aims to prevent bloating. Bloat happens when there is a significant growth of trees during the evolution, leading to an unproportional increase of the solutions size compared with their fitness improvement (Vanneschi & Poli, 2012). This is an important drawback of GP because it slows down the algorithm, making it practically unworkable. The lexicographic parsimony pressure selector modifies the selection to prefer smaller trees when fitnesses are equal (or equal in rank).

2.1.5. Crossover operator

The basic implementation of the conservative variation operator is called one-point crossover. It uses two parent individuals coming from two independent selection steps to generate two new individuals. Each parent is broken in a random point, that is the same for both parents, and the broken up subtrees (the branches below this point) are exchanged. Figure 3 shows an example of the crossover operator that produces two offsprings. There are many variations of this basic crossover operator in literature. For example, the most commonly used crossover generates only one offspring with the root of the first parent and the subtree of the second parent, and discards the second offspring (Poli et al., 2008). Other variations try to improve evolution, especially aiming to control the individuals' growth. Examples include the size-fair crossover, which chooses the second parent subtree to guarantee that it is not too big compared to the first parent subtree, and the homologous crossover, which works like the size-fair but chooses deterministically the most similar subtree in the second parent (Langdon, 2000).

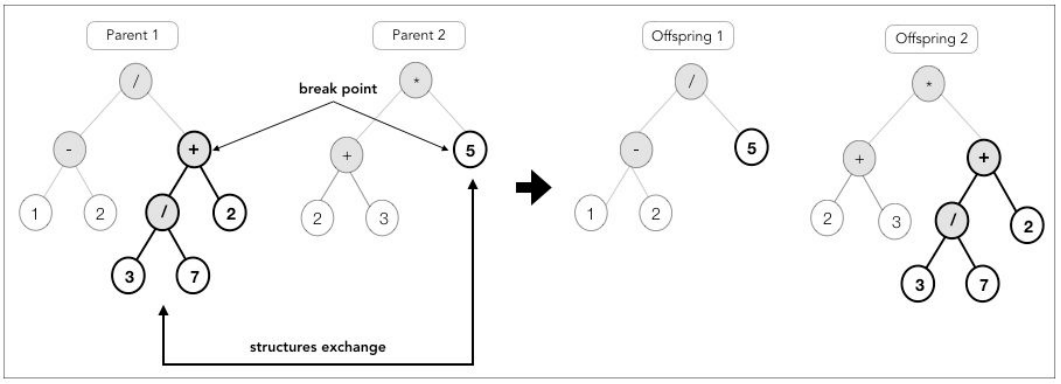

Figure 3: Example of a tree based GP one-point crossover with two offsprings. Source: the author.

2.1.6. Mutation operator

The basic GP innovative variation operator is called subtree mutation. It changes the tree in a random point by introducing a new random subtree. Figure 4 shows an example of the subtree mutation operator. There are many variations to prevent trees from growing or changing their structure too much. The importance of preventing uncontrolled tree growth has been explained above. Preventing large changes in tree structure is also important in order to prevent loss of knowledge gained during the evolutive process. Some of the mutation operators proposed in literature are (i) the point mutation that only exchanges a tree element by another with the same arity, (ii) the hoist mutation, in which the offspring is the parent subtree defined by a random mutation point (Kinnear, 1993) and (iii) the shrink mutation, in which a random parent subtree is replaced with a randomly selected terminal (Angeline, 1996).

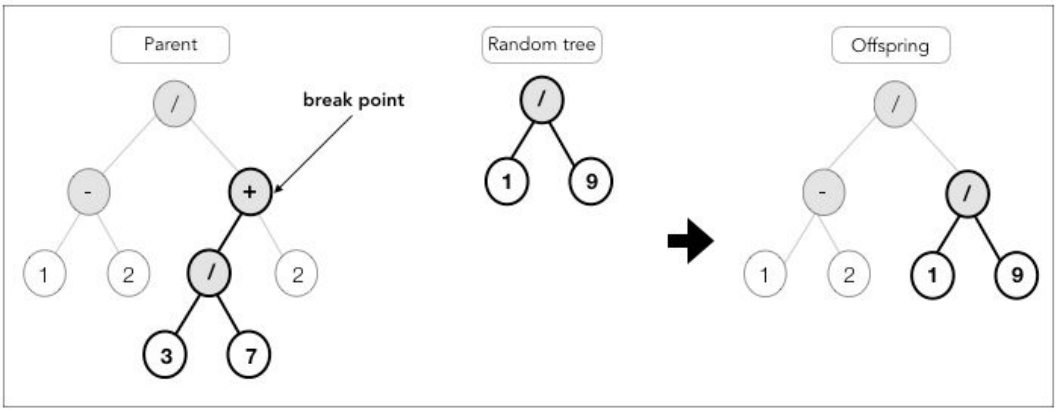

Figure 4: Example of a GP tree mutation. Source: the author.

2.2. GP for classification problems

For classification problems, the input data $\chi \in \Re$ should be mapped by a discriminant function $g(\chi) : \Re^d \rightarrow \Re$, such that each observation can be mapped to a class k, based on the evidence given by χ. Since $g(\chi)$ has its image in real numbers, its output needs to be converted to a categorical value. For binary classification, *i.e.* with two target classes, a threshold can be defined to separate the image values of $g(\chi)$ that correspond to one class from those that correspond to the other. A common application is to transform the GP output with the logistic sigmoid function (Eq. 1) and to use the threshold value 0.5. In this case, the tree solution is made up by the logistic function as the root node and the evolved tree attached to it.

$$S(x) = \frac{1}{1 + e^{(-x)}} \tag{1}$$

Apart from tuning GP parameters, some authors propose modifications in the basic GP design to handle binary classifications. Eggermont et al., 1999 presented a study with GP for binary classifications using stepwise adaptation weights and atomic features representation. The former increases progressively the weights of observations misclassified by the best solution to evaluate the fitness in the subsequent generation. The latter transforms all features into binary values, making tree nodes with more simple functions and increasing the interpretability of the model.

The classical GP returns just one discriminant function. Therefore, if the objective is multiclass classification (three or more target classes), the classical GP design won't work and a modification is required. There are two possible strategies to adapt the classical GP design to MCC GP, the wrapper or the direct approaches. They are summarised in Figure 5 and described below.

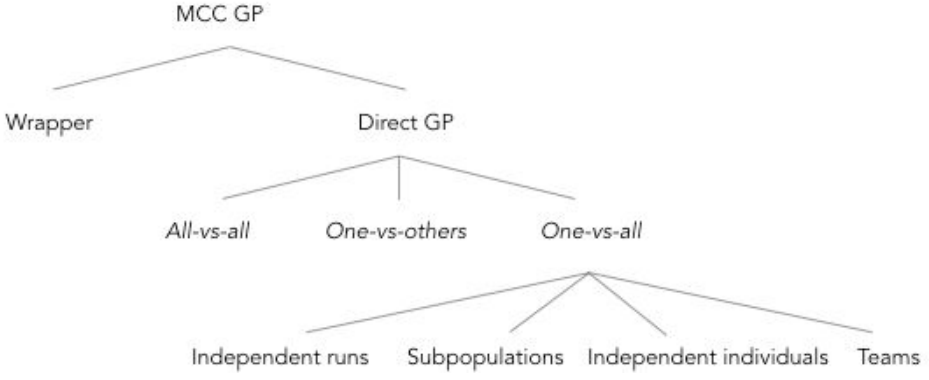

Figure 5: Strategies for multiclass classifications with GP. Source: the author.

2.2.1. Wrapper strategy

The wrapper strategy consists in applying GP to features transformation or features selection to enhance posterior multiclass classification by other algorithms. For example, Muñoz et al. (2015) used GP to project the dataset instances into a transformed space where the data of each class can be grouped into unique clusters. Then, their Mahalanobis distances to the clusters' centroids were evaluated and each instance was assigned to the closest cluster. In Raymer et al. (1996) the GP solutions transformed the data features and these transformed features were used as input for a KNN classifier. In Tan et al. (2005) a similar approach was taken, but using a Bayesian classifier. Al-Madi & Ludwig (2013) presented a wrapper MCC GP method in which a K-Means algorithm was fed with the GP solutions outputs.

2.2.2. Direct GP strategy

The direct GP MCC strategy consists in evolving a GP capable of providing a solution that gives the classification prediction directly, without any posterior classifier procedure. More attention is given to Direct GP strategies in this thesis because it was the one applied in the present work. The concept behind Direct GP is equivalent to the one used in GP for binary classification, but extended for many target classes. When comparing three or more classes, the very first decision is how to compare them. Three possibilities exist, which are explained in more detail below: all-vs-all, all-vs-others or one-vs-all.

2.2.2.1. All-vs-all comparison

This is the most simple extension of the binary classification approach. In this strategy, a single GP solution is generated and *K-1* thresholds are applied to its outcomes for a *K* classes problem. A single model must therefore be able to discriminate among all classes. Zhang & Smart (2004) give an example of a single classifier with *K-1* thresholds dynamically evolved during the GP run. Usually, this approach is less likely to produce good models, since it will have to handle all the problem complexity at once.

2.2.2.2. One-vs-others comparison

In this comparison, the problem of classifying *K* classes is decomposed into $K * (K - 1)$ binary problems, contrasting each class with others in pairs, to generate *K* classifiers, one for each target class, and combining their predictions in a final algorithm result. Kishore et al. (2000) presented a MCC GP in which the dataset was split through *K* classes instances. To evolve each of the *K* classification models, a dataset containing interspersed classes splits of

the target class with each of the other classes was built to compose the training set. With these training sets, the authors evolved a GP run for each class solution and evaluated a measure called strength of association, that assesses how much the solution was associated with its class. This measure was used to decide which solution prediction was chosen as the final algorithm prediction when more than one class model gave a positive prediction. The algorithm fitness was the classification accuracy. Silva & Tseng (2008) also used $K * (K - 1)$ binary problems with pairwise comparisons but the classifiers were assessed together for the GP fitness, evaluating the percentage of points assigned to more than one class. The goal of the GP was to minimise this value, abdicating a pos-processing to combine the classifiers' predictions.

2.2.2.3. One-vs-all comparison

In this comparison, the problem of classifying K classes is decomposed into K binary problems, contrasting each class with all others once, to generate K classifiers, one for each target class. The predictions of these K classifiers are then combined in a final algorithm result. In GP context, these K classifiers can be evolved in four different ways:

(i) In independent runs, simply running the algorithm K times, one for each class, with the dataset split for the corresponding one-vs-all comparison.

(ii) In the same run, but in different subpopulations. The subpopulations can be totally separated or they can interchange their individuals. If individuals cannot interchange between subpopulations, the subpopulations are called islands. Otherwise, the subpopulations are called demes (Wilson, 1977).

(iii) All together in the same population, but as independent individuals. The individuals evolve as in a standard GP implementation, but at each generation they are evaluated and set to be responsible for classifying one of the target classes.

(iv) All together as a team. It corresponds to evolving all classifiers in the same population together, dependently. The team is an individual in which the root node combines the results of its members. Each team member is a single threshold classifier that is specialised in a corresponding class. Both the team and its members evolve in the GP process. Thus, the two-level nature of evolving K classifiers that are combined to be a single GP solution becomes explicit in this approach. Evolving only the specialists can produce strong

individuals that perform poorly for the combined prediction. Nevertheless, the specialists should also evolve individually to be able to improve the team's output. For that to happen, it is necessary to define their individual evolution criteria, *i.e.* their individual fitness. Therefore, the team's approach creates a new decision requirement that is to define how the team fitness will be shared and distributed among the team's members. This is called the credit assignment problem (Brameier & Banzhaf, 2007). The team outcome will be the class whose specialist member gives a positive result. It could happen that more than one team member gives a positive result for the same data instance. Then, the team will require a disambiguation procedure, to define which of the positive classes will be its final classification result.

There are many studies in literature using these approaches. For example, Lin et al. (2007) used the (i) independent runs approach, proposing a multi layer with independent multi populations GP for MCC problems. In each run, the first layer used the training set configured for one specific target class. Before the last layer the proposed GP had many solutions, one for each population. In the last layer, the solutions obtained in the previous layer were combined in a single population, and a single GP solution was produced. The final prediction decision was given by a measure called z-value (Chien et al., 2004). This measure is obtained by splitting the training set to produce statistics for the classifiers and these statistics are used in a disambiguation step, if necessary, to decide the final classification. Chen & Lu (2007) used the (ii) island subpopulation approach in which the specialised solutions of a MCC GP evolved using the convex hull of the Receiver Operating Characteristic (ROC) curve as their fitness measure. Then, for each observation the classifiers made their predictions and the final GP prediction was decided by majority voting among classes models. Smart & Zhang (2005) used the (iii) all together with independent individuals approach for evolving all classifiers in a single GP run with solutions evaluated for every target class at each generation. The solution that provided the best separation for a binary class problem was assigned to be the classifier of that class. For the GP prediction, the data instance was evaluated by all K solutions and was assigned to the class to which it has the highest probability of belonging to.

Haynes et al. (1995) published pioneering work using the (iv) team's approach with STGP. Their focus was in the role of the crossover operator in making team populations evolve in coordination. The presented crossover operator essentially controlled if individuals specialised in classifying a target class could exchange genetic material with individuals

specialised in other target classes. In a later and more complete publication, Haynes & Sen (1997) proposed five crossover operators: (1) the team-branch, in which the exchange can occur between any specialist of one team at any point and any specialist of another team at any point; (2) the team-all, in which every specialist of a team exchanges genetic material at two independent random points with the correspondent specialist of another team; (3) the team-all-random, in which every specialist of a team exchanges genetic material with random specialists of another team (not necessarily of the same specialisation); (4) the team-uniform, that randomly sets pairs of specialists, one from each parent team but from any specialisation class, to participate in the crossover. Then, these paired specialists exchange genetic material at random points; and (v) the team-k-cross, in which a defined number (k) of crossover points are defined in each team, independently of how many there will be in each specialist. For the problem the author studied, the team-uniform was the best crossover operator, since it sped up the evolution and increased the team fitness (that is, the GP fitness its own).

There are also mixed approaches. In Brameier & Banzhaf (2007), the authors applied the team approach together with the demes subpopulation approach for two binary classifications and a regression problem with LGP. Lichodzijewski & Heywood (2008) presented a mixed independent individuals and team approach, in a GP that evolves the training subset (called point population), the individual binary classifiers and the team, each in a separate evolution process. The training set populations had the objective of selecting useful training sets for the classification task. The classifiers population had the objective of evolving good binary predictors. Finally, the team had the objective of evolving good multiclass predictors. Soule & Komireddy (2007) also presented a mixed independent individuals and team approach in which specialist individuals evolved in islands. At each GP generation, for each specialisation class two individuals were selected for crossover and mutation. The offsprings replaced two low fitness teams. Thomason & Soule (2007) presented a variation of this approach in which teams are selected and replace individuals in islands too.

The next section presents the new design developed in this work for a MCC GP with a mixed approach for independent individuals and teams evolution.

3. Progressively insular cooperative GP

In the real world there is no definitive best strategy between having strong individuals that can perform extremely well on their own or having just good individuals that together can do a great job. It depends mostly on the task. However, in a probabilistic reasoning, as used in MCC GP context, the best of both worlds can be explored: strong individuals that can interplay well will produce a more robust outcome. When the individual has to work on its own, it does its job well and when the team cooperates, it improves the individuals' good decisions. It is like having a "dream team" to produce the best possible result.

The present work proposes the Progressively Insular Cooperative (PIC) GP, a one-vs-all mixed individuals and teams approach for cooperative MCC GP. Subpopulations of specialist individuals begin as demes but further in the algorithm evolution they can become islands. The main idea is to create a flexible cooperative GP in which specialists can be strong individually but also good in cooperating. For this purpose, the rate of interaction between specialists of different classes can be changed over the algorithm evolution. It can vary from unrestricted cooperation to no cooperation at all. Specialists evolve independently from the teams' evolution. The teams evolve with improved individuals, giving priority to the stronger ones but also giving the chance to weaker individuals to participate in the team.

It is important to evolve the team because it is the team that makes the final multiclass classification. However, this brings two difficulties to GP: (i) the credit assignment, already mentioned in section 2.2.2.3 One-vs-all comparison, and (ii) the fact that the specialist individuals do not evolve enough when the evolution is guided by the teams' performance (Soule & Komireddy, 2007). The credit assign problem is hard to solve because when individuals interact they create synergy, *i.e.*, the effect of their combined work can be bigger than the sum of the effect of the individuals separately. The second difficulty, the limitation of specialists' development in a team-based evolution, is related to the fact that if individuals' evolution is associated with the team's evolution, the search space exploration by the individuals can be slowed down. For example, it can happen that some change in the individual's structure would produce an improvement in its own performance, but a decrease in the team's performance. In this case, the individual will not be allowed to change, which means that it is not allowed to explore the search space properly.

The balance between exploration and exploitation of the search space is decisive in GP performance. Depending on the problem and on the algorithm settings, it can be more advantageous to promote one or other. Exploration means to look more widely, broadly, to

farther sections of the search space. Exploitation means to look more closely, in more detail, to a pre-explored search space section. To have too much exploration means to do a random walk in the search space and to have too much exploitation means to be trapped in a small portion of that. It is not guaranteed that specialists that evolve without being guided by the teams' evolution will properly explore and exploit the search space. Nonetheless, if their evolution is independent from the teams, it will be easier to control this balance. That is why it is important to allow specialist individuals to evolve by themselves. As explained above, the specialist individuals evolution in a team-based GP can be done through islands, totally separated subpopulations, or demes, overlapping subpopulations. Working with specialisation islands can restrain the search space exploration because solutions tend to become all similar through the GP evolution process, depending on the algorithm and problem configurations (Leung et al., 1997). It is not guaranteed that having all individuals in the same population, an extreme full demes situation, permitting them to exchange genetic material with individuals from other specialisations indiscriminately, will work. On one hand, it can lead solutions to explore novel and worthy portions of the search space. In a tree-based GP, *e.g.*, one specialist solution can share a part of its tree that is crucial for discriminating its class and thus to help another specialist tree in separating its own class instances from those of the class of the specialist that had shared the code. On the other hand, one specialist solution can share just irrelevant genetic material, making new solutions explore novel but worthless areas of the search space that will not contribute to their improvement.

In a traditional GP, solutions have all the same specialisation and the balance between exploration and exploitation is carried mainly by crossover and mutation rates and the selection pressure, which will define which solutions will go over mutation and which will go over crossover to form the next generation population. In a cooperative GP, the interaction among specialists can also help to control this balance. In a LGP study, Luke & Spector (1996) found that restricting the interaction of the individuals with individuals of the same specialisation improved the algorithm performance. Soule (2000) made experiments in a GP regression problem and concluded that heterogeneity among teams is necessary but not sufficient, while individuals' high specialisation, that is related with heterogeneity, is key for improving the algorithm performance. Nevertheless, it is not yet clear in literature how to benefit from the balance between cooperation and the evolution of highly specialised individuals to properly explore and exploit the search space.

In the present study, variations in the level of specialists interaction over time during algorithm evolution were explored. Individuals begin the algorithm distributed in class-based demes that detach over the evolution process up until working fully as islands. The idea is to enable the search space exploration more intensively in the beginning of the evolution and, as the individuals become more prepared for their specialised task, to intensify the search space exploitation.

Figure 6 shows a diagram of the PIC GP evolution. Part A shows in detail the specialists evolution. In the initial phase, specialists are in demes, that work like overlapping subpopulations. Two individuals specialised in different classes can undergo a crossover operation and generate offspring for any class. The level of interaction among demes decreases at each generation. Later the algorithm starts working in the islands phase, when individuals can make crossover only with individuals of the same specialisation class. Part B of this figure shows the teams' evolution. Initially, teams are composed of the best individuals from the specialists' population. Then, at the start of each generation, new teams are created with evolved specialists that are included in teams' population to participate in the selection process. This means the teams' evolution receives an input of new genetic material at each generation. If this new genetic material didn't come from the specialists' population, this insertion could be prejudicial to the GP evolution, making the algorithm vary too much and to lose a significant portion of the learned information. However, the new genetic material was already improved by the specialists' evolution. After selection, the teams' evolution follows the standard GP steps, crossover and/or mutation, to create the next generation population.
The next sections describe in detail how each step of PIC GP works and its entire algorithm is described in Algorithm 4.

3.1. Specialists evolution components

3.1.1. Solutions structure

The specialists are trees with a specialisation class. To classify instances, the tree has a logistic function (Eq. 1) at its root node and uses the trheshold 0.5 for classes discrimination.

The definition of the class in which individuals are specialised can be done in two ways. It can be automatic, simply set by the class for which the individual works better, *i.e.*, has a higher fitness. Or it can be assigned by the algorithm to balance the number of specialists in the population.

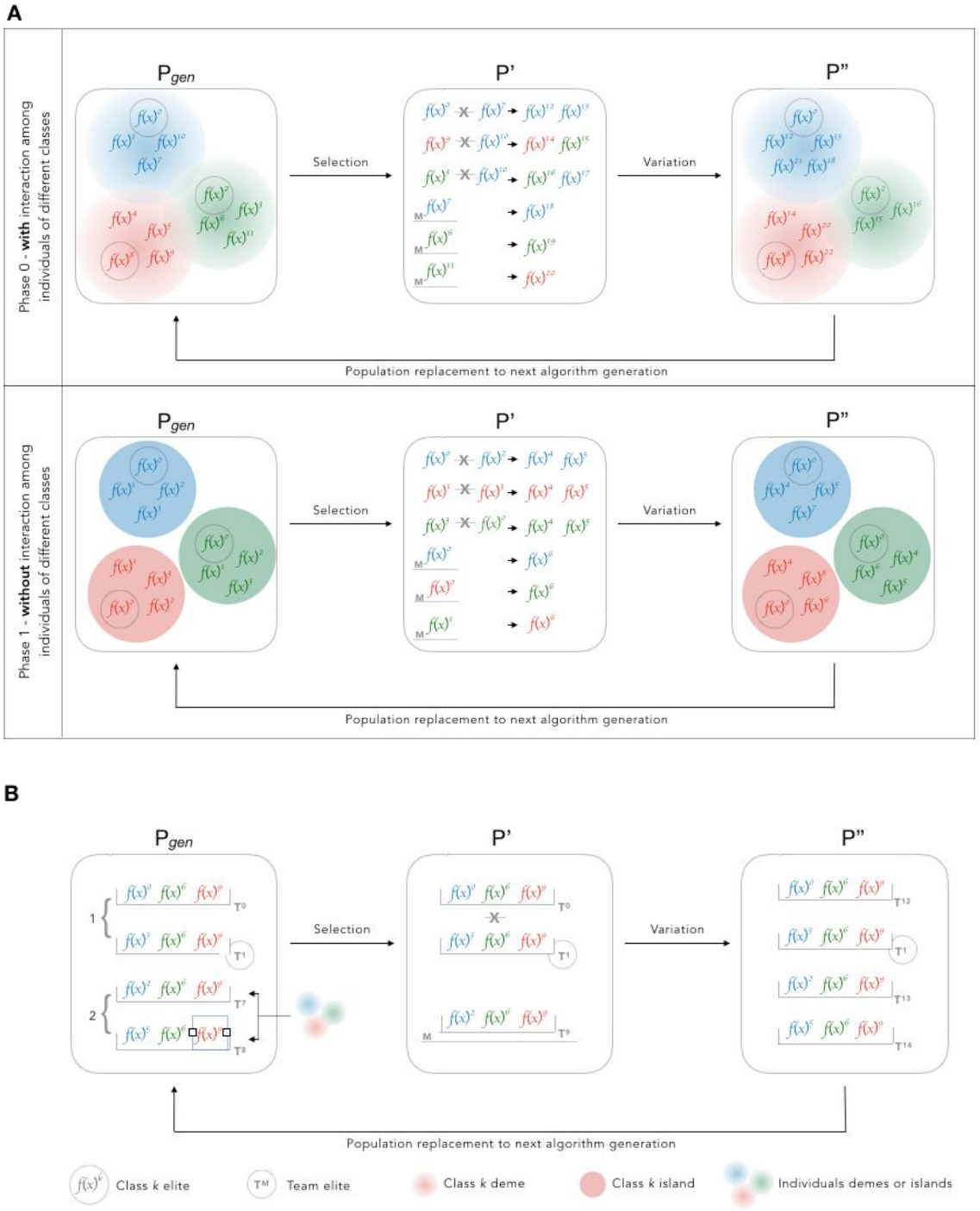

Figure 6: Progressively Insular cooperative GP. A: the specialists' evolution process. It begins with demes that can be transformed into islands over the algorithm evolution. B: the parallel teams evolution that at each generation receives new teams from the specialists population (2) and proceeds with a traditional GP selection and variation steps. Source: the author.

3.1.2. Solutions fitness measures

As mentioned in Chapter 2, the fitness is responsible for guiding the evolution direction. Since in PIC GP the specialised solution's goal is to make predictions for one single class, its

fitness is a measure of the quality of the classification of the class in which the solution is specialised. It can be one of the four following options:

- *Accuracy*: it is the percentage of corrected classified observations for a given class. It is calculated as follows:

$$acc_k = \frac{\#TP_k + \#TN_k}{\#TP_k + \#FP_k + \#TN_k + \#FN_k} \quad (2)$$

, where $\#TP_K$ is the number of true positive classifications for class k, $\#TN_K$ is the number of true negative classification for class k, $\#FP_K$ is the number of false positive classifications for class k and $\#FN_K$ is the number of false negative classifications for class k.

- *Fuzzy accuracy*: it is a measure of the strength/certainty of the predictions that are obtained. This is made by the application of a fuzzy concept on the accuracy evaluation taking into account that the last step for obtaining predictions is the conversion of a continuous outcome, the logistic value of the solution, into a binary value. Generally, the farthest the logistic value is from the threshold in the direction of the correct prediction, the more trustworthy the prediction should be. The opposite is also true, because if the outcomes are all close to the threshold, small variations in the input data will more easily make the resulting logistic value cross the classes threshold. Surely, this relation will depend on the statistical distribution of the features and on the tree structure. It is calculated as follows:

$$\Delta l_{Tk} = \frac{\sum 2*(l_{Tk}-0.5) + \sum 2*(0.5-l_{\overline{Tk}})}{N} \quad (3)$$

, where l_{Tk} are the logistic outcomes of true positive classifications, $l_{\overline{Tk}}$ are the logistic outcomes of true negative classifications and N is the total number of instances.

- *Receiver Operating Characteristic (ROC)*: The ROC curve is a tradeoff between sensitivity and specificity. For a single threshold point on the curve it becomes a simple arithmetic mean between the true positive and the true negative rates and it can be evaluated as (Chien et al., 2004):

$$ROC_k = 0.5 \times \left(\frac{\#TP_k}{\#P_k} + \frac{\#TN_k}{\#N_k} \right) \quad (4)$$

, where $\#P_K$ is the number of positive instances of the class k and $\#N_K$ is the number of negative instances of the class k.

- *F-score*: it is the harmonic mean of precision and recall rates of a target class:

$$f-score_k = \frac{2 * precision_k * recall_k}{precision_k + recall_k}, \text{ where:} \quad (5)$$

$$precision_k = \frac{\#TP_k}{\#TP_k + \#FP_k} \quad (6)$$

$$\text{and } recall_k = \frac{\#TP_k}{\#TP_k + \#FN_k} \quad (7)$$

The precision assesses the proportion of positive results that truly are positive and the recall assesses the proportion of correctly classified positives.

3.1.3. Specialists initial population

To ensure that there will be specialists of all classes in the initial population, after generating initial trees with the RHH method, the individuals are equally relocated over the classes. For each class, only the N/K best individuals are kept, where N is the size of the entire specialists population and K is the number of target classes. If there are more than N/K individuals specialised in a class, the remaining are randomly changed to other specialisation classes in which the number of individuals are less than N/K. If N/K is not a natural number, the next natural number is used.

Individuals that are relocated tend to have worse fitnesses and individuals that are not relocated will start from a better point. It is expected that the GP evolution will improve them all. Thus, the naturally best individuals for classifying a class are used for this and the others have to learn to make the classification that they are designed for. Algorithm 2 shows the PIC initialisation method.

Algorithm 2: *Specialists population initialisation.*

```
1: Define the population size N as a multiple of K.
2: Generate N individuals with rhh method.
3: for each k in target classes:
4:     Keep the N/K best individuals and store the others.
5: Shuffle stored individuals.
6: for each m in subpopulation classes with less than N/K individuals:
7:     while m has less than N/K individuals:
8:         Include a stored individual.
9:         Re-calculate the fitness of the included individual for the
   new class specialisation.
```

3.1.4. Specialists selection algorithm

In specialists' population, if the algorithm is in the demes phase, the selection algorithm works with two individuals at a time. The first individual is selected with roulette wheel or tournament selectors. To keep the balance of specialists in the population, this individual is chosen from a specific deme or island. The second parent is, then, selected with roulette wheel or tournament over the entire population with the fitnesses weighted by the cooperation intensity rate, the parameter that controls the quantity of interaction between individuals from different specialisations. The algorithm 3 shows the PIC specialists' selection methods for the demes algorithm phase.

Algorithm 3: *Specialists' selection method for demes algorithm phase.*

```
1: Define k, the class of the first parent according to the class that
   has less individuals in the new generation population.
2: if selection method is roulette wheel:
3:     for each individual in population:
4:         if the individual hasn't the specialisation class ≠ k:
5:             Recalculate its fitness:
6:             f' = f * η
7:     Select the second parent from the entire specialists population.
8: else:
9:     Select tournament_size individuals from the entire population.
10:    for each individual in tournament:
11:        Recalculate its fitness:
12:        f' = f * η
13:    Select the individual with higher f' for the second parent.
14:return both parents.
```

Note that the selection does not determine the class of the second parent. Moreover, it is not guaranteed that the offspring individuals will belong to the same specialisation class as the parents. Consequently, in the end of a generation, the proportion of individuals in each specialisation may change. Despite this, to control the class of the first parent, it is enough to keep the number of individuals in specialisation groups approximately balanced.

3.1.5. Cooperation intensity rate

This parameter is used to lower the fitness of individuals from other specialisations when specialists are competing in the second parent selection step, according to:

$$f'_i = f_i \times \eta, \text{ if } k_i \neq k_1 \qquad (8)$$
$$f'_i = f_i, \text{ otherwise.}$$

, where f_i is the *i-th* individual's fitness, η is the cooperation intensity rate, k_i is the *i-th* individual's specialisation class and k_1 is the specialisation class of the first selected parent.

The cooperation intensity rate (CIR) can be decreased over the evolution process by the decrease rate, another parameter of the algorithm. The decrease rate reduces constantly at each generation the rate of interaction among specialists over the GP evolution. This decrease does not convert the algorithm to the island approach when using the tournament selection for the second parent, even if the CIR goes down to zero. If the CIR is zero, the fitness of the specialists from other classes than the class of the first parent will all be zero. Despite that, it can happen that the tournament is composed of these individuals of other specialisation classes because the tournament is made randomly, without considering the individuals' fitness. Therefore, the subpopulations will still be demes, since they will still interact.

When using tournament selection for the second parent, to transform the algorithm from demes to islands phase, a phase change parameter is needed. It is the generation in which the algorithm should change the approach from demes to islands subpopulations.

Thus, with these two parameters, the CIR decay and the phase change, the demes could begin the evolution with overlapping areas that would be reduced over the generations up to a moment in which they have no more overlapping areas and are transformed into islands.

3.1.6. Crossover and mutation specialists operators

For crossover and mutation, the PIC GP uses the one point crossover with two offsprings and the one-point mutation operators.

3.2. Teams evolution components

3.2.1. Teams structure

A team is a tree with a prediction function at its root node, with arity of *K*, being *K* the number of target classes in the dataset, and with one specialist of each class in each of its branches. The specialists' structures are not changed on teams' evolution. It can be seen as if the teams' building blocks were not other trees themselves, but black-box classifiers. Considering that individuals can already exchange genetic material among them in a parallel evolution process that is dedicated to their improvement, it is assumed that there is no any major advantage in making individuals evolve into the teams' population.

3.2.2. Teams prediction

When working with teams for MCC, if only one of its members gives the positive prediction, it is the specialisation class of this member that will be the GP prediction. However when there is more than one positive result among team members, some technique is necessary to decide which prediction to choose.

In the present work, the team prediction is given by one of the following options:

- *Softmax*: it uses the softmax (Eq 9) result of the specialists' logistic outcomes.

$$\sigma(z)_j = \frac{e^{zj}}{\Sigma e^{zk}}, \; j = 1, \ldots, K \tag{9}$$

  The softmax function normalises the input values into a probability distribution consisting of probabilities that are proportional to the exponentials of the input values. Therefore, the team's decision is to choose the classifier that gives the higher probability of a positive outcome.

- *Specialist weighted*: it considers that not all the team members have the same quality in their predictions. So, before using the specialists logistic outcomes in the softmax function, the individuals logistic outcomes are weighted by their respective fitness, which is a measure of their individual prediction abilities. It is calculated as follows:

$$l' = f \times l \tag{10}$$

  , where f is the fitness of the specialist and l is its logistic outcome for a single data observation.

3.2.3. Teams initial population

The teams' population starts with one special team, deterministically created with the best specialist of each class from the specialists' population. The other teams are created with specialists selected with a roulette wheel selection from the specialisation subpopulations.

3.2.4. Teams elitism

If the best team has the same fitness as the previous best team, the best team with the best fitness for the test partition is kept in the population deterministically. Otherwise, if the best

team of a generation has better fitness then the best-so-far team, the new team is kept in the teams' population.

3.2.5. Teams crossover operator

The crossover operator of teams exclusively allows them to exchange entire branches of the same class. In other words, they can only exchange specialised individuals for individuals of the same specialisation class. Figure 7 shows an example of the teams crossover operator.

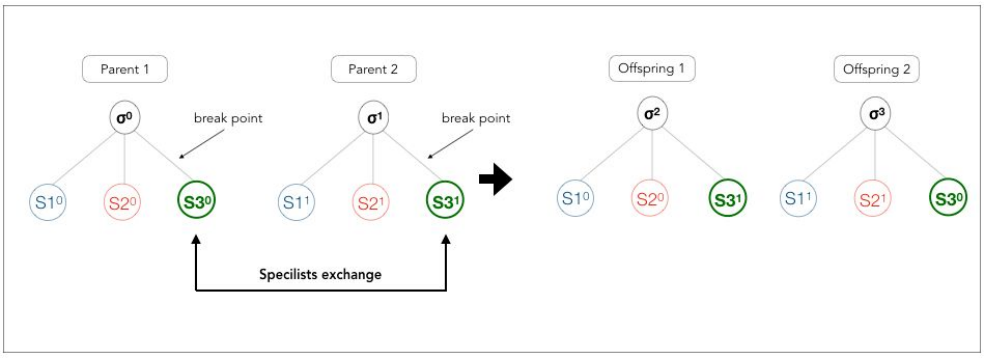

Figure 7: Teams crossover. Teams exchange entire specialist trees from the same class. Source: the author.

3.2.6. Teams mutation operator

The mutation operator is responsible for the innovative changes in GP individuals, which is important to allow the algorithm to explore the search space. Tree teams mutations are implemented in PIC GP.

- *Random* teams mutation: this is the most innovative teams mutation operator. A random specialist is substituted by a new random tree (Figure 8A).
- *Specialist* teams mutation: a random specialist is substituted by an individual with the same specialisation from the specialists' population. This is less innovative than the random teams mutation, but it still can provide more innovation than the introduction of new teams in the selection step because it does not prioritise the best individuals (Figure 8B).
- *Weaker* specialist team mutation: it works like the specialist teams mutation, but instead of removing a random specialist from the team, it chooses the specialist to be exchanged with a probability inversely proportional to the individuals' fitnesses (Figure 8C).

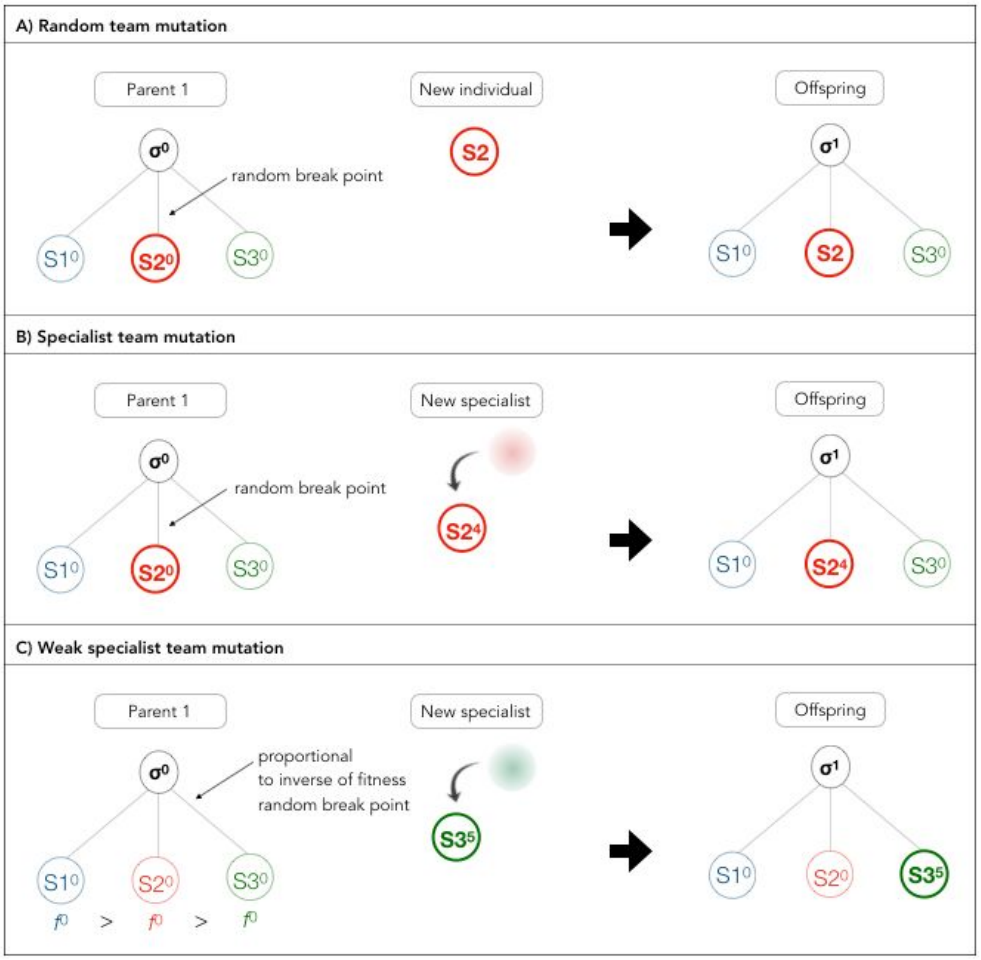

Figure 8: Teams mutation operators. Random: a random specialist is exchanged by a new random tree. Specialist: a random specialist is exchanged by a random individual with same specialisation in from specialists' population. Weak specialist: works like the specialist teams mutation, but the mutation point is chosen with inversely proportional probability to the team individuals' fitnesses. Source: the author.

Algorithm 4: *PIC GP Algorithm.*

```
Set:
    generations G, target classes K
    specialists population size N, teams population size M
    specialists crossover and mutation probabilities
    teams crossover and mutation probabilities
    specialists elitism, teams elitism
    specialists selection methods, teams selection method.
 Initialise specialists sp_population.
 Initialise teams tm_population.
for each g in G:
    Instantiate sp_population'.
    until sp_population' size is smaller than N:
       Select two specialists with specialists selection.
       if makes crossover:
          Apply specialists crossover to parent specialists
          Set the specialisation class and fitness of offsprings.
       if makes mutation:
          Apply specialists mutation to parent specialists
          Set the specialisation class and fitness of offsprings
       Add offspring 1 to sp_population'.
       if sp_population' size is smaller than N:
          Add offspring 2 to sp_population'.
    if apply specialists elitism:
       Apply elitism to sp_population'
    Replace sp_population with sp_population'.
    for i from 0 to i < M/2:
       Instantiate a team t.
       for each class in K:
           Select a class K specialist from sp_population.
           Put the selected specialist in t.
       Add t to tm_population.
    Instantiate tm_population'.
    until tm_population' size is smaller than M:
       Select a team from tm_population.
       if apply crossover:
          Select another team from tm_population.
          Apply crossover to generate two teams offspring.
       if apply mutation:
          Apply mutation to generate two teams offspring.
       Add offspring 1 to tm_population'.
       if exists offspring 2:
          if tm_population' size is smaller than N:
             Add offspring 2 to tm_population'.
    if apply teams elitism:
       Apply elitism to tm_population'
    Replace tm_population with tm_population'.
return the best-so-far team.
```

4. Results and discussion

Three multiclass datasets from the UCI Machine Learning Repository (archive.ics.uci.edu) were analysed, namely the Iris (IRS), the Thyroid (THY) and the Yeast (YST) datasets. The IRS was used to explore the proposed algorithm and its parameters. The others were used to assess the algorithm results in comparison to other classification algorithms from literature.

4.1. Experimental design

4.1.1. Dataset partition

All experiments were run 30 times, each with different data partitioning and algorithm seed. Thus, the variability observed in the results comes from the randomness in the data and in the algorithm. As shown in Figure 9, the data was first split into 5 k-folds crossvalidation (80% for train, 10% for test and 10% validation partitions). Then, the instances were shuffled and the k-fold crossvalidation partitioning was repeated 6 times.

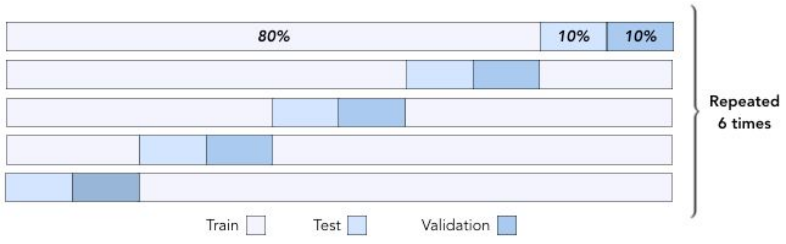

Figure 9: Data partition for the 30 runs used in experiments.

4.1.2. Run settings

In all experiments the trees were initialised with the RHH method with a tree maximum depth of 3. Specialists elitism and teams elitism were always used. The node set of the tree's structure was composed by $+$, $-$, $\times$ and protected $\div$. The terminal set was composed by ephemeral constants from $]0, 1[\in \Re$ interval in addition to the dataset features. The other default settings are presented in Table 1. The modified settings for each experiment are presented in the corresponding section.

4.1.3. Measurements

The measurements taken in all experiments are described in Table 2. Only results that are relevant for the discussion are presented in this Chapter, but the complete results are provided in Appendix A as supplementary material.

PIC GP Setting	IRS	THY	YST
Dataset normalisation	No	No	Yes
Trees maximum depth	6	10	10
Trees fitness measure	*f*-score	*f*-score	*f*-score
Specialists population size	90	90	120
Parent 1 selection	tournament size 3	tournament size 3	tournament size 3
Parent 2 selection	tournament size 3	tournament size 2	roulette wheel
Crossover probability	0.8	0.8	0.8
Mutation probability	0.2	0.2	0.2
Maximum generations	250	250	300
Phase change	200	200	240
Specialists interactions initial rate	1.00	1.00	1.00
Specialists interactions rate decrease	0.00	0.00	0.00
Team fitness method	Accuracy	Accuracy	Accuracy
Teams evolution	No	No	No

Table 1: PIC GP base settings used in experiments.

Measurement	Description
Accuracy mean (±sd)	Accuracy mean ± one standard deviation.
Team fitness	The train fitness of the best team in a generation.
Team fitness validation	The validation fitness of the best team in a generation.
Best team fitness	The train fitness of the best-so-far team.
Best team validation fitness	The validation fitness of the best-so-far team.
Specialists interactions	The number of times that two individuals of different specialisations participate in a crossover operation.
Trees mean size	The mean of the tree sizes of the individuals of a population.
Phenotype diversity	The variance of the fitnesses of the individuals of a population.
Genotype diversity	The number of different tree structures in a population divided by its number of individuals.
Number of specialists of each class	The number of individuals in each specialised subpopulation.
Mean fitness for each class	The mean of the training fitness of the individuals in each specialised subpopulation.
Phenotype diversity for each class	The phenotype diversity for each specialised subpopulation.
Genotype diversity for each class	The genotype diversity for each specialised subpopulation.

Table 2. Experiment measurements.

4.1.4. Dataset balance

To train individuals for each class, converting the all-vs-all method into the one-vs-all method, the dataset was divided in two parts, the positive cases (with the instances that belong to a target class) and the negative cases (with the instances that do not belong to that target class).

One problem that can arise from this procedure is that it may produce training data with an important imbalance between the positive and negative classes. In this scenario, the class with fewer observations is more likely to be misclassified than the class with more observations (Chawla et al., 2002). To balance back the instances, one can either replicate the less frequent class instances, or subsample the more frequent ones. The first option modifies the distribution of the target class data, since with replicated data the dataset variability will decrease. The second may cause the loss of information from the dataset. There is no ideal solution. The PIC GP presented in this thesis uses the second approach and it can be run with or without balancing. The following options were implemented in the algorithm:

- *Full*: All data instances are used as in raw data.
- *Balanced*: The class instances that are more frequent are randomly sampled to have the same number as of less frequent class instances. The sampling is repeated in each GP generation, thus providing different instances from the more frequent class to the algorithm training each time. With this strategy, no class prevails by quantity of instances and the classifier will lose less information over the algorithm evolution. This procedure does not balance originally unbalanced data, it only reverses the unbalancing caused by the split of the data into only two classes. If a class has too many or too few instances in the dataset, the only thing this procedure will do is to use as much information as possible to train the model with balanced data. Even if this means training the model for this specific class with less information than the models for other classes.

Figure 10 shows the balance between positive and negative cases for the three datasets analysed. The IRS dataset is the least unbalanced because it has only three target classes and they are perfectly balanced, with exactly the same number of instances for each class. The THY also has only three classes, but they are already unbalanced in the original data (*92.6*% of the data for hypothyroidism, *5.1*% for hyperthyroidism and *2.3*% for no disease). So, when split for a one-vs-all training, it becomes very unbalanced for all target classes (92.6% positive for hypothyroidism vs 7.4% negative; 5.1% positive for hyperthyroidism vs 94.9% negative;

2.3% with no disease vs 97.7% with disease). The YST dataset has seven target classes and also becomes highly unbalanced for each target class.

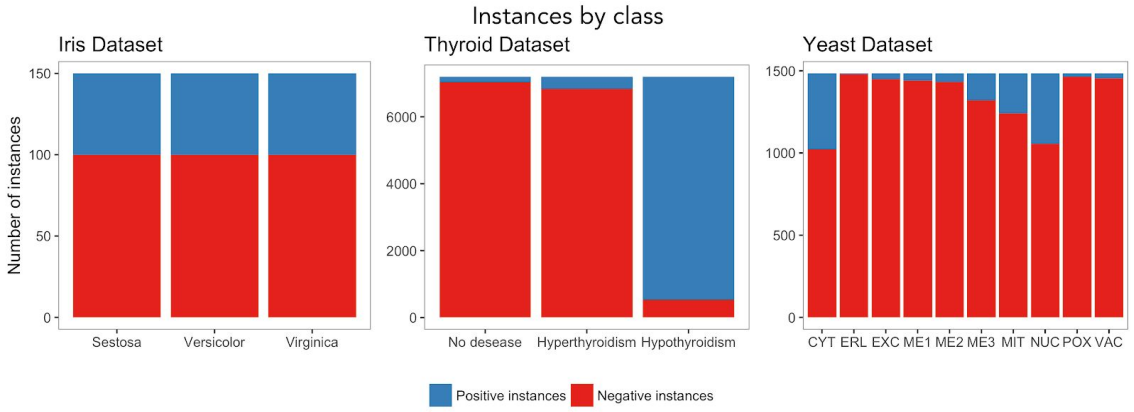

Figure 10: Classes balances for each of the tested datasets.

Table 3 presents the results for experiments conducted with full (not balanced) and with balanced methods for the three analysed datasets using the default GP settings.

		Mean ± (sd)		Best	
		Train	Validation	Train	Validation
IRS	full	**0.974** ± (0.015)	**0.967** ± (0.042)	**0.992**	**1.000**
	balanced	0.972 ± (0.020)	0.964 ± (0.049)	**0.992**	**1.000**
THY	full	0.967 ± (0.013)	0.967 ± (0.014)	**0.986**	**0.992**
	balanced	**0.969** ± (0.010)	**0.968** ± (0.013)	**0.986**	0.990
YST	full	**0.536** ± (0.027)	0.380 ± (0.172)	**0.570**	**0.642**
	balanced	0.530 ± (0.032)	**0.406** ± (0.130)	0.567	0.608

Table 3: Accuracy in train and validation partitions for dataset balance experiments.

The proposed sampling balanced method did not improve the results for any of the analysed datasets (t-test *p-values* 0.671 for training and 0.805 for validation in IRS; 0.421 for training and 0.621 for validation in THY; 0.326 for training and 0.469 for validation in YST).

4.2. Iris dataset

This is a very known dataset that has 4 real number features for flower measures (length and width of the petals and of the sepals) and 3 target classes, the flower species. As seen in Figure 11, the target class *setosa* is linearly separable from the other two species based on petal length and width. For sepal length, the *setosa* is more different from *virginica* than from *versicolor*, while the latter two are similar between them. For sepal width, *versicolor* and *virginica* are

almost matching and *setosa* is more different but still difficult to separate. Therefore, the easiest class to classify is *setosa*.

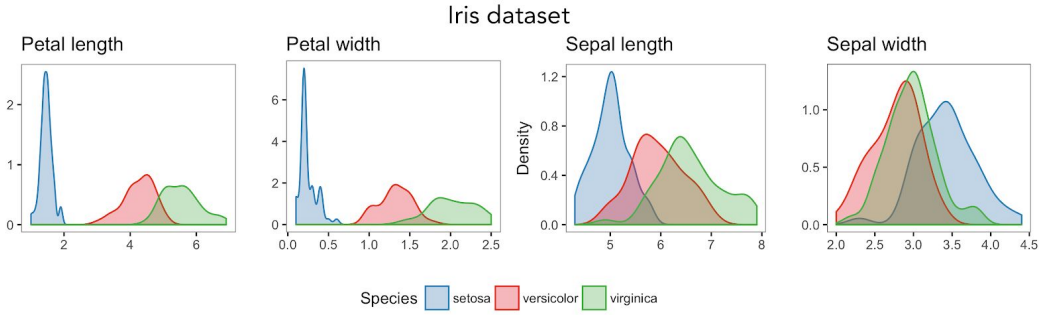

Figure 11: Iris dataset features distribution for each target class.

4.2.1. Specialists selection algorithm

The following selector methods were used in the specialists selection experiments:

Method	First parent	Second parent
T3_R	Tournament size 3	Roulette wheel
R_T3	Roulette wheel	Tournament size 3
T3_T3	Tournament size 3	Tournament size 3
T5_R	Tournament size 5	Roulette wheel
R_T5	Roulette wheel	Tournament size 5
T5_T5	Tournament size 5	Tournament size 5

Table 4: Specialists selection algorithms used in the experiments.

For all selection methods, the best accuracy for the validation partition was 1.00 (Table 5). The accuracy was also 1.00 for the training partition for T5_R and T5_T5 selection methods. The best accuracy mean for the training partition was 0.977 for R_T5 and for the validation partition it was 0.978 for the T5_T5 method. For the training set, the difference was significant for T5_R vs. all other methods (all pairwise adjusted *p-values* in a Tukey HSD test smaller than 0.001). This method also had the worst training set accuracy amongst the best runs of all methods. The other methods were not significantly different among them. For the validation set, there was no significant difference among the methods (*p-value* 0.267 for a one-way ANOVA test).

The selection methods for first and second parents have different effects in PIC GP. The first parent operator just controls the selection pressure inside the subpopulation of one single specialisation class. Apart from the selection pressure, the second parent operator also

controls the interaction among specialists of different classes. This can be seen in Figure 12, which shows the number of interactions between specialists from different classes for the tested selection methods.

	Mean ± (sd)		Best run	
	Train	Validation	Train	Validation
R_T3	0.973 ± (0.014)	0.953 ± (0.056)	0.992	1.000
R_T5	**0.977** ± (0.011)	0.967 ± (0.042)	0.992	1.000
T3_R	0.974 ± (0.016)	0.953 ± (0.047)	**1.000**	**1.000**
T3_T3	0.974 ± (0.015)	0.967 ± (0.042)	0.992	1.000
T5_R	0.944 ± (0.049)	0.951 ± (0.068)	0.983	1.000
T5_T5	0.974 ± (0.018)	**0.978** ± (0.036)	**1.000**	**1.000**

Table 5: Mean, one standard deviation and the best accuracy for training and validation sets in the specialists selection method experiments. In bold are the best values of the respective column.

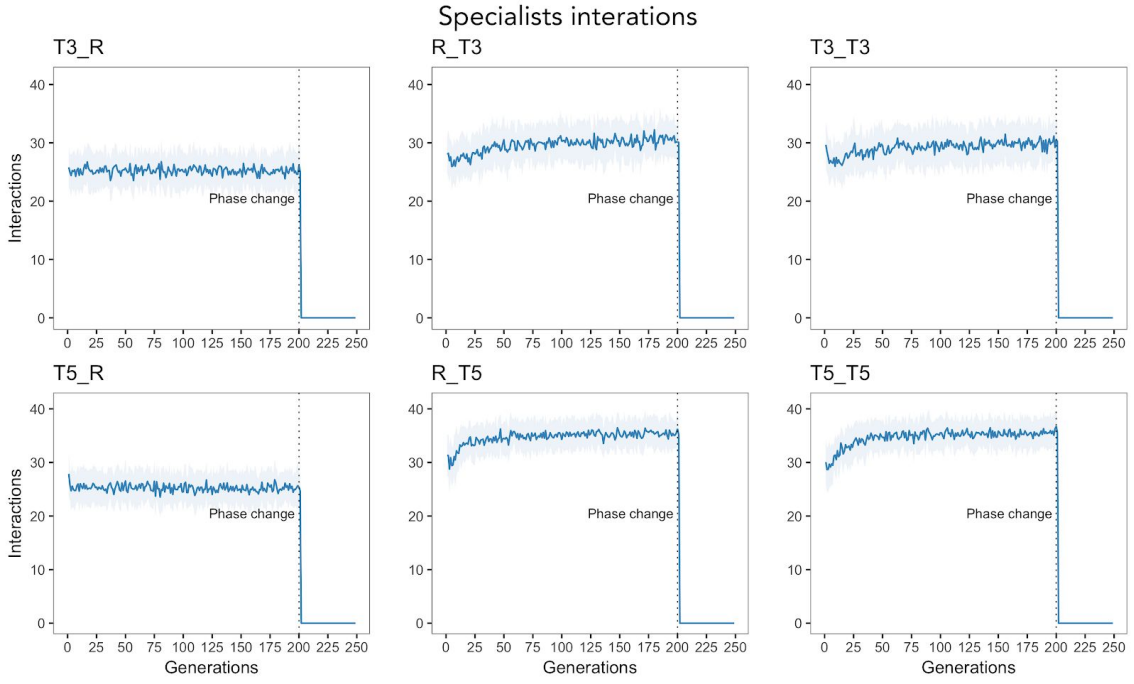

Figure 12: Specialists interactions mean and one standard deviation through GP evolution for each specialists selection method.

Independently of the selection method used for the first parent, the interaction among specialists of different classes were the same for the same selection method used in the second parent (T3_R is very similar to T5_R; R_T3 to T3_T3; R_T5 to T5_T5). Moreover, the bigger the tournament for the second parent, the more interactions among different specialists happened in each generation, as can be seen in the same plots. The first step of tournament selection is completely random, *i.e.* it is not related with the individuals'

fitnesses. However, the number of individuals in each class subpopulation affects the selection pressure, favouring individuals of the more abundant subpopulation. In addition to the fact that the first parent is chosen to balance the number of individuals among the specialisation classes, favouring the more abundant class in the second parent selection increased the interactions among classes. Since the *setosa* species is the easiest to discriminate, its specialists tend to have higher fitness and, hence, to be more prevalent in the algorithm population. This can be seen in Figures 13 and 14, which show the number of individuals and the mean fitness in each class subpopulation for each generation of the experiments T3_T3 and T5_T5. These plots show that in the demes phase of the algorithm the number of specialists in subpopulations of the classes *versicolor* and *virginica* tended to decrease while the number of *setosa* specialists tended to increase. These plots also show that the fitness of *setosa* specialists tended to be higher than the fitness of *versicolor* and *virginica* specialists, giving to *setosa* individuals an even higher competitive advantage.

In addition to this selection advantage, the higher number of *setosa* individuals is also related to the higher proportion of offspring generated with this specialisation class. As the *setosa* is the easiest class to classify, the individuals' fitness for this class will tend to be higher and, hence, in PIC GP most of the new individuals will be set to be specialised in this class.

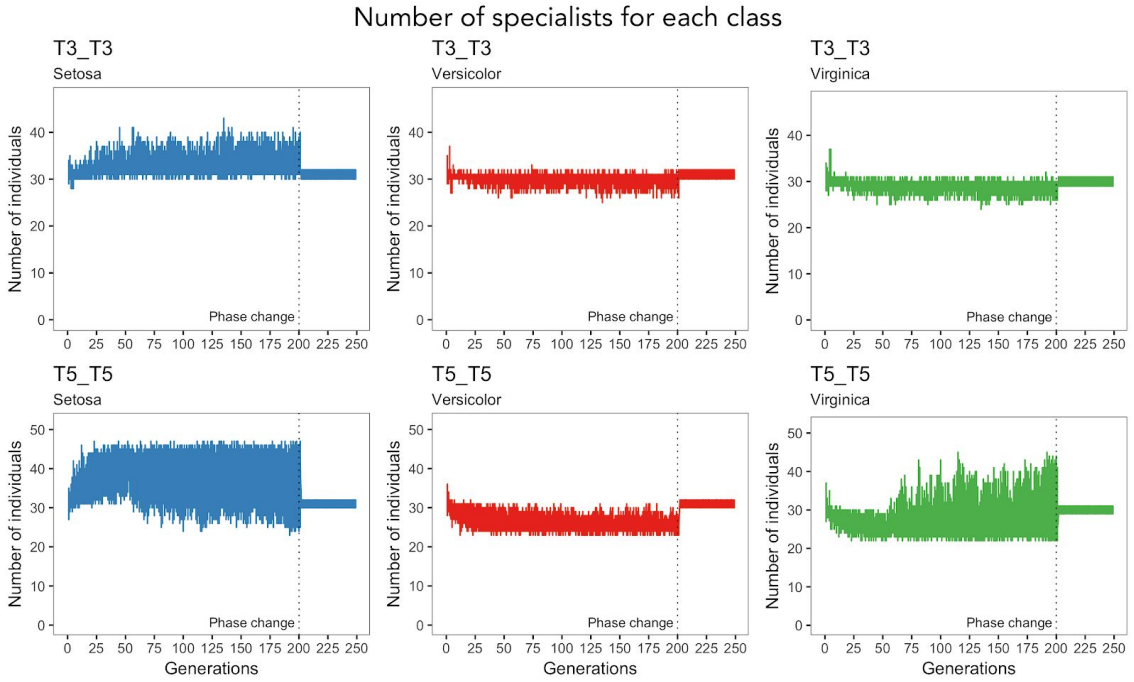

Figure 13: Number of specialised individuals in each class subpopulation through GP evolution for each specialists selection method.

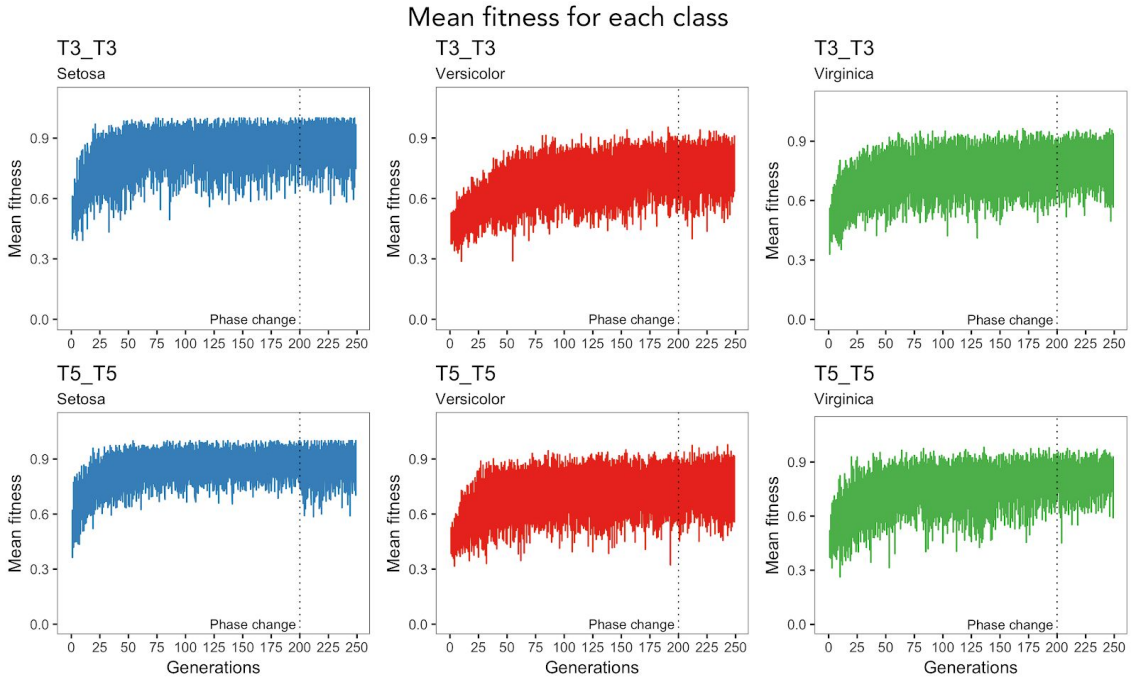

Figure 14: Mean of fitness of training partition in each class subpopulation through GP evolution for each specialists selection method.

The selection methods with the tested settings were not important for team fitness, tree sizes, genotype and phenotype population diversities and genotype and phenotype subpopulation diversities. Detailed results are presented in Appendix A.

4.2.2. Specialists fitness measures

The specialist fitness methods used in the experiments are listed in Table 6.

Method	Description
Accuracy	The accuracy of the individual classification for its own class. Eq. (2).
Fuzzy accuracy	The accuracy for its own class with correct answers weighted by its logistic values. Eq. (3).
ROC	The convex hull of the ROC curve for the threshold 0.5 for the individual classification for its own class. Eq. (4).
f-score	The *f*-score of the individual classification for its own class. Eq. (7).

Table 6: Specialists fitness measures used in experiments.

There was no significant difference among specialists fitness measures for final GP accuracy (*p-value* 0.323 for training set and 0.549 for validation set in a one-way ANOVA test), as can be seen in Table 7. The best mean in the training set was found using the ROC fitness measure and in the validation test it was found using the *f*-score. Again, all the best validation set accuracies were 1.00.

	Mean ± (sd)		Best run	
	Train	Validation	Train	Validation
Accuracy	0.969 ± (0.020)	0.962 ± (0.053)	**0.992**	**1.000**
Fuzzy accuracy	0.974 ± (0.014)	0.956 ± (0.051)	**0.992**	**1.000**
ROC	**0.977** ± (0.009)	0.949 ± (0.057)	**0.992**	**1.000**
f-score	0.974 ± (0.015)	**0.967** ± (0.042)	**0.992**	**1.000**

Table 7: Mean, one standard deviation and the best accuracy for training and validation sets in the specialists fitness measures experiments. In bold are the best values of the respective column.

The ROC and the *f*-score fitnesses do not consider only the overall correct classification without differentiating if the correct predictions are in positive/negative or most/less frequent classes. This can be important when the dataset is unbalanced and, for a more complex dataset, the different fitness measures may affect the final algorithm classification accuracy. Consider an extreme (but not unusual) condition in which a classifier predicts all instances for the same class. If the correspondent class is prevalent, the accuracy will be high. The ROC fitness is the arithmetic mean between the true positive and the true negative rates and it will balance these results. The *f*-score is the harmonic mean between precision and recall, which respectively assess the correctness of the positive predictions and the ability of the classifier to find the positive instances of the data. This offset will also balance the results.

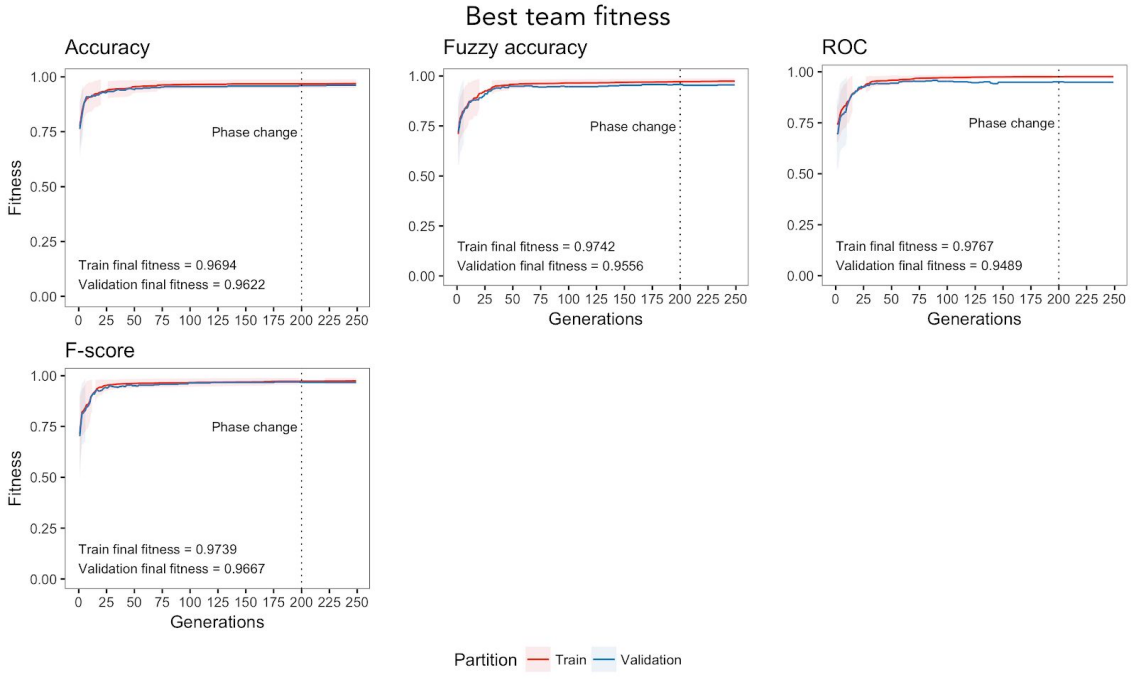

Figure 15: Mean and one standard deviation of train and validation best team fitness through the GP evolution for each specialists fitness method.

Figure 15 shows the algorithm fitness evolution for the four experiments. The differences in the beginning of the evolution process are small and indicate that for this dataset and GP settings, the specialists fitness measure was not important to control the algorithm speedness of convergence.

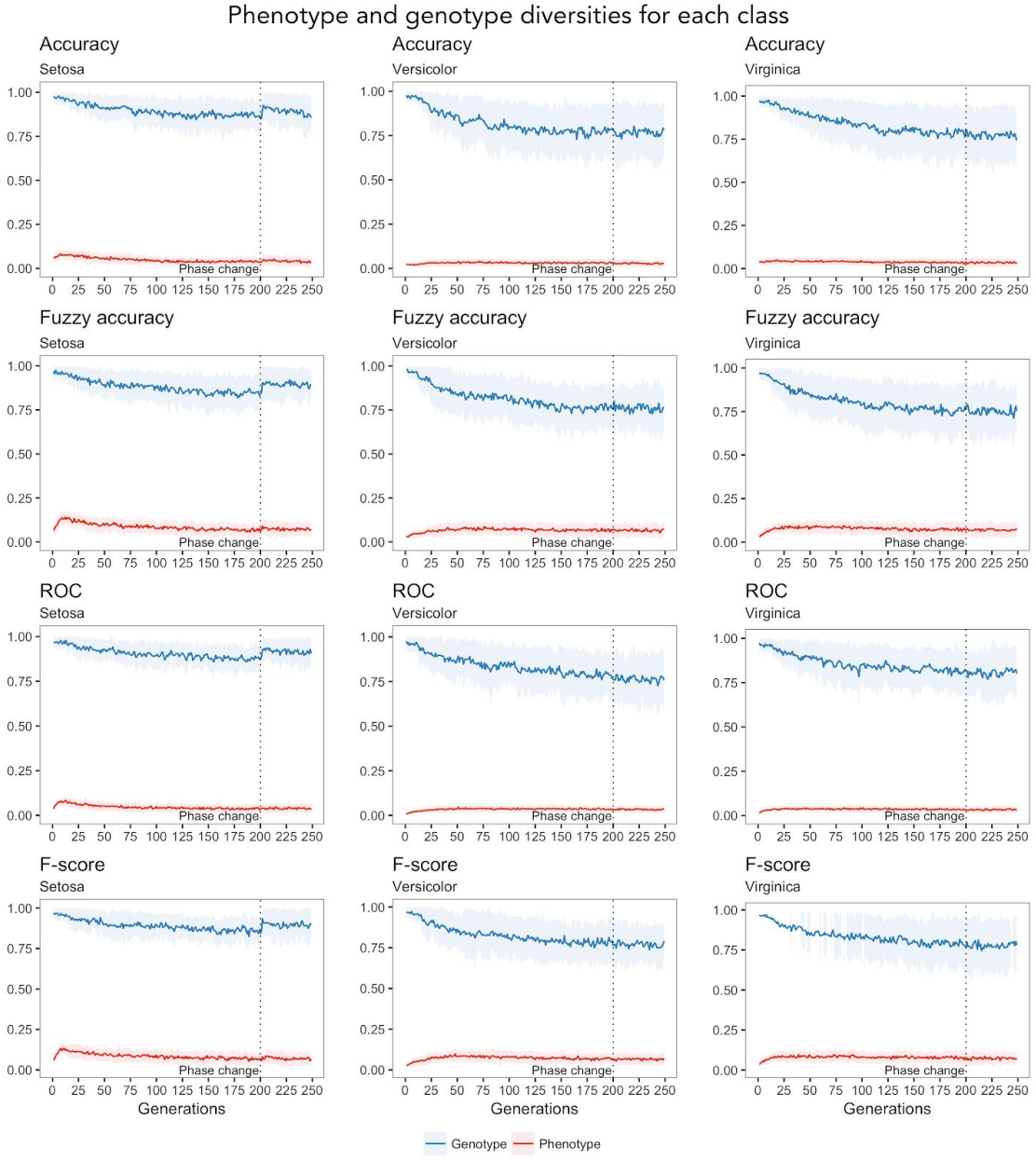

Figure 16: Genotype and phenotype diversities mean and one standard deviation for each class subpopulation through GP evolution for each specialists fitness method.

Figure 16 shows the phenotype and the genotype diversities for each specialisation class. The maximum of the phenotype diversity is reached in the beginning of the evolution for *setosa* and this is more prominent for fuzzy accuracy and *f*-scores measures. Due to the very

numerical nature of the specialists' fitness measures, these two can vary more precisely and, hence, their use will allow the creation of populations with individuals with more phenotype diversity. Table 8 presents numerically this result, by showing the maximum value of the specialised classes phenotype diversity and the corresponding generation in which this was achieved. The pattern is similar for the entire population, but since the subpopulations showed differences in phenotype diversity, class-specific results are presented.

For each class subpopulation, the higher values of phenotype diversity were achieved using either the fuzzy accuracy or the *f*-score fitness measures. In a scenario in which the algorithm is converging prematurely, it can be helpful to have a fitness measure that allows better discrimination of the individuals. For all the fitness measures, the *setosa* class subpopulation had higher diversity than the other two.

Method	Species	Maximum value	Generation
Accuracy	*setosa*	0.081	8
	versicolor	0.040	71
	virginica	0.050	18
Fuzzy accuracy	*setosa*	**0.140**	14
	versicolor	0.087	76
	virginica	**0.097**	30
ROC	*setosa*	0.084	13
	versicolor	0.045	51
	virginica	0.043	83
f-score	*setosa*	0.134	11
	versicolor	**0.099**	48
	virginica	**0.097**	90

Table 8: Maximum phenotype diversity achieved for each class subpopulation and the generation in which it was achieved. In bold are the highest values for each class subpopulation.

The specialists fitness measures with the tested settings were not important for team fitness, tree sizes and specialists interactions. Detailed results are presented in Appendix A.

4.2.3. Phase change

The phase change experiments tested different generations in which the algorithm design changed from demes to islands, as presented in Table 9.

For the cooperation intensity rate used in the experiments (equals 1.0 during the entire evolution of the algorithm) and the other algorithm settings, there was no difference among the four methods tested. Table 10 presents the mean and best final PIC GP accuracy for each of the four experiments. The differences were not significant (*p-values* 0.819 for training set

and 0.346 for validation set with a one-way ANOVA test). Again, all the best validation set accuracies were 1.00.

Method	Description
Full islands	The algorithm worked with island subpopulations (no interactions) during the entire evolution.
Gen 125	The algorithm worked with demes subpopulations (with interactions) until generation 125 (50% of total generations).
Gen 200	The algorithm worked with demes subpopulations until generation 200 (80% of total generations).
Full demes	The algorithm worked with demes subpopulations during the entire evolution.

Table 9: Phase change generations used in the experiments.

	Mean ± (sd)		Best run	
	Train	Validation	Train	Validation
Full islands	0.972 ± (0.018)	0.962 ± (0.049)	**0.992**	**1.000**
Gen 125	0.971 ± (0.012)	**0.980** ± (0.036)	0.983	1.000
Gen 200	**0.974** ± (0.015)	0.967 ± (0.042)	**0.992**	**1.000**
Full demes	0.970 ± (0.019)	0.960 ± (0.057)	**0.992**	**1.000**

Table 10: Mean, one standard deviation and the best accuracy for training and validation sets in the phase change experiments. In bold are the best values of the respective column.

The phase change with the tested settings also were not important for team fitness, specialists interactions, tree sizes, genotype and phenotype population diversities and genotype and phenotype subpopulations diversities. Detailed results are presented in Appendix A.

4.2.4. Cooperation intensity rate

4.2.4.1. Initial rate (CIR_0)

The following values of CIR were tested: 0.0, 0.2, 0.4, 0.6, 0.8 and 1.0. For all experiments, the rate was kept constant for the entire PIC GP evolution.

For all CIR tested values, the best runs achieved accuracy of 1.000 for the validation set (Table 11). For CIR 0.6 and CIR 0.8, the best accuracy in the training set was also 1.000. The highest mean accuracy in the training and validation sets were obtained for CIR 0.8 and CIR 0.6, respectively. Nevertheless, the differences among CIRs were not statistically significant (the one-way ANOVA test for the differences among the training fitness had the *p-value* 0.377 and for the differences among the validation fitness it was 0.843).

	Mean ± (sd)		Best run	
CIR_0	Train	Validation	Train	Validation
0.0	0.975 ± (0.011)	0.958 ± (0.057)	0.992	1.000
0.2	0.976 ± (0.012)	0.969 ± (0.060)	0.992	1.000
0.4	0.971 ± (0.019)	0.969 ± (0.038)	0.992	1.000
0.6	0.977 ± (0.010)	**0.973** ± (0.041)	**1.000**	**1.000**
0.8	**0.978** ± (0.011)	0.971 ± 0.034	**1.000**	**1.000**
1.0	0.974 ± (0.015)	0.967 ± (0.042)	0.992	1.000

Table 11: Mean, one standard deviation and the best accuracy for training and validation sets in the CIR experiments. In bold are the best values of the respective column.

The number of specialists' interactions presented different behavior for different CIR values, as seen in Figure 17. The specialists' interactions decreased slightly over the algorithm evolution in experiments with CIR from 0.0 to 0.4. For CIR 0.6 and CIR 0.8, they reached a maximum in the early generations, decreasing afterwards. For CIR 1.0, in contrast, they increased over the evolution process.

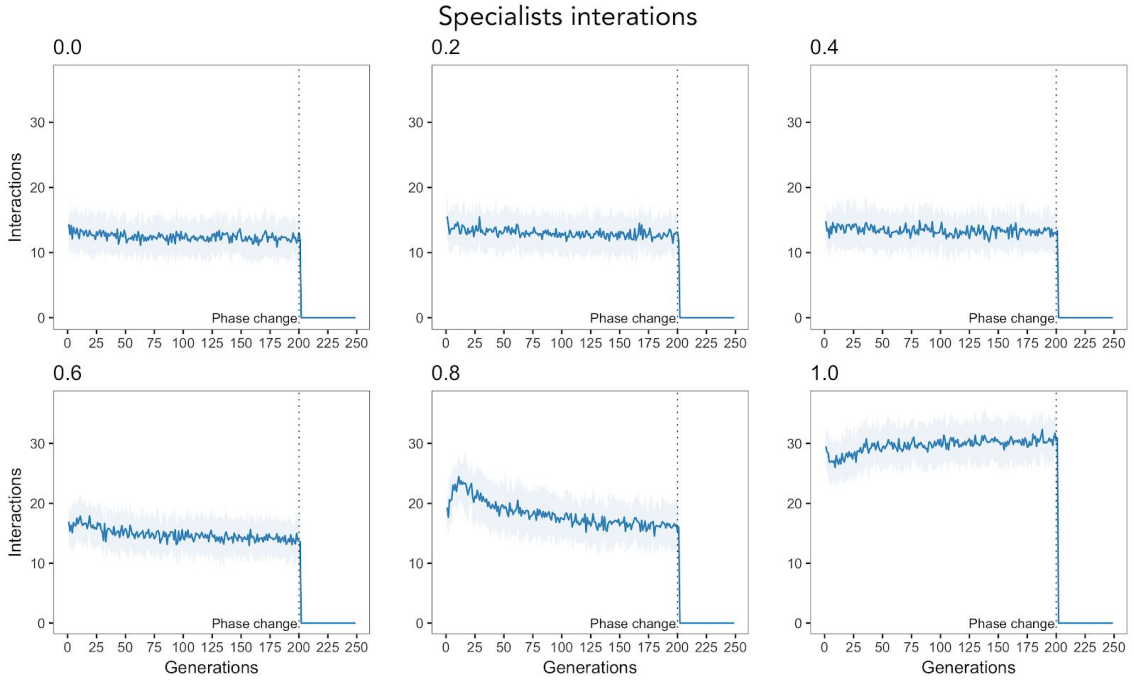

Figure 17: Specialists interactions mean and one standard deviation through GP evolution for each CIR values experiment.

In general, the specialists' interactions increased with the increase of CIR. However, this correlation was not linear because it also depends on the number of individuals in each class specialisation, as explained in the section describing the specialists' selection experiments. The bigger the difference in the number of individuals among the specialised subpopulations, the

more the individuals of different specialisations will interact when using the tournament selection for the second parent. Figure 18 shows that the pattern of the number of specialists in each class was different for different CIR values. For CIR values from 0.0 to 0.4, the number of individuals in each specialisation subpopulation was stable and balanced. For CIR 0.6 to 0.8, the *setosa* specialists started to prevail at the expense of the decrease of the *versicolor* and *virginica* number of specialists. But this pattern tended to smooth with the algorithm evolution, more intensely for CIR 0.6 and less for CIR 0.8. For CIR 1.0 the prevalence of *setosa* individuals lasted for the entire demes phase of the algorithm.

Also as a consequence of the differences in the number of individuals among specialisation classes, the genotype and the phenotype diversities in the subpopulations were affected, as can be seen in Figure 19.

With CIR 0.0, the phenotype diversity of all class subpopulations decreased after changing from demes to island phase of the algorithm. This also happened with CIR 0.2 and CIR 0.4 for the *setosa* class. This decrease in phenotype diversity shows that the diversity in the demes phase was maintained through interactions of individuals from different specialisations. However, with lower interaction rate, this diversity was not enough to produce good search space exploration. It was kept "artificially" by the interaction among different specialists and as soon as the subpopulations were separated in islands, the phenotype diversity decreased. With CIR 0.0 to CIR 0.8, the genotype diversity of *versicolor* and *virginica* classes (the weaker classifiers) also had a pronounced decrease after the algorithm phase change. For the *setosa* (the strongest classifiers) this happened only for CIR 0.6. For CIR of 1.0, the genotype diversity presented the opposite behavior for the *setosa* subpopulation: it increased after the phase change. Hence, for the classifiers of this class, the island approach was better in terms of genotype diversity. For the weaker classifiers the genotype diversity using CIR 1.0 was kept after the phase change.

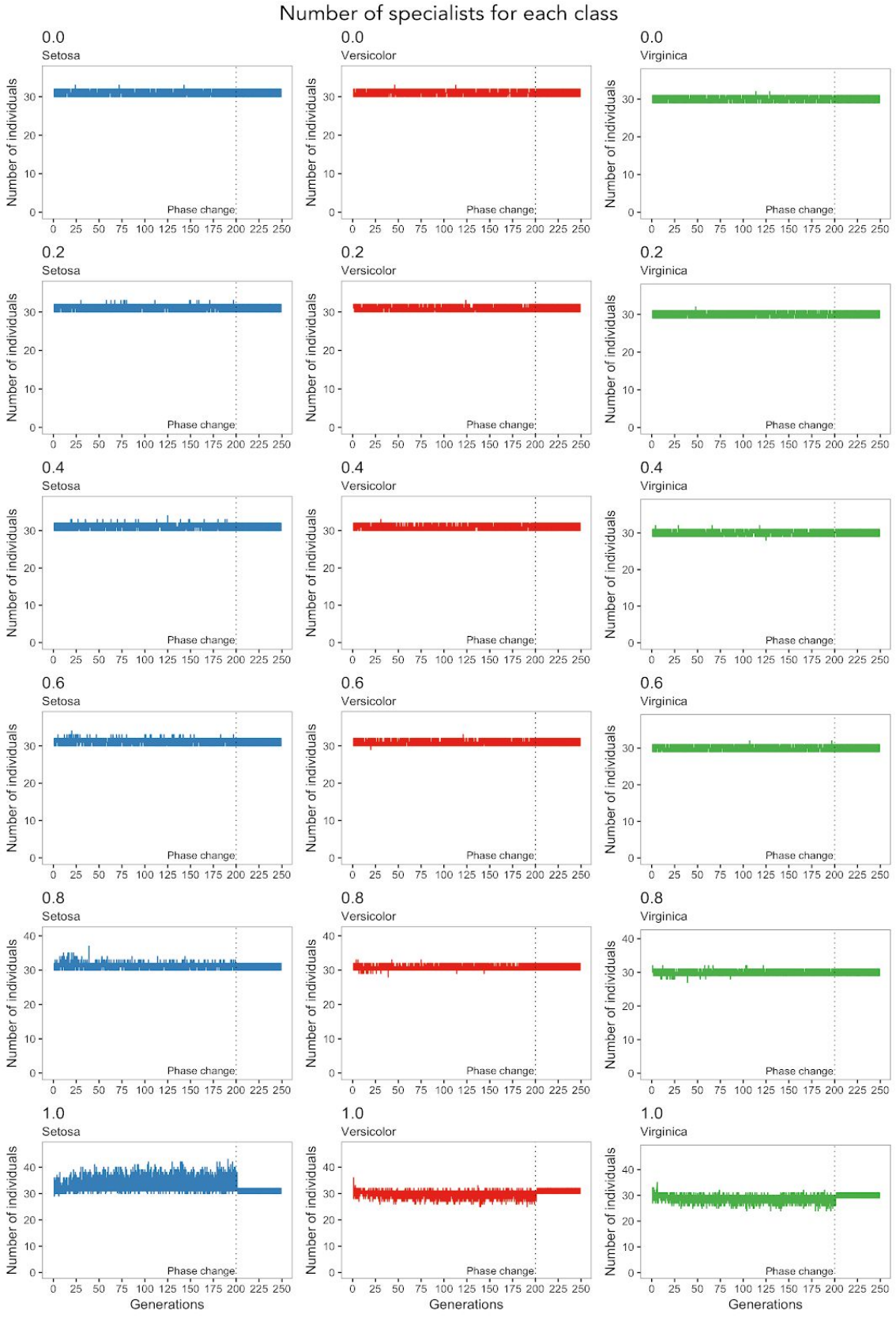

Figure 18: Number of specialised individuals in each class subpopulation through GP evolution for each CIR values experiment.

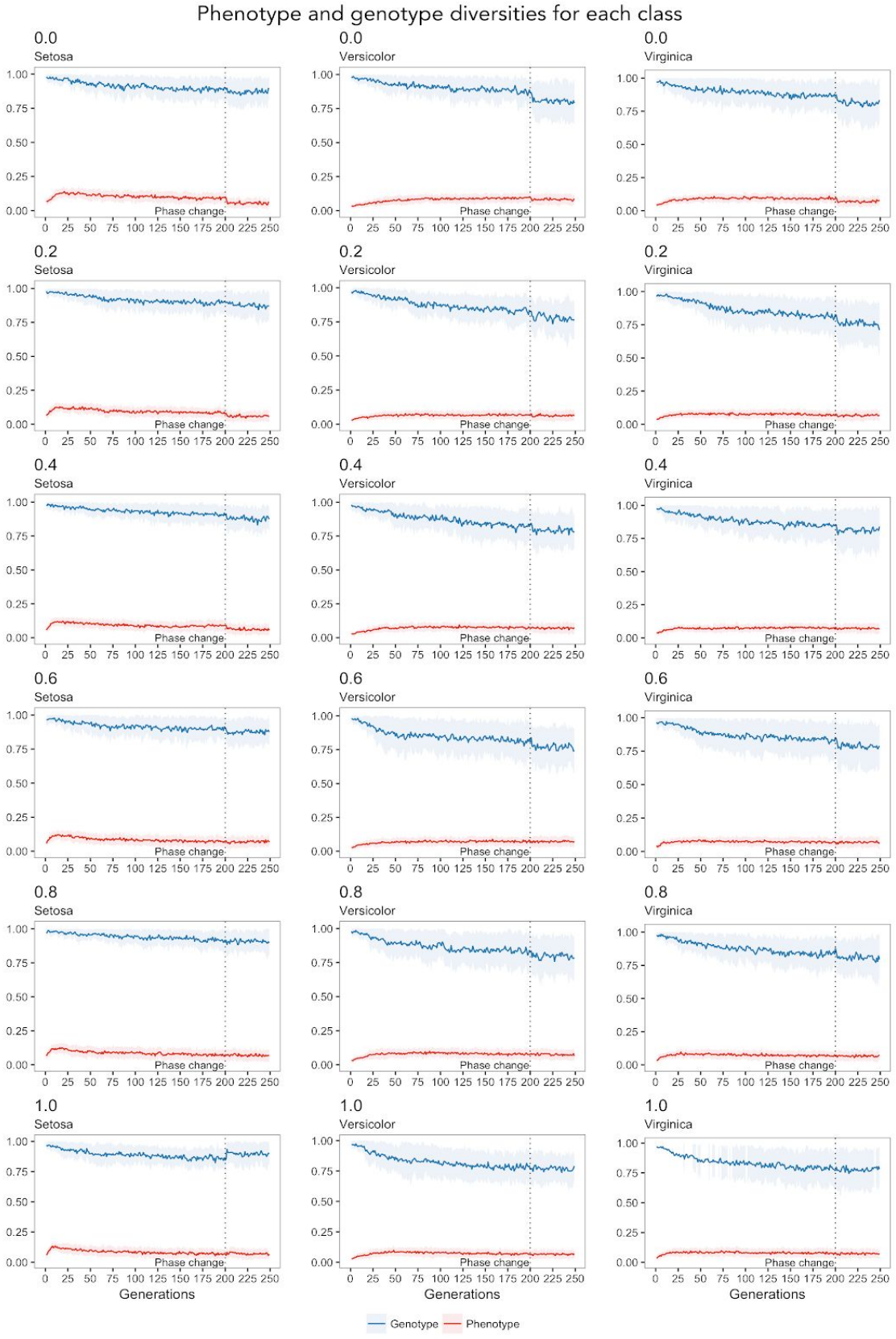

Figure 19: Genotype and phenotype diversities mean and one standard deviation for each class subpopulation through GP evolution for each CIR values experiment.

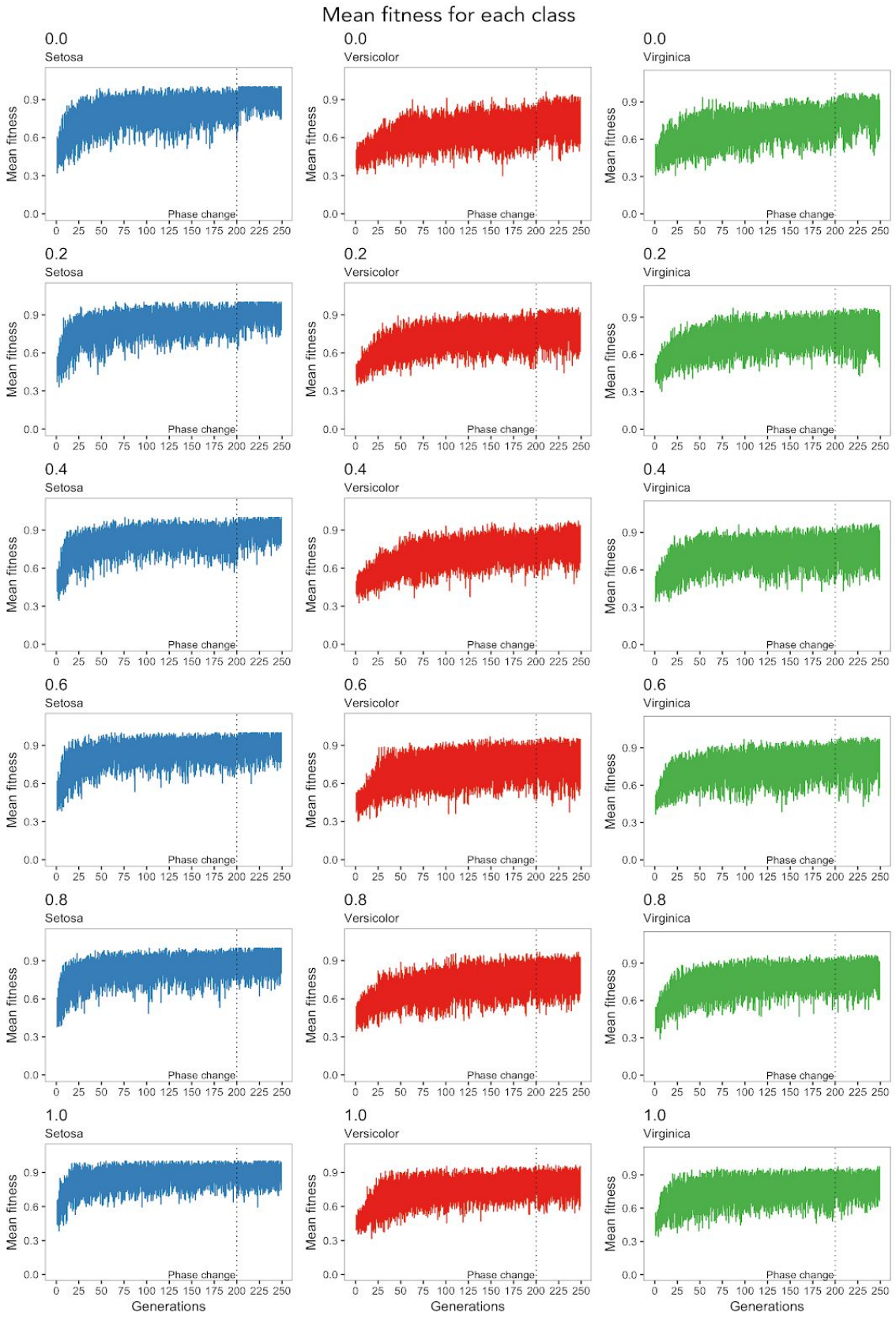

Figure 20: Mean of fitness of training partition in each class subpopulation through GP evolution for each CIR values experiment.

The CIR value was also important for the mean fitness of class subpopulations over the evolution process. With higher CIR values, *i.e.* with more interactions between specialists of different classes, the mean fitness of the class subpopulations increased earlier for all classes. Before the phase change, *setosa* and *versicolor* classes' subpopulations had higher mean fitness for CIR 1.0 and the *virginica* class for CIR 0.8. Comparing only the values before the phase change, the weakest class subpopulation (*versicolor*, the hardest to separate) presented the greatest difference (0.128) between the mean fitness with CIR 0.0 and with CIR 1.0. For the *virginica* class subpopulation this difference was 0.055 and for the *setosa* it was 0.048. After the phase change, *setosa* had the higher mean fitness with CIR 0.0 (0.913), *versicolor* with CIR 1.0 (0.811) and *virginica* with CIR 0.0 and CIR 0.6 (0.818). This can be seen in Table 12 and in Figure 20.

CIR	Class	Before	After	Difference
0.0	*setosa*	0.845	**0.913**	0.086
	versicolor	0.671	0.757	0.077
	virginica	0.741	**0.818**	0.064
0.2	*setosa*	0.856	0.910	0.054
	versicolor	0.743	0.791	0.049
	virginica	0.780	0.816	0.037
0.4	*setosa*	0.843	0.902	0.059
	versicolor	0.756	0.794	0.038
	virginica	0.763	0.799	0.035
0.6	*setosa*	0.873	0.897	0.024
	versicolor	0.754	0.793	0.039
	virginica	0.780	**0.818**	0.038
0.8	*setosa*	0.873	0.893	0.020
	versicolor	0.752	0.780	0.028
	virginica	**0.802**	0.816	0.013
1.0	*setosa*	**0.893**	0.892	-0.001
	versicolor	**0.799**	**0.811**	0.012
	virginica	0.796	0.809	0.014

Table 12: Mean of differences between the mean fitnesses of the class subpopulations 50 generations before and 50 generations after the phase change. In bold are the best values for each class before and after the phase change.

Table 12 also shows that the smaller the CIR value, the bigger the increase of the subpopulations' mean fitness after the phase change. This is due to both smaller mean fitness values before the phase change and higher values after the phase change. Although with CIR 0.0 the mean fitness of the specialised subpopulations increased more with the phase change (as described above), for the weakest classifier subpopulation (*versicolor*) the highest mean fitness was reached with CIR 1.0 after phase change. This was not observed for the strongest

classifier subpopulation (*setosa*), for which the highest mean fitness was reached with CIR 0.0 after phase change. Thus, a higher CIR value favoured an improvement of the subpopulation of weaker classifiers for the final algorithm prediction.

Two interesting conclusions can be drawn from the results presented above. First, that before the phase change, the interaction among the specialists of different classes was helping especially the weaker classifiers. Second, that the islands phase is also important for subpopulations to evolve. Both demes and islands seem to be important for best algorithm evolution.

The different CIR values with the tested settings were not important for team fitness, tree sizes and genotype and phenotype population diversities. Detailed results are presented in Appendix A.

4.2.4.2. Rate decrease

The following values of CIR rate decrease were tested: 0.00, 0.05 and 0.10. For 0.05 and 0.10 values, as the initial rate was 1.00 for all experiments, the CIR becomes smaller than 0.001 in generations 136 and 67, respectively, *i.e.* too early in the evolution process. Therefore, the evolution patterns for (i) interactions between specialists, (ii) population and subpopulation diversities, (iii) subpopulation mean fitness and (iv) number of individuals were similar to those presented in the previous section (see Appendix A).

Table 13 presents the algorithm mean, one standard deviation and best accuracies for each CIR decrease experiment. There was no significant difference among them (*p-values* of 0.721 for training set and 0.522 for validation set). Again, all the best validation set accuracies were 1.00. Lower decrease rates would need to be tested to allow for a more detailed discussion.

	Mean ± (sd)		Best run	
CIR decrease	Train	Validation	Train	Validation
0.00	**0.974** ± (0.015)	0.967 ± (0.042)	0.992	1.000
0.05	0.971 ± (0.020)	**0.973** ± (0.038)	**1.000**	**1.000**
0.10	**0.974** ± (0.012)	0.960 ± (0.054)	0.983	1.000

Table 13: Mean, one standard deviation and the best accuracy for training and validation sets in the CIR decrease rate experiments. In bold are the best values of the respective column.

4.2.5. Team prediction

The team prediction methods used in the team predictions are presented in Table 14.

Method	Description
Softmax	The softmax (Eq. 9) of the team members logistic values.
Specialist weighted	The softmax of the team members logistic values weighted by the members' fitnesses (Eq. 10).

Table 14: Team prediction methods used in experiments.

Table 15 shows the final fitness results obtained for the Softmax and Specialist weighted methods. There was no significant difference between these two methods (*p-value* 0.230 for a t-test for the training set means and 0.682 for the validation set means). This could be due to the small differences among the team members' fitness and, therefore, the small differences in the weights used to adjust the logistic outcomes of the specialists. In the weighted accuracy experiments, the mean of the final fitness in *setosa* specialists was 0.893, in *versicolor* specialists it was 0.791 and in *virginica* specialists it was 0.818. Again, all the best validation set accuracies were 1.00.

	Mean ± (sd)		Best run	
	Train	Validation	Train	Validation
Softmax	**0.974** ± (0.015)	**0.967** ± (0.042)	**0.992**	**1.000**
Specialist weighted	0.969 ± (0.018)	0.962 ± (0.042)	**0.992**	**1.000**

Table 15: Mean, one standard deviation and the best accuracy for training and validation sets in the team prediction experiments. In bold are the best values of the respective column.

The different team prediction methods with the tested settings were not important for all other algorithm measurements. Detailed results are presented in Appendix A.

4.2.6. Team evolution

For the experiments on team evolution the following settings were used (Table 16):

PIC GP Setting	**Value**
Team fitness measure	Accuracy
Teams population size	5
Selection method	Tournament size 3
Crossover probability	0.5
Mutation probability	0.5
Mutation operator	Specialist
Team elitism	True
Team fitness method	Accuracy

Table 16: PIC GP settings used in teams' experiments.

4.2.6.1. Teams mutation operator

The following team mutation operator methods were used in the experiments (Table 17):

Method	Description
Random	A random specialist of the team is replaced by a new random tree.
Specialist	A random specialist of the team is replaced by a specialist of the same specialisation from the specialists population.
Weaker	A specialist of the team is probabilistically chosen according to the inverse of its fitness to be replaced by a specialist of the same specialisation from the specialists population.

Table 17: Team mutation operator methods used in the experiments.

The results for the teams mutation operator experiments show that the specialist method was slightly better than the others for the generalisation ability of the algorithm (Figure 21). The mean of the validation fitness of the specialist method was greater than the mean of the training fitness through almost the entire evolution process.

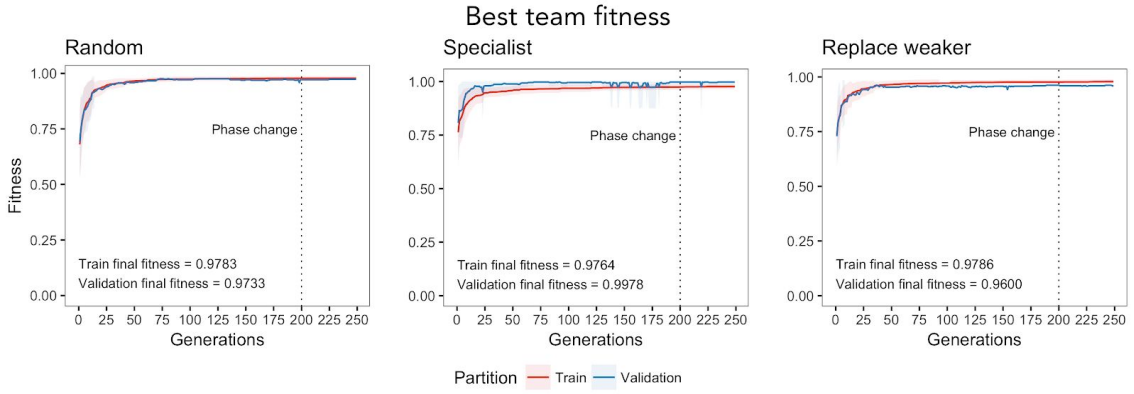

Figure 21: Mean and one standard deviation of train and validation best team fitness through the GP evolution for each team mutation operator.

Table 18 presents the final accuracy of PIC GP using the three teams mutation operator methods that were tested. Again, all the best validation set accuracies were 1.00. For the training set, the final mean accuracy of the algorithm was not significantly different among methods (*p-value* 0.673 with a one-way ANOVA). For the validation set, on the other hand, the difference was significant (*p-value* 0.011 in the one-way ANOVA) and a posterior Tukey's HSD test indicated that this difference was between the specialist and the weaker mutation operators (adjusted *p-value* 0.009 for this pair). Thus, the random mutation wasn't too innovative for this dataset. In fact, the results of these experiments showed that the group performance isn't compromised even by including a non-evolved individual in the team as occurs when using the random method. Moreover, the weaker mutation didn't improve the

team performance. These results demonstrate the strength of working in teams. For more complex classification problems though, it is expected that the random method will decrease the team performance because it becomes more crucial to work with stronger members.

	Mean ± (sd)		Best run	
	Train	Validation	Train	Validation
Random	0.976 ± (0.012)	0.969 ± (0.060)	**0.992**	**1.000**
Specialist	**0.977** ± (0.011)	**0.998** ± (0.012)	**0.992**	**1.000**
Weaker	0.974 ± (0.015)	0.967 ± (0.042)	**0.992**	**1.000**

Table 18: Mean, one standard deviation and the best accuracy for training and validation sets in the team mutation experiments. In bold are the best values of the respective column.

4.2.7. Final remarks

The classification accuracy was high both in the training and in the validation partitions for all experiments, indicating that, for this dataset, the algorithm performed well independently of the settings used. The best accuracy mean for the specialist experiments was achieved in phase change tests for a change halfway in the evolution process, at generation 125 (mean accuracy 0.971 for the training set and 0.980 for the validation set). The best of all means was achieved in the teams mutation operator experiments (mean accuracy 0.977 for the training set and 0.998 for the validation set obtained using the specialist method).

A comparison of the mean fitness obtained with the default experiment settings without teams evolution and with teams evolution using the specialist team mutation operator was made to assess the importance of teams evolution in PIC GP for the IRS dataset. The fitness means using teams evolution were 0.003 and 0.031 greater than without teams evolution for the training and validation sets, respectively. T-tests resulted in *p-values* 0.475 and <0.001, respectively. A statistically significant increase in accuracy in the validation set of 3% is considered relevant. It shows that the teams evolution improved the algorithm generalisation ability.

All experiments have shown a good generalisation ability in the model, as the validation set accuracy was always similar to the training set accuracy and the best validation set accuracy was always 1.00. This is a widely explored dataset and some published results indicate that it is easy to classify. Louis Ong in the Kaggle website published a study[1] with a multilayer NN in which the classification accuracy in the test set was also 1.00. In any case, there are also

[1] https://www.kaggle.com/louisong97/neural-network-approach-to-iris-dataset

studies in the literature in which the classifiers did not show such good performances. For example, Mendes et al. (2001) published a study on a co-evolutionary system for discovering fuzzy classification rules and their model had achieved the mean accuracy of 0.953 ($\pm$0.071) for the test set.

Tests with other combinations of the settings, like using the roulette wheel for the second parent specialists' selection or other cooperation intensity rates for the phase change experiments may impact more strongly on algorithm performance than observed in the experiments presented above. Moreover, tests with a dataset less linearly separable will probably reveal more differences amongst the settings tested in this work. However, the experiments presented above gave extremely valuable hints on how the selection method, the CIR and the demes to islands phase change can affect differently the subpopulations of the classifiers depending on their strength. These relationships have shown that the specialists' interactions can benefit the weaker classifiers and that the islands phase is important for the improvement of all classifiers.

4.3. Thyroid dataset

The thyroid (THY) dataset contains registers of patients for three thyroid disease states, namely, no disease, hypothyroidism and hyperthyroidism. It has 21 features, of which 15 are binary and 6 are real numbers, for 7200 observations. The disease state is the prediction class. Figures 22 and 23 present the distribution of these features for each target class. None of the features is able to linearly separate any of the classes and, hence, class discrimination is not easy for this dataset (overlapping distributions for continuous features and similar yes/no proportions for binary features).

As reported in Table 3, the best PIC GP run for the THY dataset, using the algorithm's default settings (Table 1), achieved an accuracy of 0.992 for the validation set.

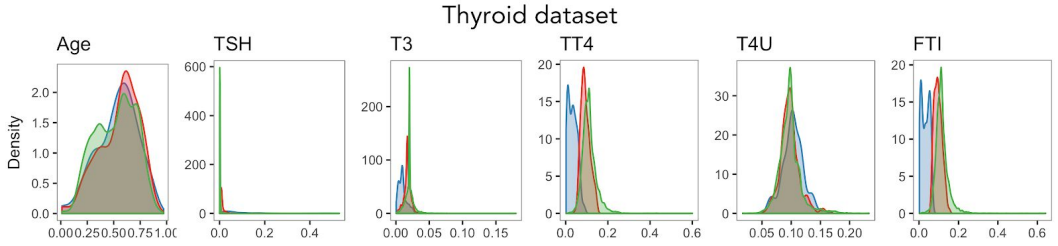

Figure 22: Thyroid dataset real number features distribution for each target class.

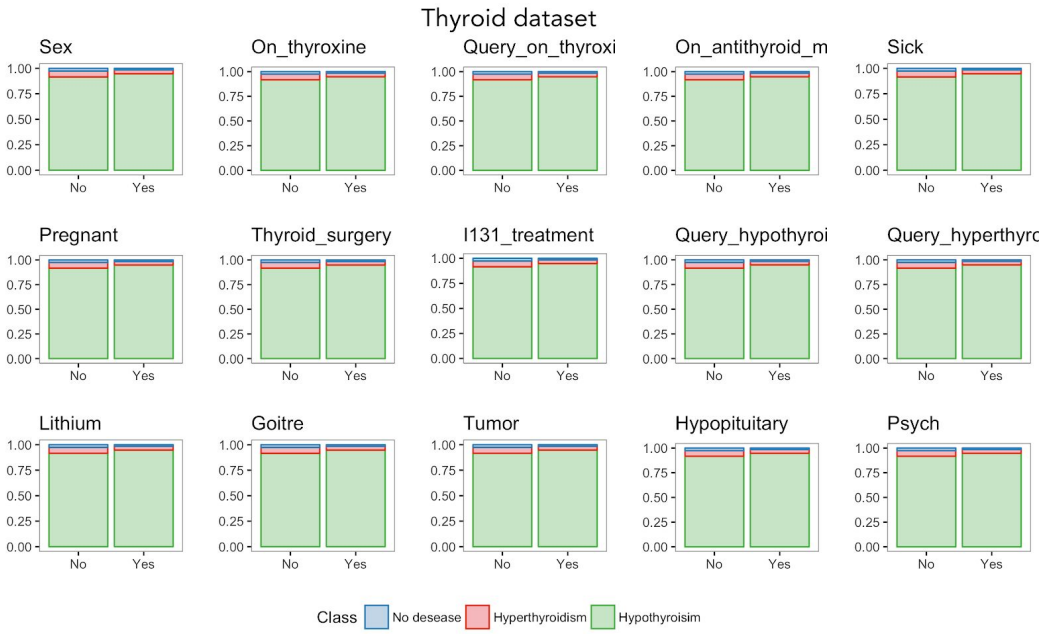

Figure 23: Thyroid dataset binary features distribution for each target class.

The algorithm performed quite well, since it outperformed some results found in literature without any fine tuning. For example, Tsakonas (2006) tested four grammar-guided GP configurations: with decision trees, with fuzzy rule-based training, with fuzzy petri-nets and with neural networks. The best training accuracies were respectively 0.988, 0.947, 0.942 and 0.940 for the training set and 0.976, 0.941, 0.940 and 0.940 for the validation set. Ionita & Ionita (2016) also compared methods of machine learning for this dataset. They found that the best runs for NB, Decision Trees, Multilayer Perceptron and Radial Basis Function Network achieved classification accuracies of 0.917, 0.969, 0.951 and 0.960, respectively. They also made a second experiment, manually removing *Query_on_thyroxine*, *Query_on_hypothyroid*, *Query_on_hyperthyroid* features and were able to improve the accuracy of the Decision Trees classifier to 0.973, but obtained worst results for the other classifiers. Finally, Zhang et al. (2017) published a comparison among the following machine learning algorithms for MCC: Stochastic Gradient Boosting Decision Trees (GBDT), Random Forests (RF), Extreme learning machine (ELM), Support Vector Machine (SVM), C4.5, Sparse Representation based Classification (SCR), KNN, Logistic Regression (LR), AdaBoost (AB), NB and Deep Learning (DL). The best accuracy that they could obtain for each classifier for the THY dataset is presented in Table 19. Of note, PIC GP outperformed 9 of the 11 algorithms tested by the authors and was worse than GDBT and RF.

Algorithm	Validation fitness	Algorithm	Validation fitness
PIC GP	0.992	SCR	0.903 (−)
GDBT	1.000 (+)	KNN	0.903 (−)
RF	1.000 (+)	LR	0.931 (−)
ELM	0.903 (−)	AB	0.931 (−)
SVM	0.903 (−)	NB	0.903 (−)
C4.5	0.986 (−)	DL	0.903 (−)

Table 19: Accuracy for PIC GP and the achieved accuracy for each classifier reported in Zhang et al. (2017) for the THY dataset. (+) indicates the algorithms that performed better than PIC GP, (-) those that performed worse and (=) the one that performed equal to PIC GP.

4.4. Yeast dataset

The Yeast (YST) dataset contains data from molecular analysis of yeast. The dataset has 8 real number features of biochemical analysis for 1484 observations. The target classes are 10 possible cellular localisation sites of proteins in the organism.

In Figure 24 it is possible to see that the features distributions are mostly overlapped. In features mcg, gvh, alm, mit, erl and pox it is possible to see two groups of overlapped features distributions. Thus, it is expected that his dataset is even more difficult to classify than IRS and THY.

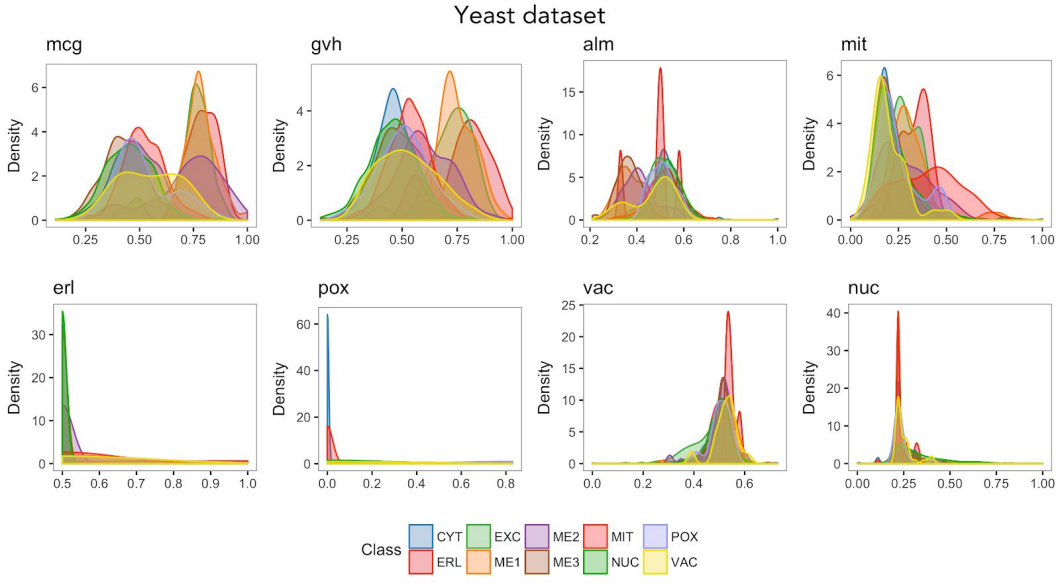

Figure 24: Yeast dataset features distribution for each target class.

As reported in Table 3, the best PIC GP run for the YST dataset, using the algorithm's default settings (Table 1), achieved an accuracy of 0.642 for the validation set.

The PIC GP performance for this dataset is comparable with some results found in literature without any fine tuning, again confirming the robustness of the algorithm. For a MCC GP wrapper algorithm, Muñoz et al. (2015) found a median of 0.562 (best run < 0.62) for classification accuracy for this dataset. The results for the 11 datasets studied by Zhang et al. (2017), described in the previous section, are presented in Table 20. The PIC GP outperformed 10 and it was worse than 1, but the difference to the best algorithm (ELM) was only 0.007 while to the worst (DL) was 0.311.

Algorithm	Validation fitness	Algorithm	Validation fitness
PIC GP	0.642	SCR	0.574 (−)
GDBT	0.622 (−)	KNN	0.574 (−)
RF	0.622 (−)	LR	0.621 (−)
ELM	0.649 (+)	AB	0.412 (−)
SVM	0.629 (−)	NB	0.595 (−)
C4.5	0.513 (−)	DL	0.331 (−)

Table 20: Accuracy for PIC GP and the achieved accuracy for each classifier reported in Zhang et al. (2017) for the YST dataset. (+) indicates the algorithms that performed better than PIC GP, (-) those that performed worse and (=) the one that performed equal to PIC GP.

5. Conclusions

The GP for multiclass classification (MCC) problems presented in this thesis combines the advantages of evolving both strong individuals and teams composed of these individuals. The algorithm is named Progressively Insular Cooperative (PIC) GP because its key feature is the possibility to decrease progressively the level of interaction between individuals specialised in classifying different classes followed by a complete separation of the specialised subpopulations, allowing them to evolve with no interaction with other subpopulations, *i.e.* as islands. Moreover, teams of individuals specialised in different classes also can evolve in an independent process from the evolution of the specialist individuals explained above. This allows teams to work with already improved individuals to find the best possible combination.

The modifications made in the standard GP for the evolution of specialist individuals were the introduction of new parameters, changes in the selection step and modifications in the individuals' fitness measures. The new parameters that were introduced are the cooperation intensity rate (CIR), the rate of CIR decrease over the algorithm evolution and the generation in which the algorithm starts working with totally separated specialised subpopulations (islands). The selection step was modified to control the interaction between specialists of different classes, using the introduced parameters. Different specialists' fitness measures were tested to improve the assessment of the individuals to better decide if they should or not participate from the teams. The modifications made in the team evolution were in the team prediction step and the team mutation operators.

The proposed PIC GP algorithm showed excellent performance for the three datasets tested (Iris, Thyroid and Yeast from the UCI Machine Learning repository). Still there is scope for further research with a more exhaustive exploration of its parameters. For example, with more complex datasets and with other combinations of the settings, as indicated in the experiments final remarks.

Importantly, the current work elucidated a major question that currently has no clear answer in the available literature. As seen in the CIR experiments, team-based GP can benefit both from the interaction between individuals specialised in different classes and from a more restricted approach with interaction only between individuals of the same specialisation class. The contribution of each approach to the algorithm's performance will depend on the performance of each group of specialised individuals. A demes approach helps weaker groups of specialist classifiers because they may benefit from receiving crucial genetic material from

stronger groups. An island approach, on the other hand, will allow strong classifiers to evolve to their best potential. The results indicate that the combination of both approaches may be the best strategy, at least for some datasets. Starting with a demes approach is important to improve the weaker performers. Later, when all groups are strong, the algorithm can change to an islands approach to allow all the specialised classifiers to reach their best. The demes approach slows down the evolution process of stronger specialised groups, as expected for any kind of team work. However, it can benefit GP algorithms in MCC tasks. Indeed, the results presented here indicate that cooperation can benefit collective evolution. On the other hand, when all classifiers perform well from the beginning, it is expected that a demes approach would not be beneficial.

The recommended progression of the current work is to use this new information to transform the PIC in the Adaptative Insular Cooperative GP. In its adaptative form, instead of having the CIR, the CIR decrease and the phase change parameters fixed, they would change according to the class subpopulations performance. It would adjust the CIR trying to help the weaker classifiers and to change from demes to islands phase when the specialised subpopulations had reached a good performance.

In addition, a more extensive exploration of the teams' evolution is recommended. More specifically, using the team's evolution with Genetic Algorithms instead of GP. As the specialised trees do not evolve in this level of PIC GP, the teams will always have the same size and, therefore, they can be a Genetic Algorithm solution.

6. References

Al-Madi, N., & Ludwig, S. A. (2013). Improving genetic programming classification for binary and multiclass datasets. *2013 IEEE Symposium on Computational Intelligence and Data Mining (CIDM)*, 166–173.

Angeline, P. J. (1996). An investigation into the sensitivity of GP to the frequency of leaf selection during subtree crossover. In J. R. Koza, D. E. Goldberg, D. B. Fogel, & R. L. Riolo (Eds.), *Proceedings of the 1st annual conference on genetic programming*. The MIT Press.

Boucher, D. (2016). *Mutualism*. *Integrative and Comparative Biology*, *56*(2), 365–367.

Brameier, M., & Banzhaf, W. (2007). *Evolving Teams of Predictors with Linear Genetic Programming*. 27.

Burke, E., Gustafson, S., & Kendall, G. (2003). Ramped Half-n-Half Initialisation Bias in GP. In E. Cantú-Paz, J. A. Foster, K. Deb, L. D. Davis, R. Roy, U.-M. O'Reilly, H.-G. Beyer, R. Standish, G. Kendall, S. Wilson, M. Harman, J. Wegener, D. Dasgupta, M. A. Potter, A. C. Schultz, K. A. Dowsland, N. Jonoska, & J. Miller (Eds.), *Genetic and Evolutionary Computation—GECCO 2003* (Vol. 2724, pp. 1800–1801). Springer Berlin Heidelberg.

Chawla, N. V., Bowyer, K. W., Hall, L. O., & Kegelmeyer, W. P. (2002). SMOTE: Synthetic Minority Over-sampling Technique. *Journal of Artificial Intelligence Research*, *16*, 321–357.

Chen, Z., & Lu, S. (2007). A Genetic Programming Approach for Classification of Textures Based on Wavelet Analysis. *2007 IEEE International Symposium on Intelligent Signal Processing*, 1–6.

Chien, B.-C., Lin, J.-Y., & Yang, W.-P. (2004). Learning effective classifiers with Z-value measure based on genetic programming. *Pattern Recognition*, *37*(10), 1957–1972.

Eggermont, J., Eiben, A. E., & van Hemert, J. I. (1999). A Comparison of Genetic Programming Variants for Data Classification. In D. J. Hand, J. N. Kok, & M. R. Berthold (Eds.), *Advances in Intelligent Data Analysis* (Vol. 1642, pp. 281–290). Springer Berlin Heidelberg.

Espejo, P. G., Ventura, S., & Herrera, F. (2010). A Survey on the Application of Genetic Programming to Classification. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, *40*(2), 121–144.

Haynes, T., Sen, S., Schoenefeld, D., & Wainwright, R. (1995). Evolving a team. *Working Notes for the AAAI Symposium on Genetic Programming*.

Haynes, T., & Sen, S. (1997). *Crossover Operators for Evolving A Team*. 7.

Ionita, I., & Ionita, L. (2016). Prediction of thyroid Disease Using Data Mining Techniques. *BRAIN. Broad Research in Artificial Intelligence and Neuroscience*, *7*(3), 115–124.

Kinnear, K. E. (1993). Evolving a sort: Lessons in genetic programming. *IEEE International Conference on Neural Networks*, 881–888 vol.2.

Kishore, J. K., Patnaik, L. M., Mani, V., & Agrawal, V. K. (2000). Application of genetic programming for multicategory pattern classification. *IEEE Transactions on Evolutionary Computation*, *4*(3), 242–258.

Koza, J. R.. (1992). *Genetic programming: On the programming of computers by means of natural selection*. MIT Press.

Koza, J. R. (1994). Genetic programming as a means for programming computers by natural selection. *Statistics and Computing*, *4*(2).

Langdon, W. B. (2000). *Size Fair and Homologous Tree Crossovers for Tree Genetic Programming*. 25.

Leung, Y., Gao, Y., & Xu, Z.-B. (1997). Degree of population diversity—A perspective on premature

convergence in genetic algorithms and its Markov chain analysis. *IEEE Transactions on Neural Networks*, *8*(5), 1165–1176.

Lichodzijewski, P., & Heywood, M. I. (2008). Managing team-based problem solving with symbiotic bid-based genetic programming. *Proceedings of the 10th Annual Conference on Genetic and Evolutionary Computation - GECCO '08*, 363.

Lin, J.-Y., Ke, H.-R., Chien, B.-C., & Yang, W.-P. (2007). Designing a classifier by a layered multi-population genetic programming approach. *Pattern Recognition*, *40*(8), 2211–2225.

Luke, S., & Panait, L. (2006). A Comparison of Bloat Control Methods for Genetic Programming. *Evolutionary Computation*, *14*(3), 309–344.

Luke, S., & Spector, L. (1996). Evolving Teamwork and Coordination with Genetic Programming. *Genetic Programming 96 (GP96) Conference Proceedings*.

Mendes, R. R. F., de Voznika, F. B., Freitas, A. A., & Nievola, J. C. (2001). Discovering Fuzzy Classification Rules with Genetic Programming and Co-evolution. In L. De Raedt & A. Siebes (Eds.), *Principles of Data Mining and Knowledge Discovery* (Vol. 2168, pp. 314–325). Springer Berlin Heidelberg.

Montana, D. J. (1995). Strongly Typed Genetic Programming. *Evolutionary Computation*, *3*(2), 199–230.

Moraglio, A., Krawiec, K., & Johnson, C. G. (2012). Geometric Semantic Genetic Programming. In C. A. C. Coello, V. Cutello, K. Deb, S. Forrest, G. Nicosia, & M. Pavone (Eds.), *Parallel Problem Solving from Nature—PPSN XII* (Vol. 7491, pp. 21–31). Springer Berlin Heidelberg.

Muñoz, L., Silva, S., & Trujillo, L. (2015). M3GP – Multiclass Classification with GP. In P. Machado, M. I. Heywood, J. McDermott, M. Castelli, P. García-Sánchez, P. Burelli, S. Risi, & K. Sim (Eds.), *Genetic Programming* (Vol. 9025, pp. 78–91). Springer International Publishing.

Poli, R., Langdon, W. B., McPhee, N. F., & Koza, J. R. (2008). *A field guide to genetic programming*. Lulu Press].

Preussger, D., Giri, S., Muhsal, L. K., Oña, L., & Kost, C. (2020). Reciprocal Fitness Feedbacks Promote the Evolution of Mutualistic Cooperation. *Current Biology*, S0960982220309866.

Raymer, M. L., Punch, W. F., Goodman, E. D., & Kuhn, L. A. (1996). Genetic Programming 1996: Proceedings of the First Annual Conference, July 28-31, 1996, Stanford University. In J. R. Koza, D. E. Goldberg, D. B. Fogel, & R. L. Riolo (Eds.), *Genetic Programming 1996: Proceedings of the First Annual Conference, July 28-31, 1996, Stanford University*. The MIT Press.

Silva, S., & Tseng, Y.-T. (2008). Classification of Seafloor Habitats Using Genetic Programming. In M. Giacobini, A. Brabazon, S. Cagnoni, G. A. Di Caro, R. Drechsler, A. Ekárt, A. I. Esparcia-Alcázar, M. Farooq, A. Fink, J. McCormack, M. O'Neill, J. Romero, F. Rothlauf, G. Squillero, A. Ş. Uyar, & S. Yang (Eds.), *Applications of Evolutionary Computing* (Vol. 4974, pp. 315–324). Springer Berlin Heidelberg.

Smart, W., & Zhang, M. (2005). Using Genetic Programming for Multiclass Classification by Simultaneously Solving Component Binary Classification Problems. In M. Keijzer, A. Tettamanzi, P. Collet, J. van Hemert, & M. Tomassini (Eds.), *Genetic Programming* (Vol. 3447, pp. 227–239). Springer Berlin Heidelberg.

Soule, T., & Komireddy, P. (2007). Orthogonal evolution teams: a class of algorithms for evolving

teams with inversely correlated errors. Genetic programming theory and practice *IV*, 17.

Soule, T. (2000). Heterogeneity and Specialization in Evolving Teams. *Conference paper https://www.researchgate.net/publication/220740394.*

Tan, X., Bhanu, B., & Lin, Y. (2005). Fingerprint Classification Based on Learned Features. *IEEE Transactions on Systems, Man and Cybernetics, Part C (Applications and Reviews)*, *35*(3), 287–300.

Thomason, R., & Soule, T. (2007). Novel ways of improving cooperation and performance in ensemble classifiers. *Proceedings of the 9th Annual Conference on Genetic and Evolutionary Computation - GECCO '07*, 1708.

Tsakonas, A. (2006). A comparison of classification accuracy of four genetic programming-evolved intelligent structures. *Information Sciences*, *176*(6), 691–724.

Vanneschi, L., Bakurov, I., & Castelli, M. (2017). An initialization technique for geometric semantic GP based on demes evolution and despeciation. *2017 IEEE Congress on Evolutionary Computation (CEC)*, 113–120.

Vanneschi, L., & Poli, R. (2012). Genetic Programming—Introduction, Applications, Theory and Open Issues. In G. Rozenberg, T. Bäck, & J. N. Kok (Eds.), *Handbook of Natural Computing* (pp. 709–739). Springer Berlin Heidelberg.

Wilson, D. S. (1977). Structured Demes and the Evolution of Group-Advantageous Traits. *The American Naturalist*, *111*(977), 157–185.

Zhang, C., Liu, C., Zhang, X., & Almpanidis, G. (2017). An up-to-date comparison of state-of-the-art classification algorithms. *Expert Systems with Applications*, *82*, 128–150.

Zhang, M., & Smart, W. (2004). Multiclass Object Classification Using Genetic Programming. In G. R. Raidl, S. Cagnoni, J. Branke, D. W. Corne, R. Drechsler, Y. Jin, C. G. Johnson, P. Machado, E. Marchiori, F. Rothlauf, G. D. Smith, & G. Squillero (Eds.), *Applications of Evolutionary Computing* (Vol. 3005, pp. 369–378). Springer Berlin Heidelberg.

Appendix A

1. Experiments results

1.1. Iris Dataset

1.1.1. Specialists selection operator

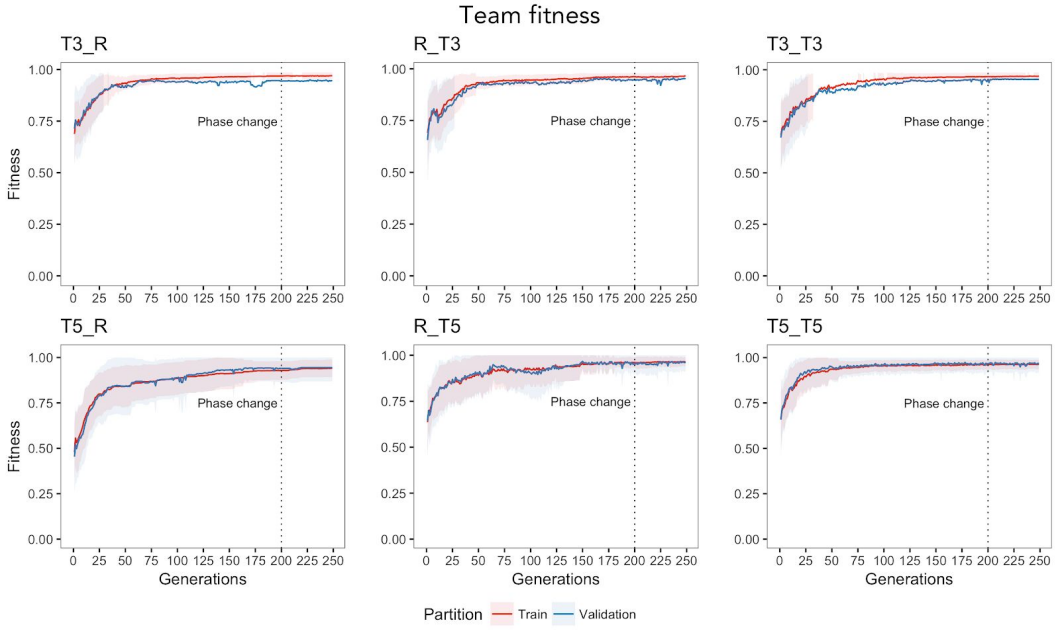

Figure 1: Mean and one standard deviation of train and validation team fitness through the GP evolution for each selection method.

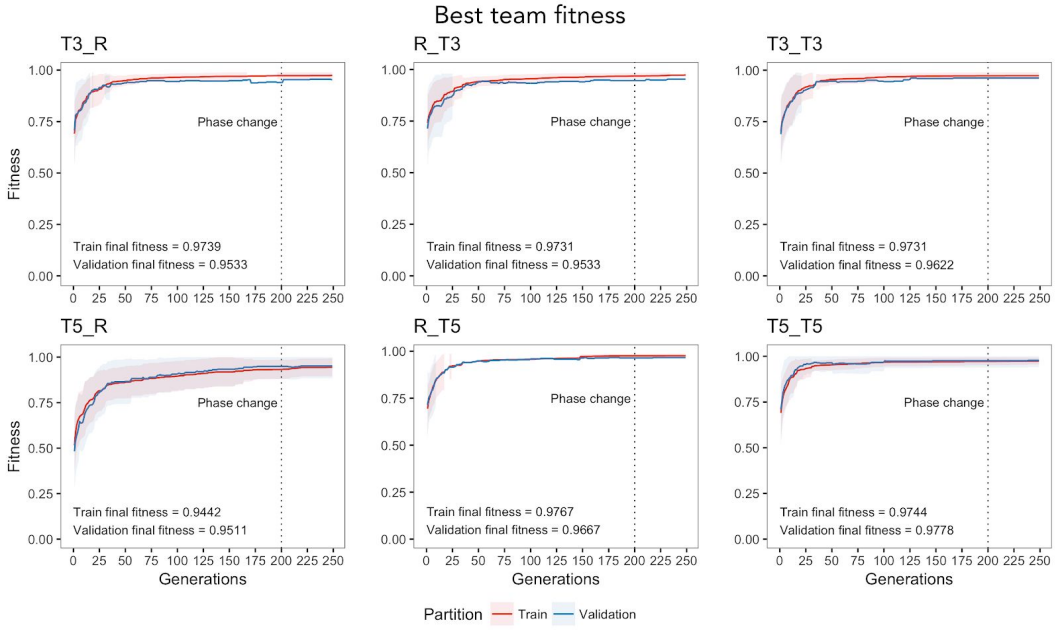

Figure 2: Mean and one standard deviation of train and validation best team fitness through the GP evolution for each selection method.

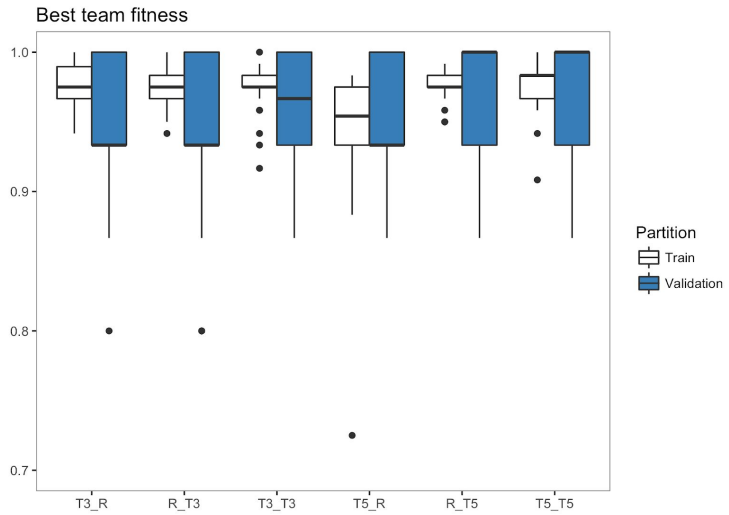

Figure 3: Train and validation best team final fitnesses for each selection method.

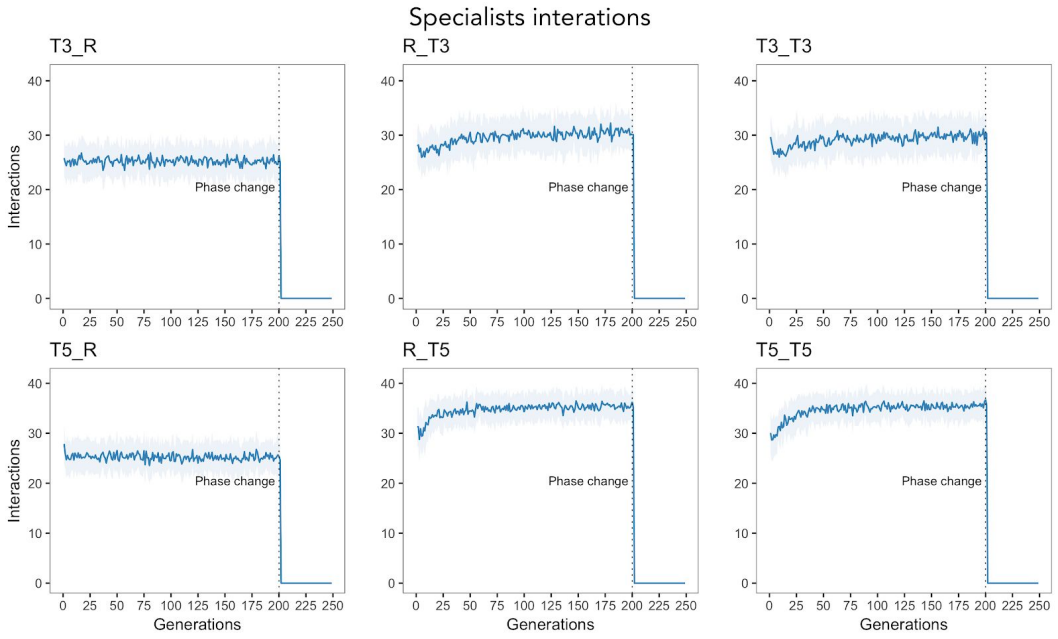

Figure 4: Specialists interactions mean and one standard deviation through GP evolution for each specialists selection method.

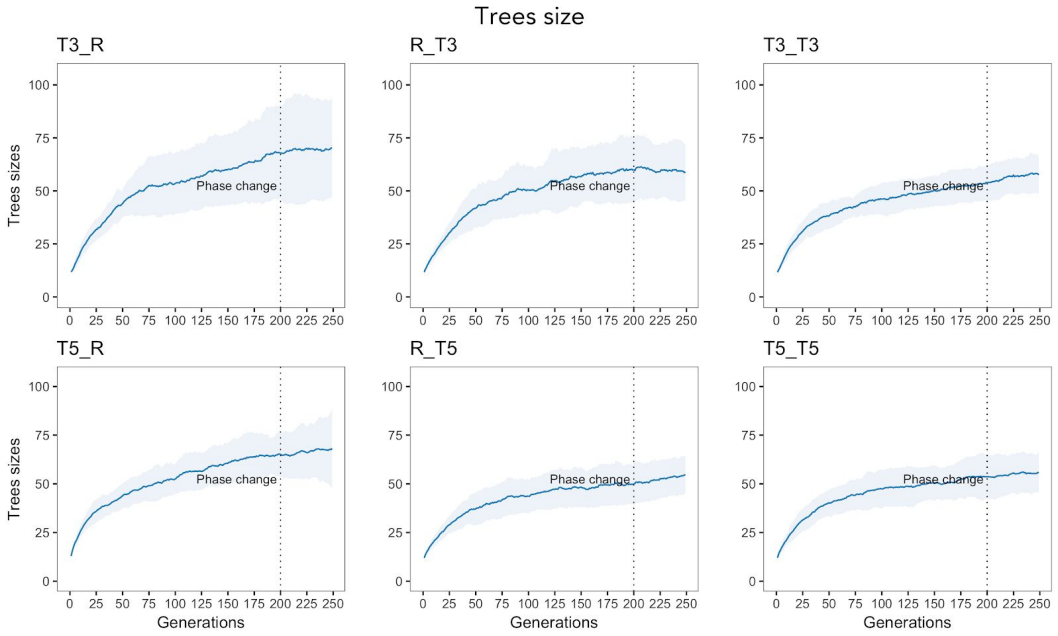

Figure 5: Trees sizes mean and one standard deviation through GP evolution for each specialists selection method.

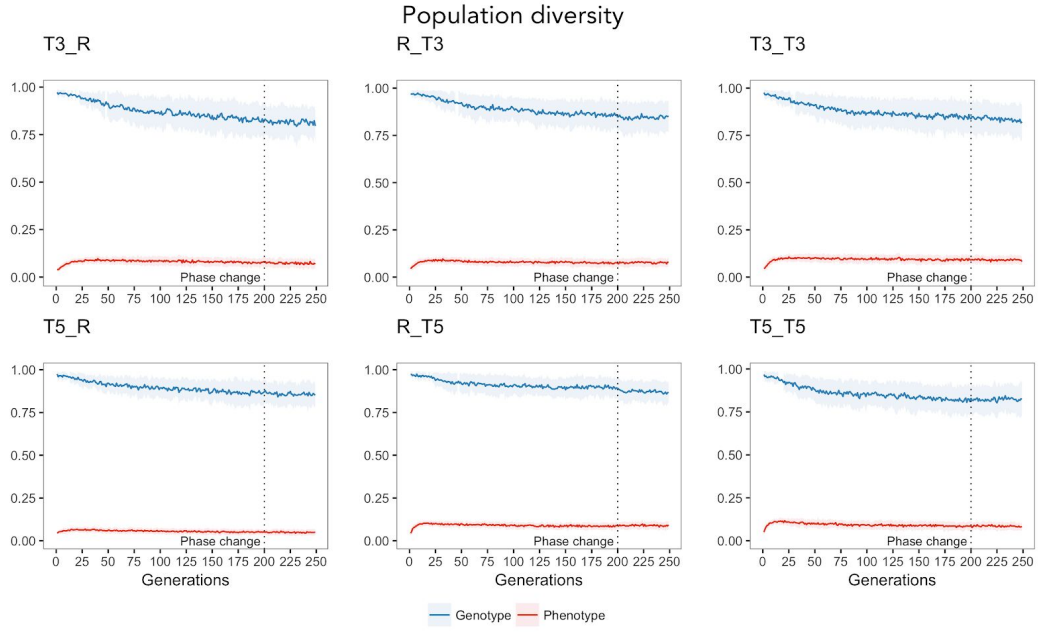

Figure 6: Genotype and phenotype diversities mean and one standard deviation through GP evolution for each selection method.

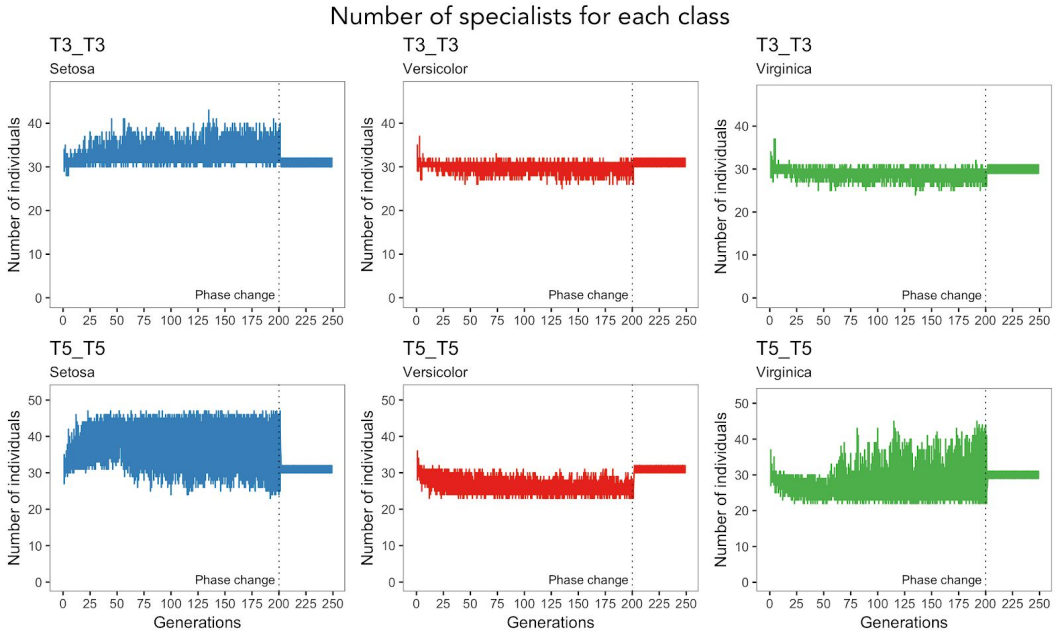

Figure 7: Number of specialised individuals in each class subpopulation through GP evolution for each specialists selection method.

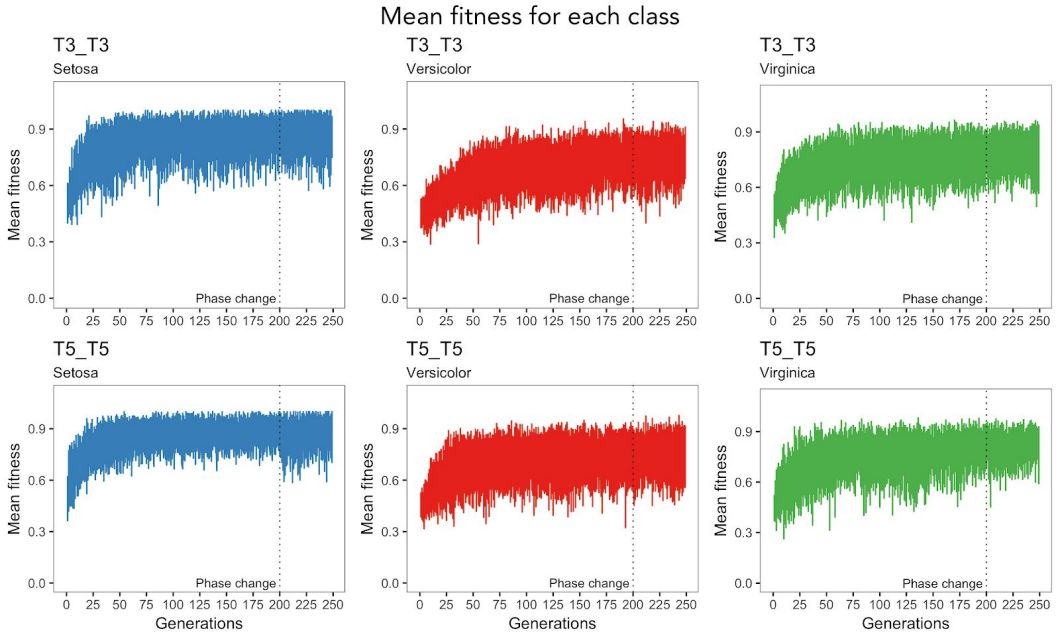

Figure 8: Mean of fitness of training partition in each class subpopulation through GP evolution for each specialists selection method.

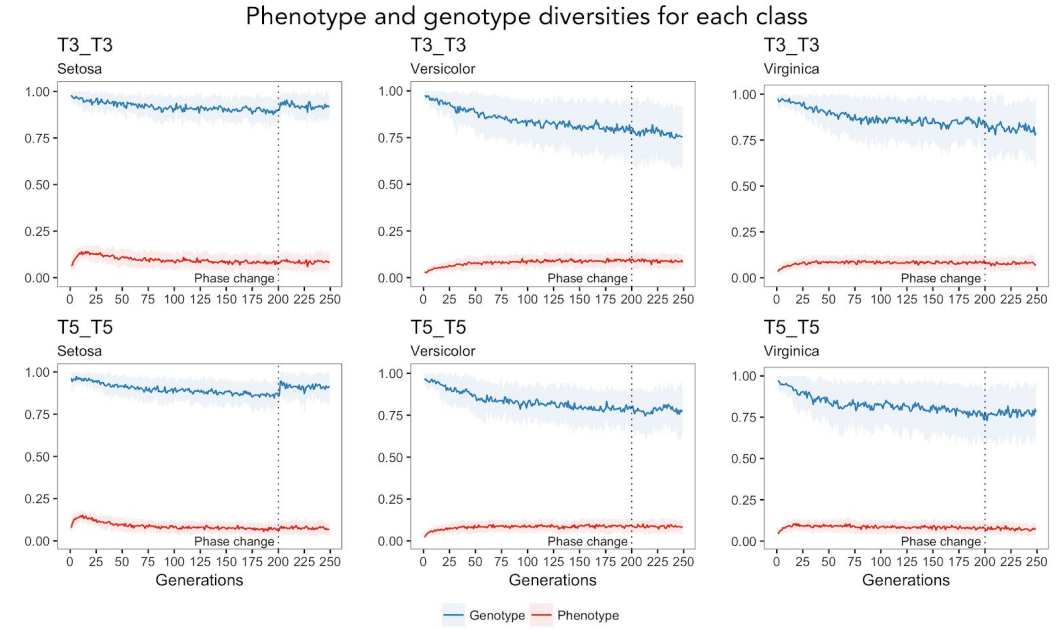

Figure 9: Genotype and phenotype diversities mean and one standard deviation for each class subpopulation through GP evolution for each selection method.

1.1.2. Specialists fitness measure

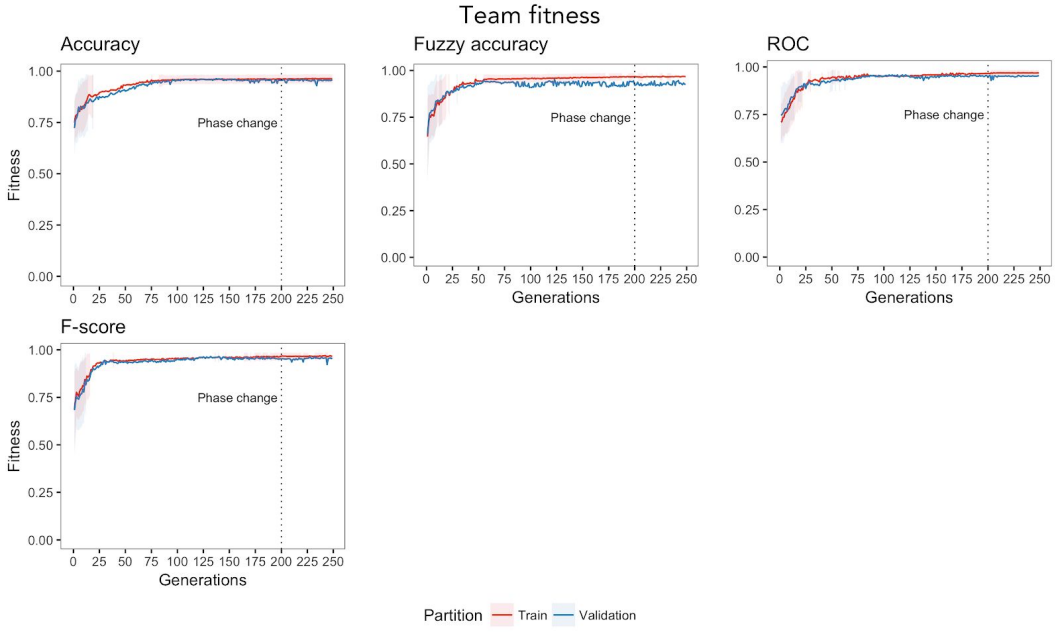

Figure 10: Mean and one standard deviation of train and validation team fitness through the GP evolution for each specialists fitness method.

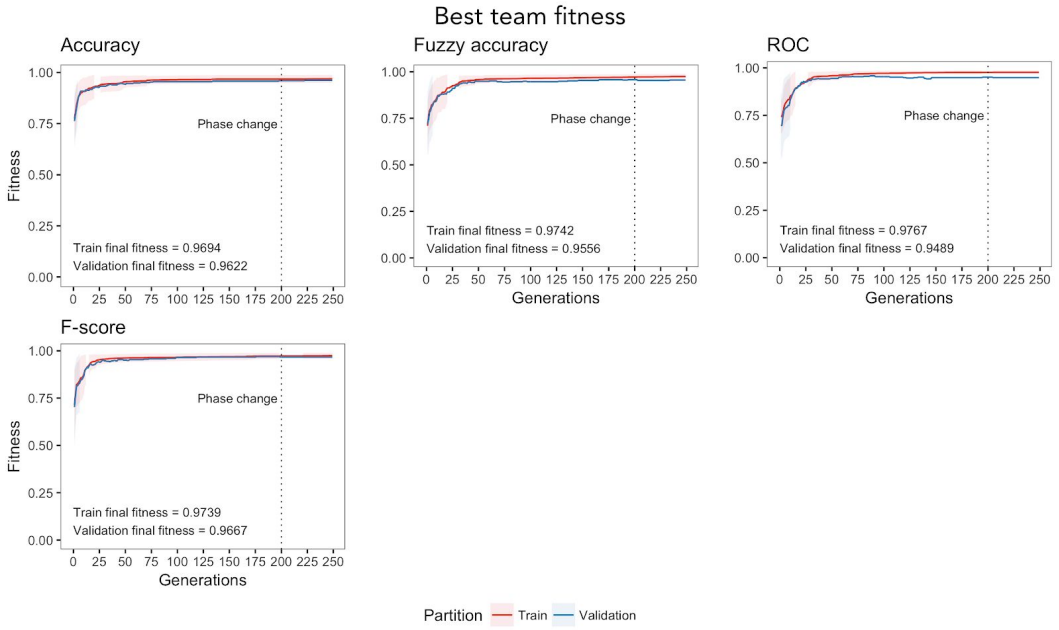

Figure 11: Mean and one standard deviation of train and validation best team fitness through the GP evolution for each specialists fitness method.

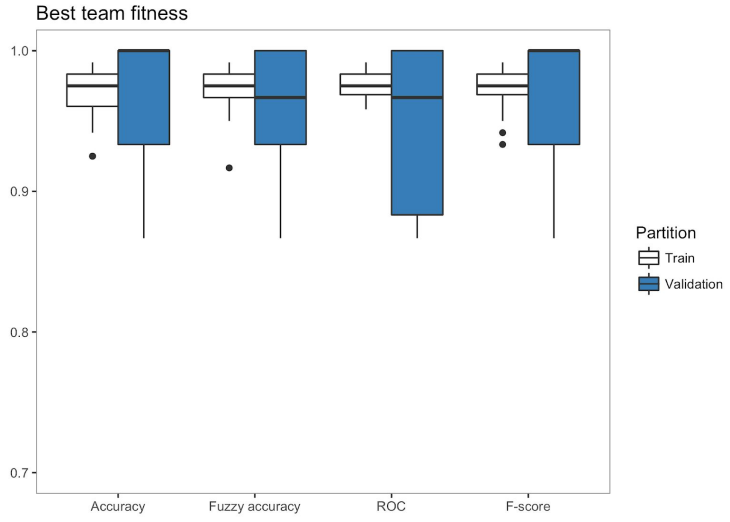

Figure 12: Train and validation best team final fitnesses for each specialists fitness method.

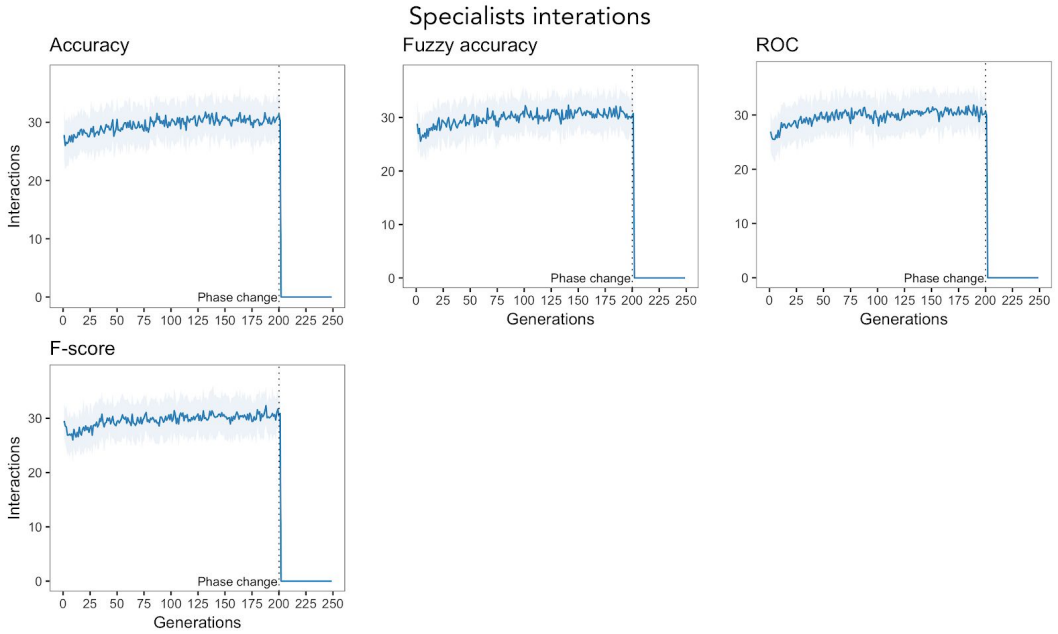

Figure 13: Specialists interactions mean and one standard deviation through GP evolution for each specialists fitness method. *method*.

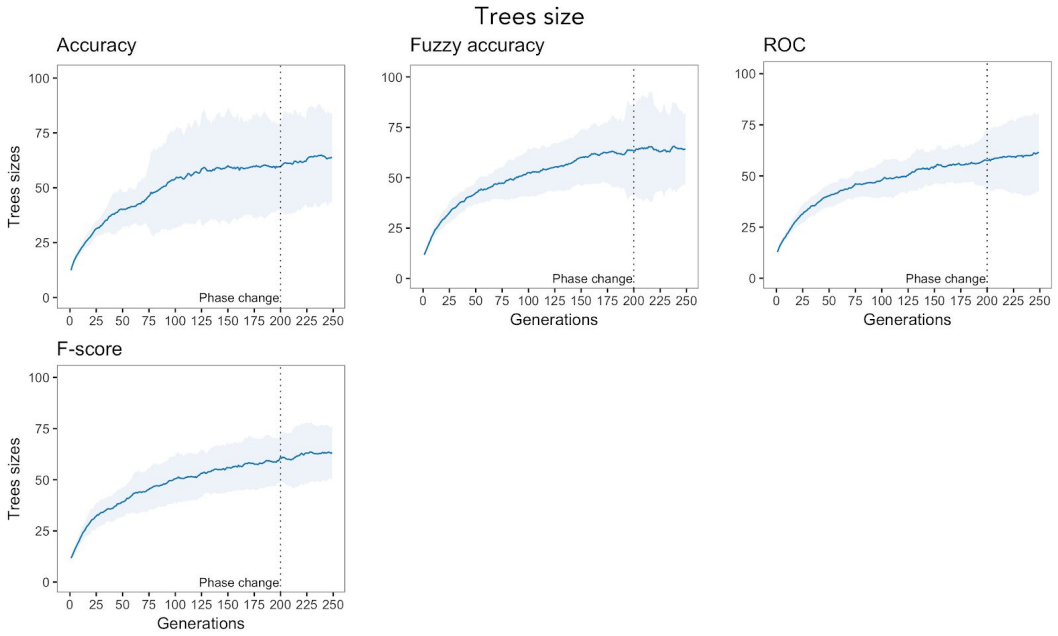

Figure 14: Trees sizes mean and one standard deviation through GP evolution for each specialists fitness method.

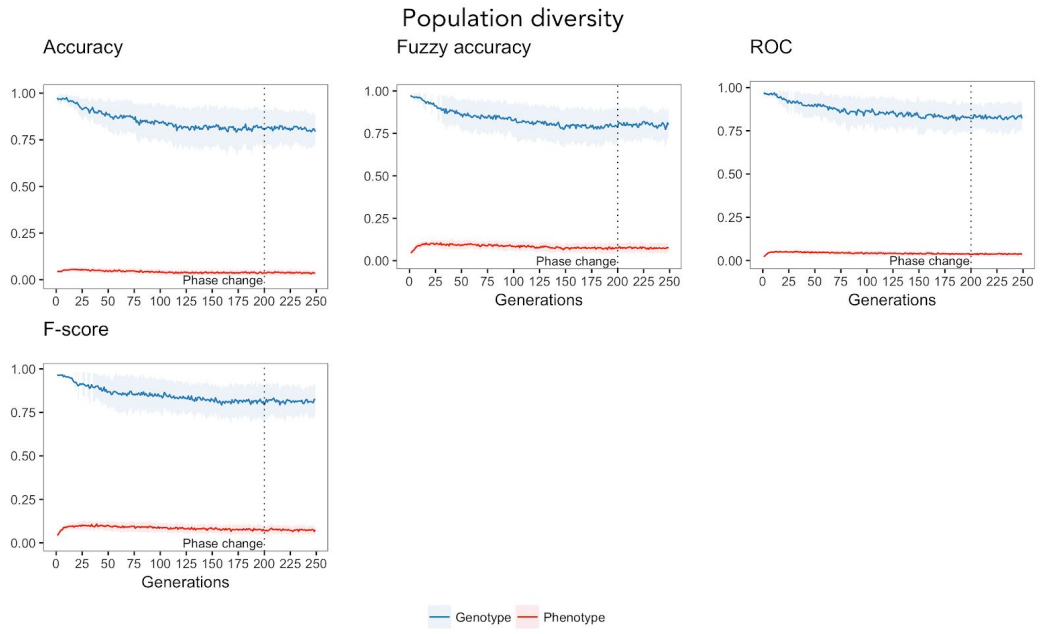

Figure 15: Trees sizes mean and one standard deviation through GP evolution for each specialists fitness method. method.

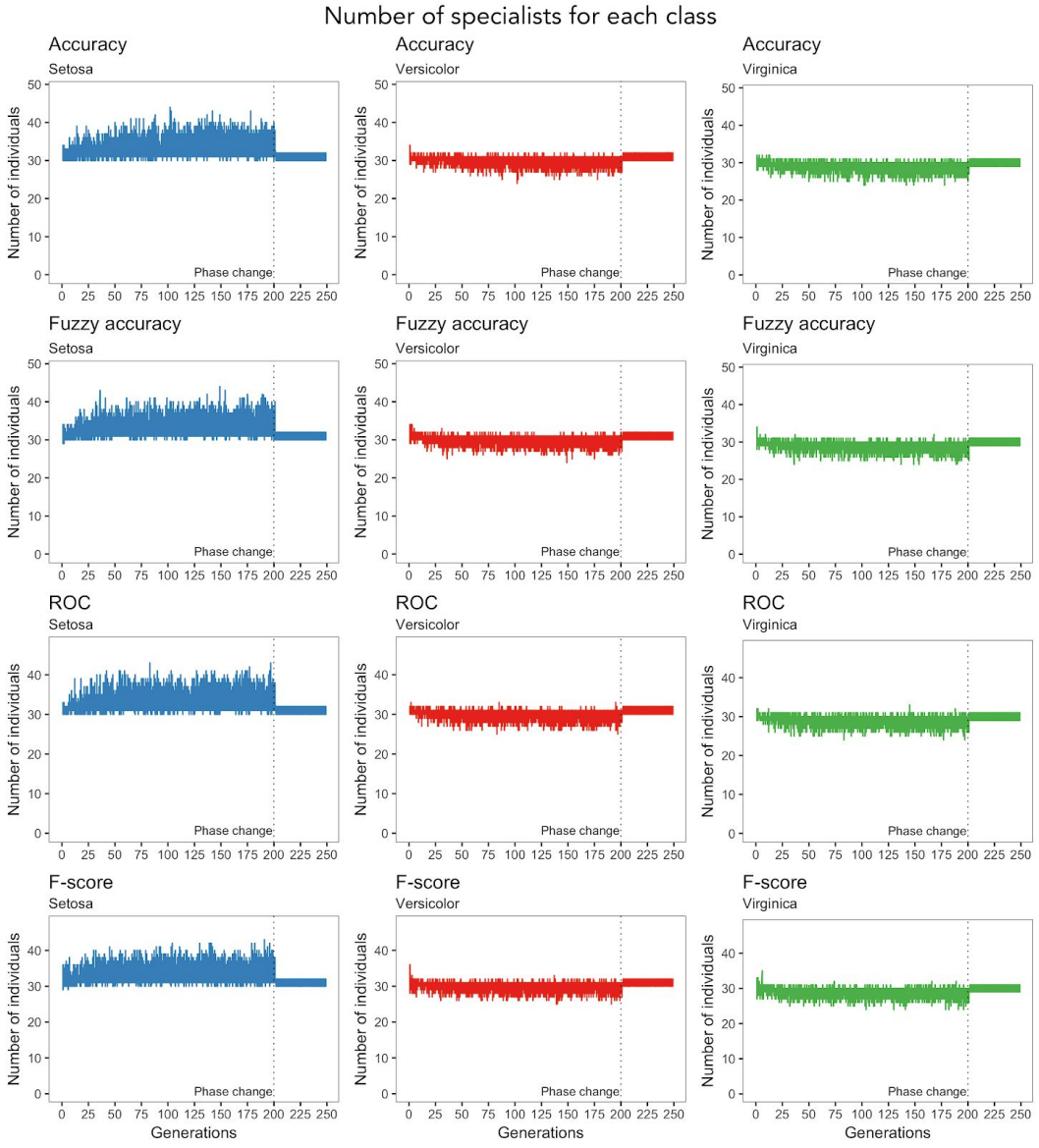

Figure 16: Number of specialised individuals in each class subpopulation through GP evolution for each specialists fitness method.

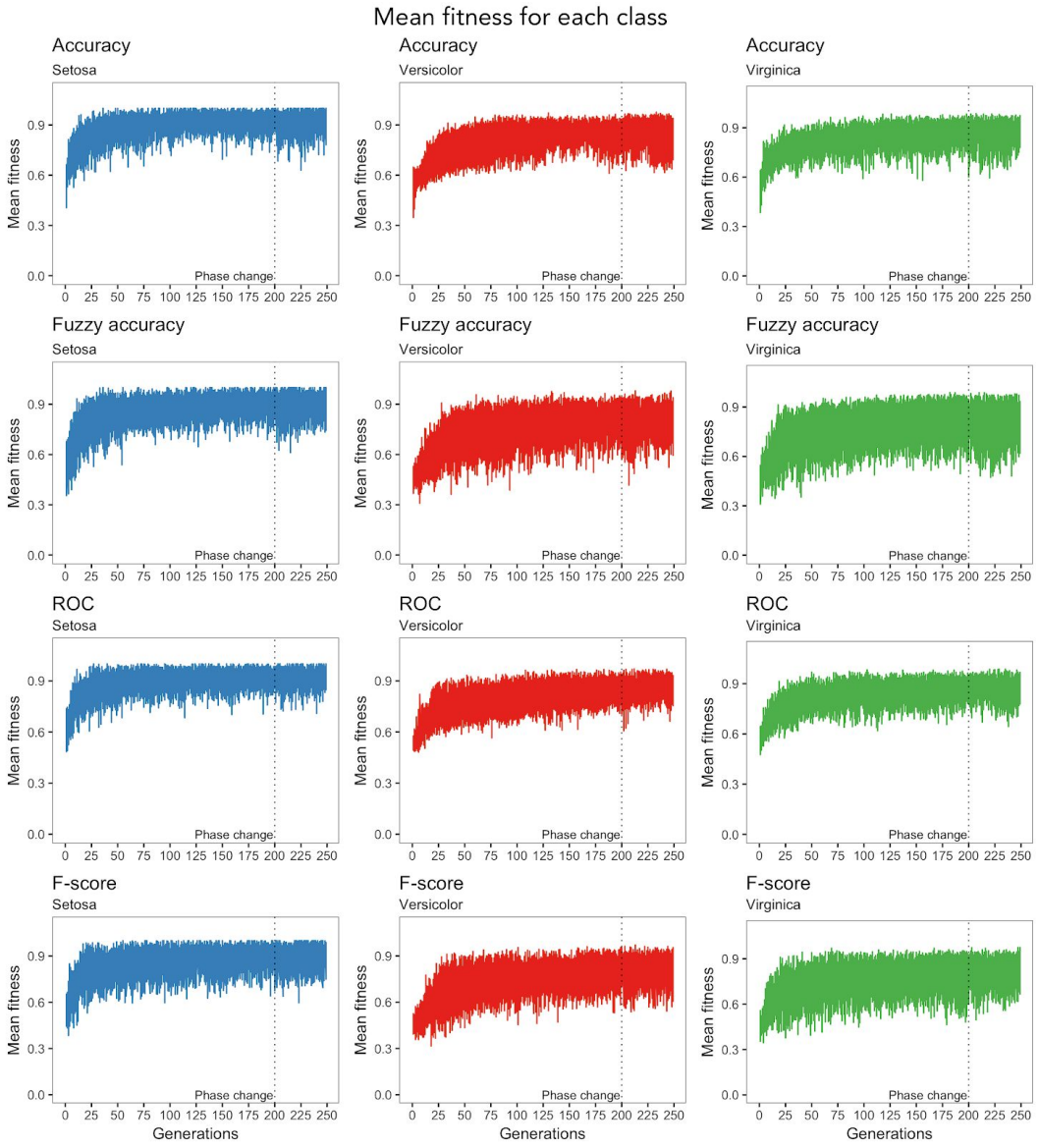

Figure 17: Mean of fitness of training partition in each class subpopulation through GP evolution for each specialists fitness method.

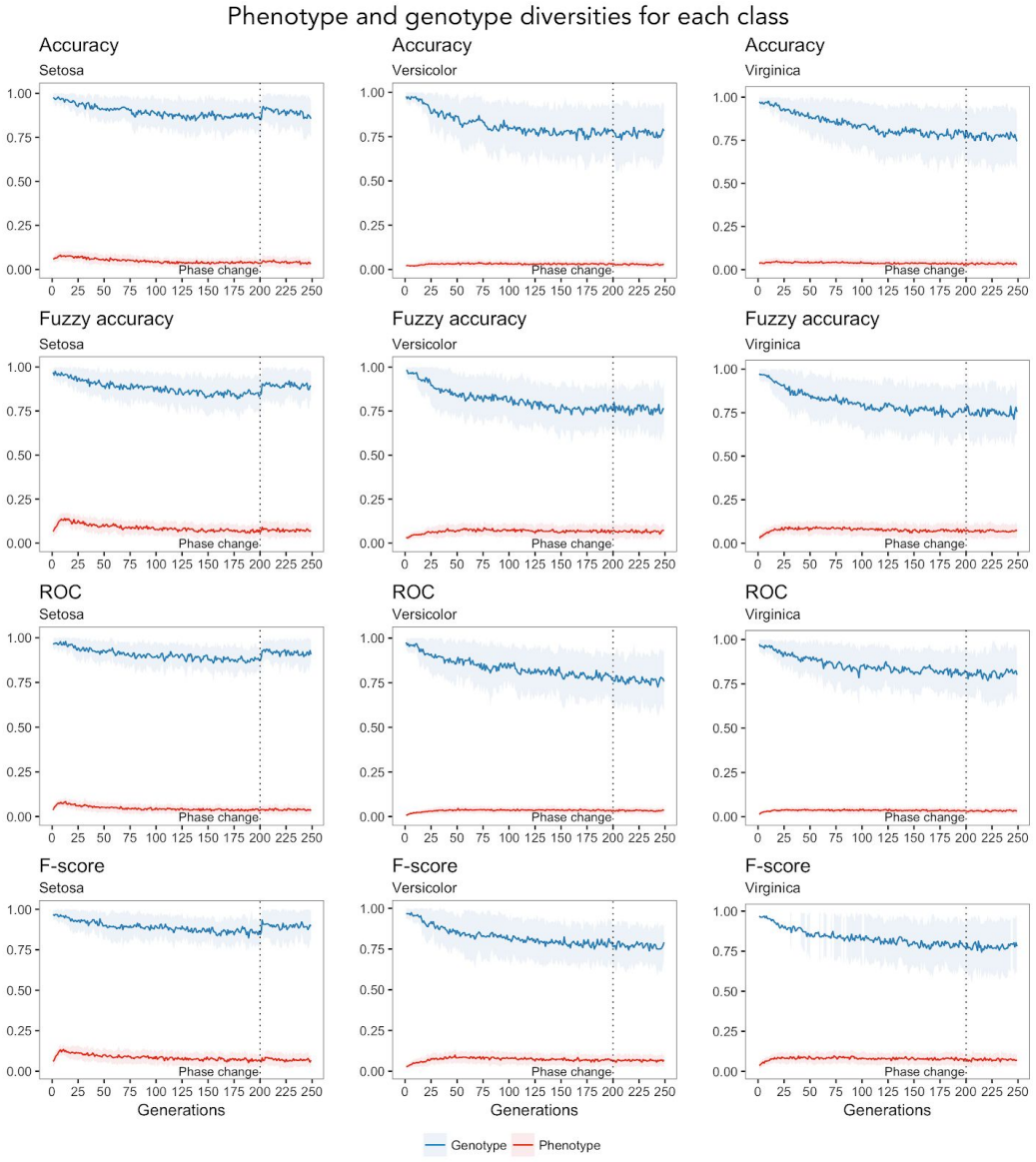

Figure 18: Genotype and phenotype diversities mean and one standard deviation for each class subpopulation through GP evolution for each specialists fitness method.

1.1.3. Phase change

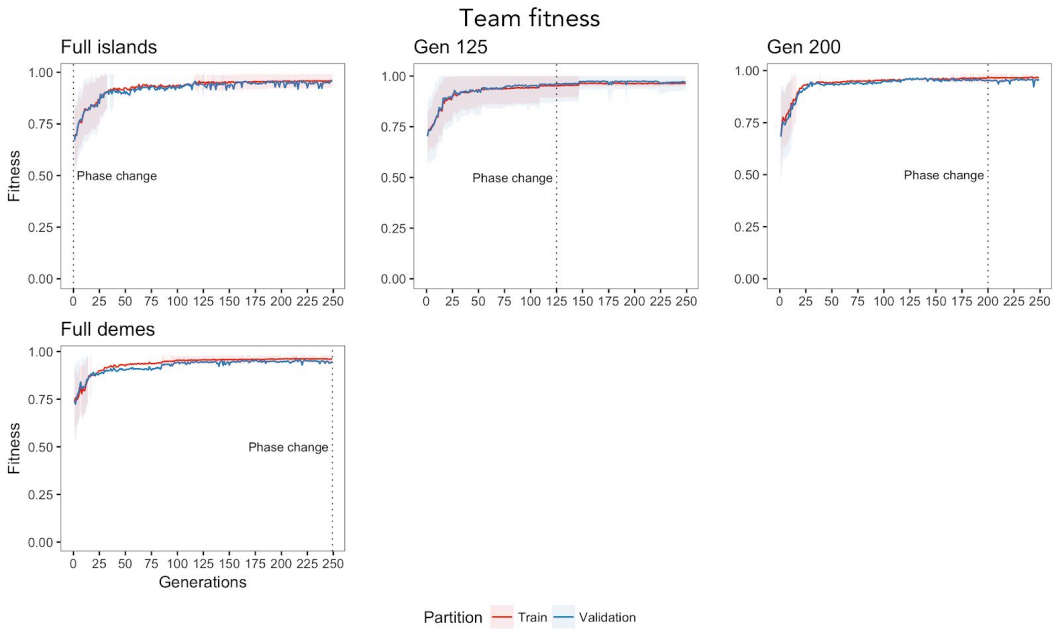

Figure 19: Mean and one standard deviation of train and validation team fitness through the GP evolution for islands-demes phase changing in different generations.

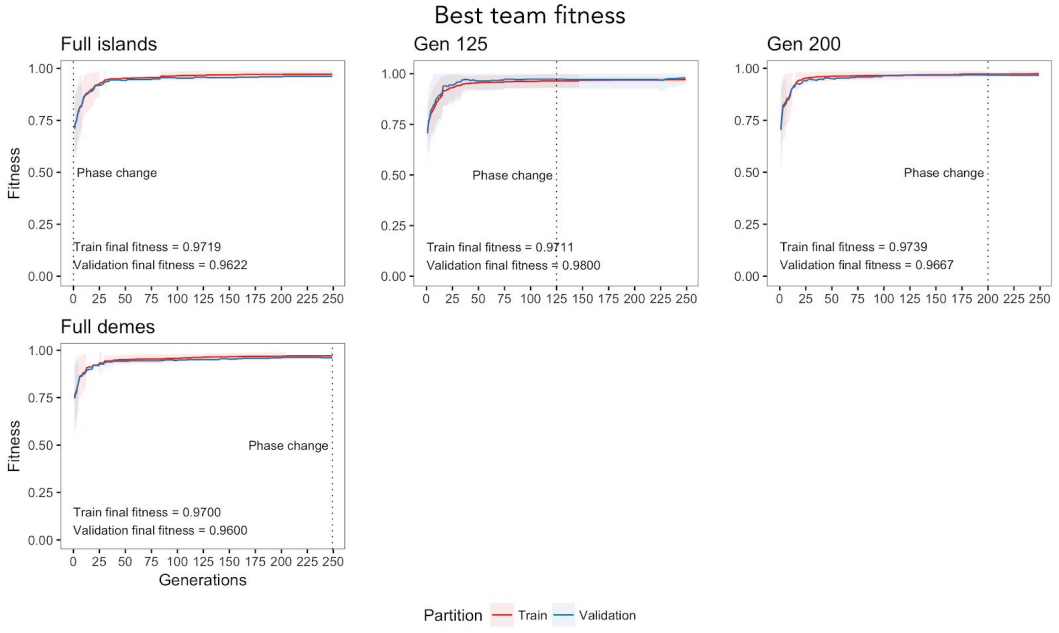

Figure 20: Mean and one standard deviation of train and validation best team fitness through the GP evolution for islands-demes phase changing in different generations.

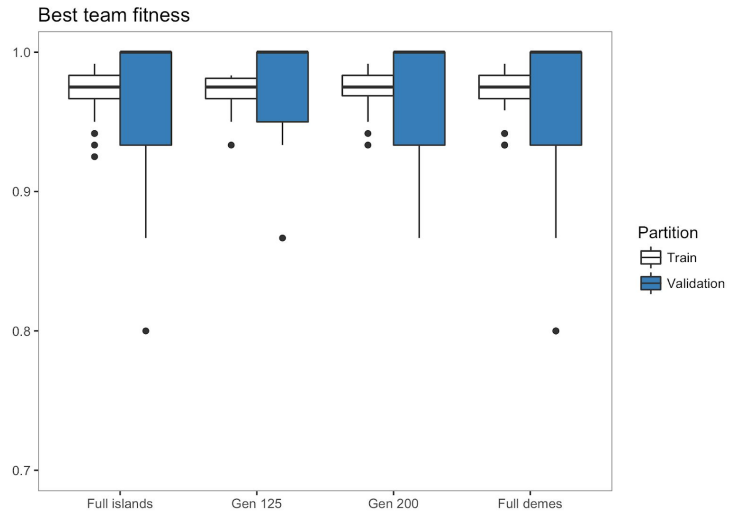

Figure 21: Train and validation best team final fitnesses for islands-demes phase changing in different generations.

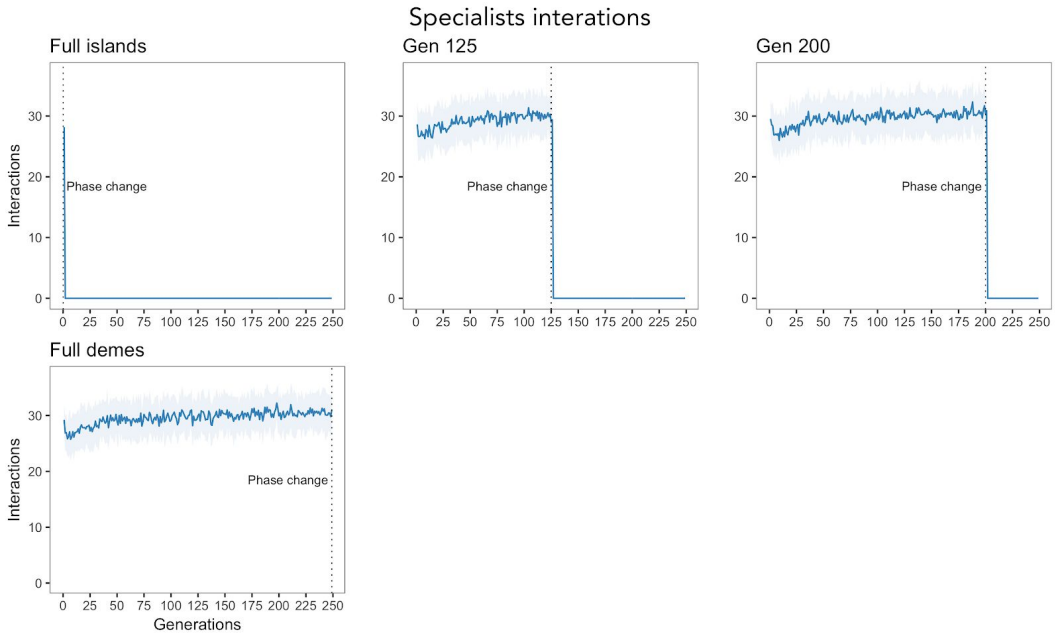

Figure 22: Specialists interactions mean and one standard deviation through GP evolution for islands-demes phase changing in different generations.

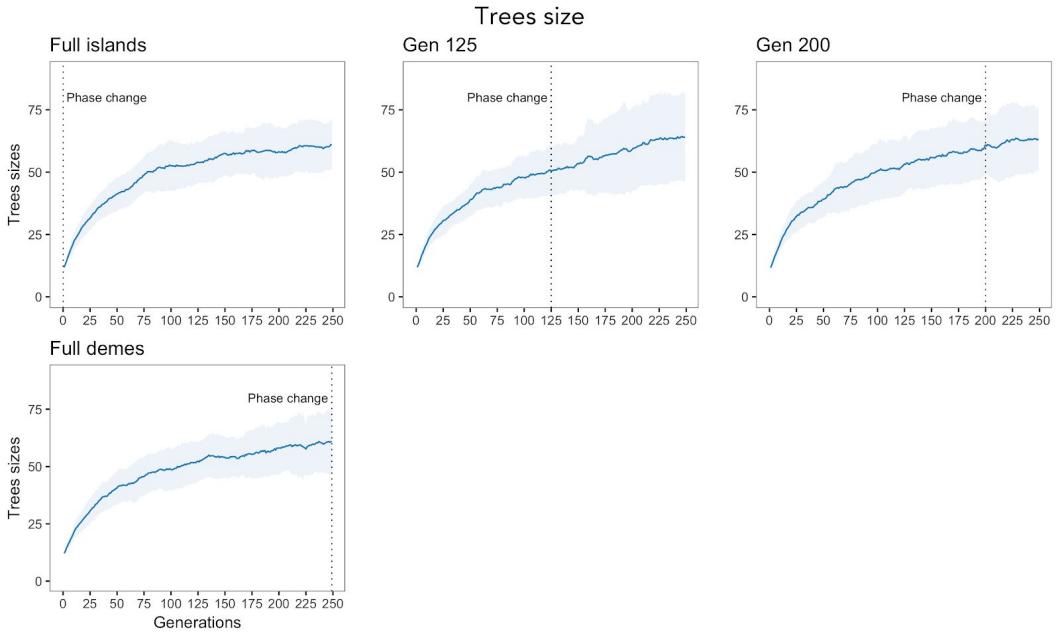

Figure 23: Trees sizes mean and one standard deviation through GP evolution for islands-demes phase changing in different generations.

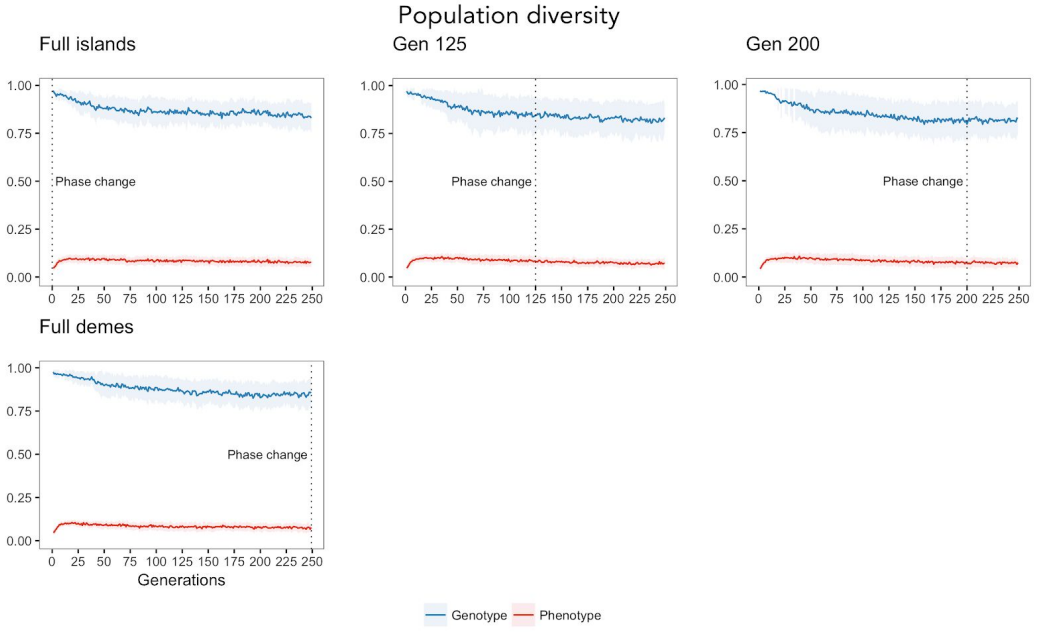

Figure 24: Trees sizes mean and one standard deviation through GP evolution for islands-demes phase changing in different generations.

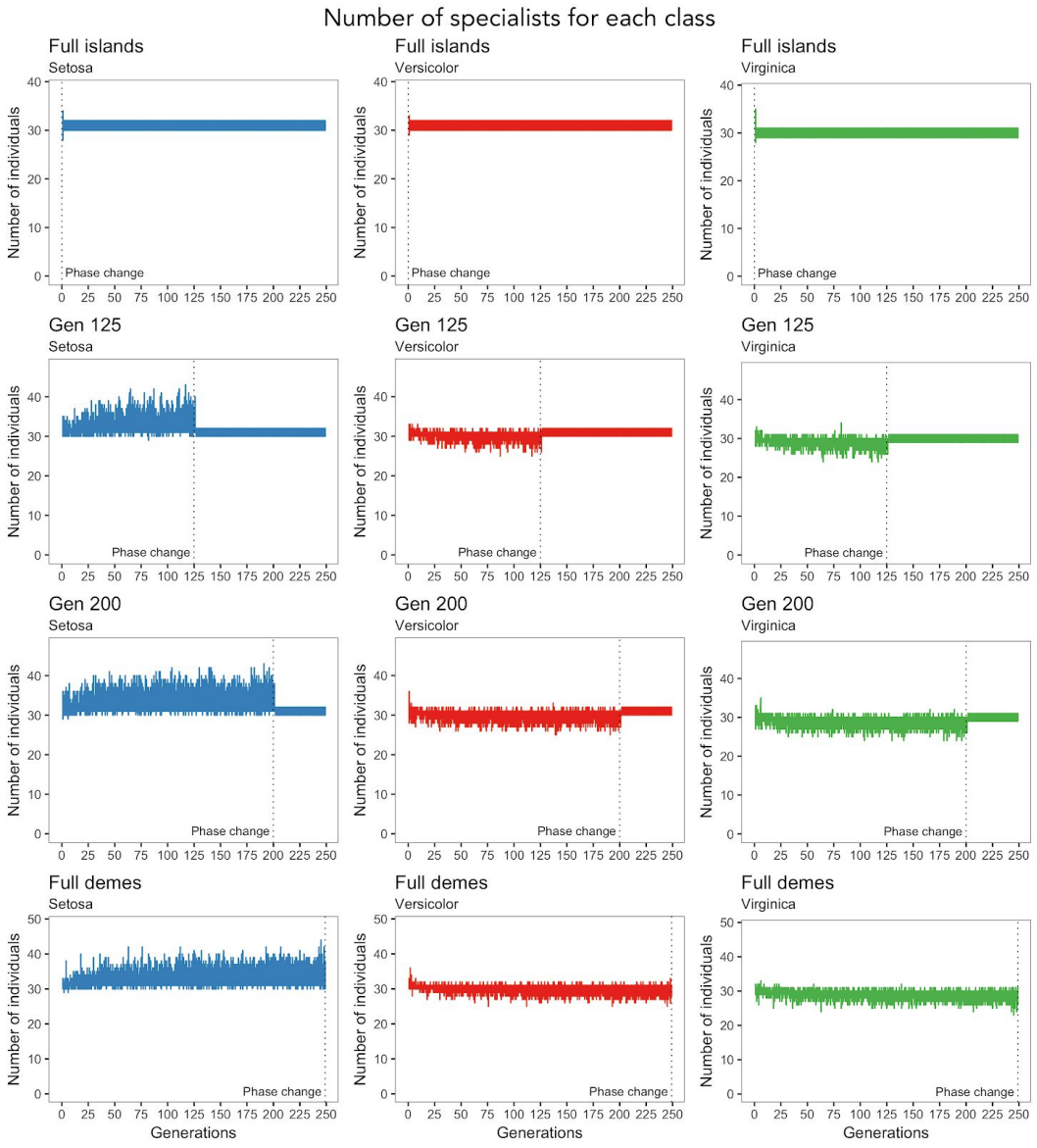

Figure 25: Number of specialised individuals in each class subpopulation through GP evolution for islands-demes phase changing in different generations.

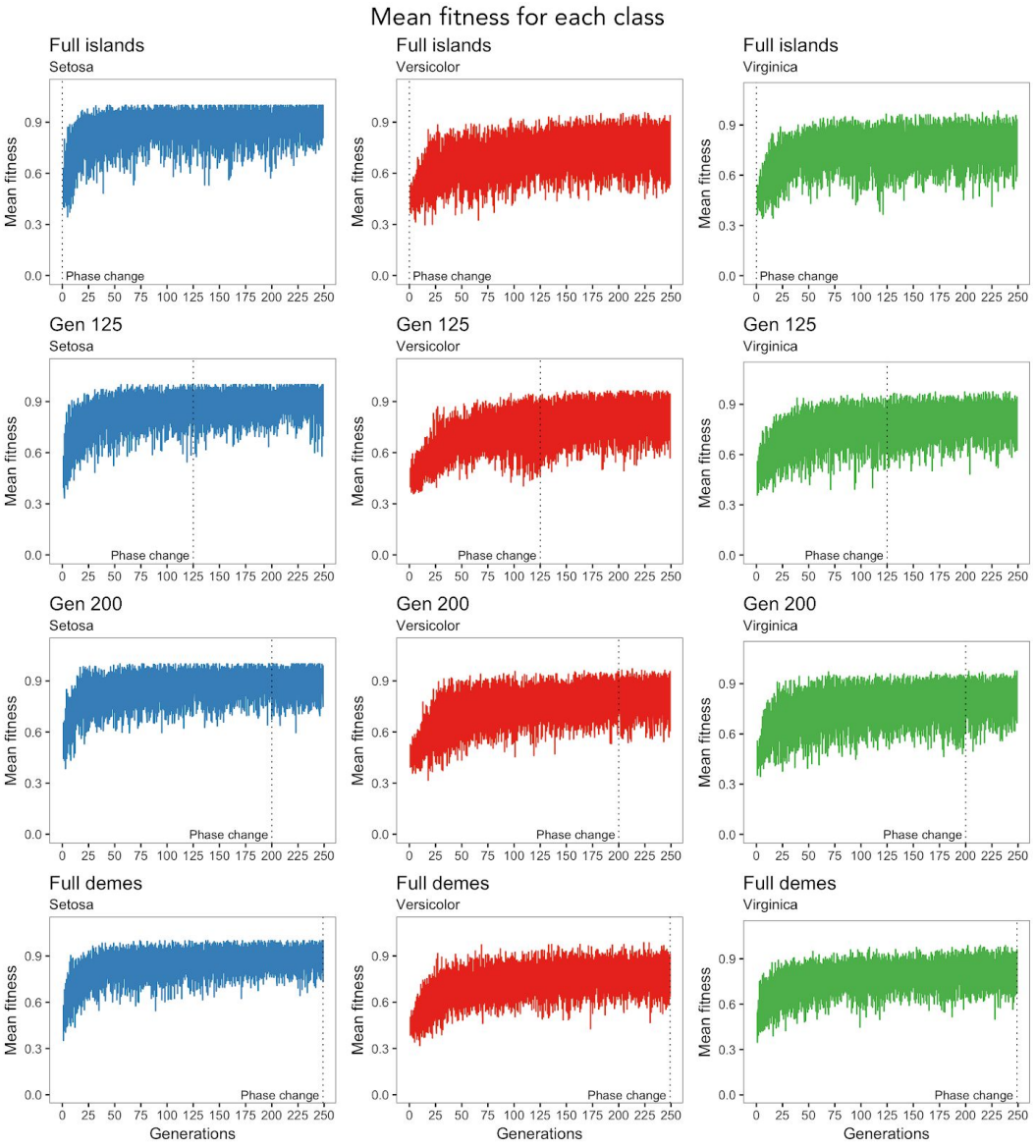

Figure 26: Mean of fitness of training partition in each class subpopulation through GP evolution for islands-demes phase changing in different generations.

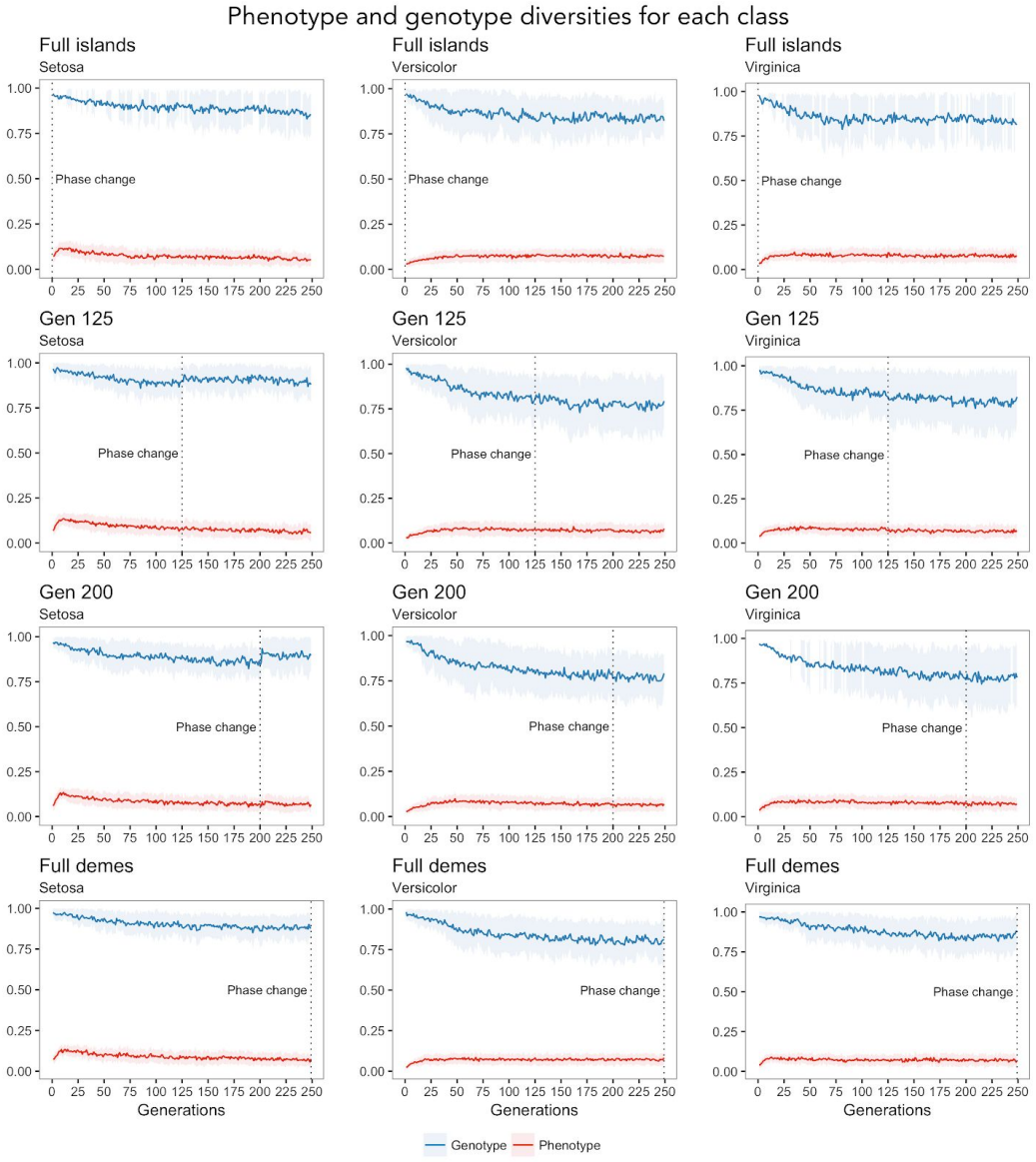

Figure 27: Genotype and phenotype diversities mean and one standard deviation for each class subpopulation through GP evolution for islands-demes phase changing in different generations.

1.1.4. Cooperation intensity rate

1.1.4.1. Initial rate

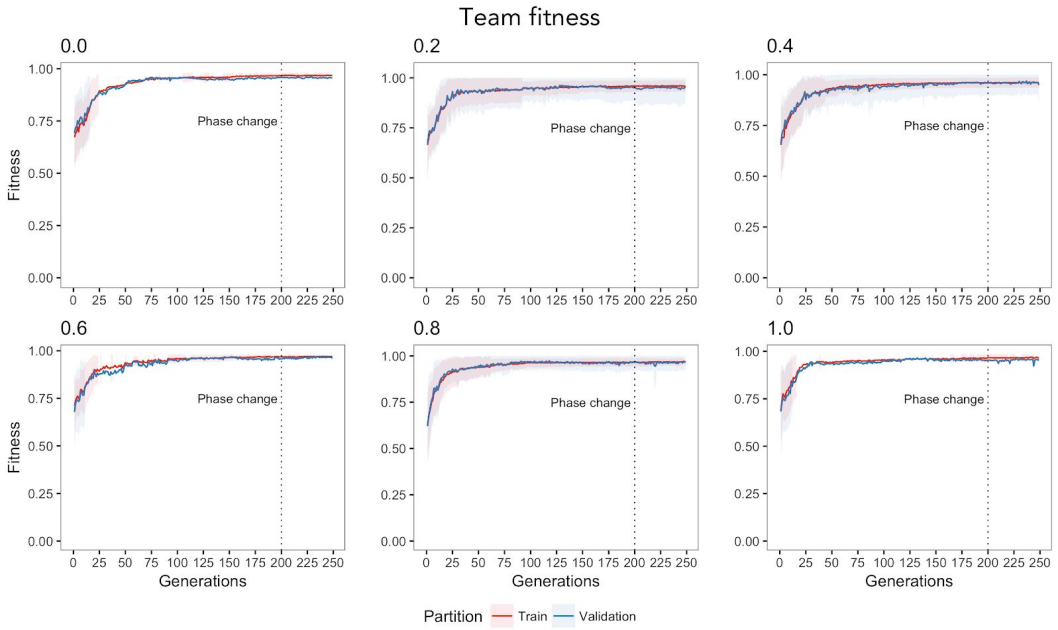

Figure 28: Mean and one standard deviation of train and validation team fitness through the GP evolution for each CIR values experiment.

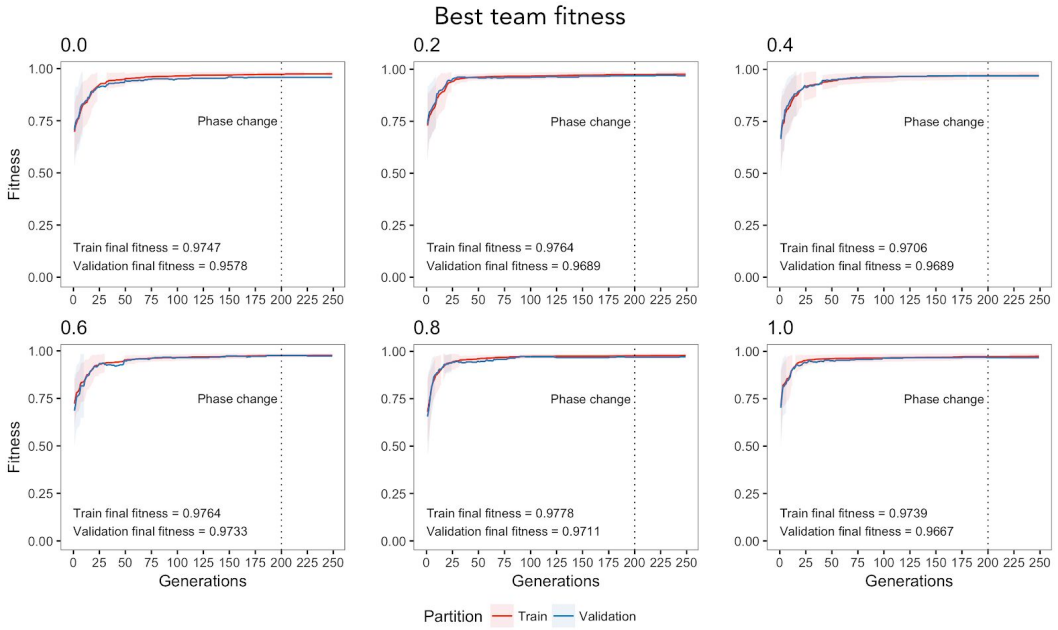

Figure 29: Mean and one standard deviation of train and validation best team fitness through the GP evolution for eachCIR values experiment.

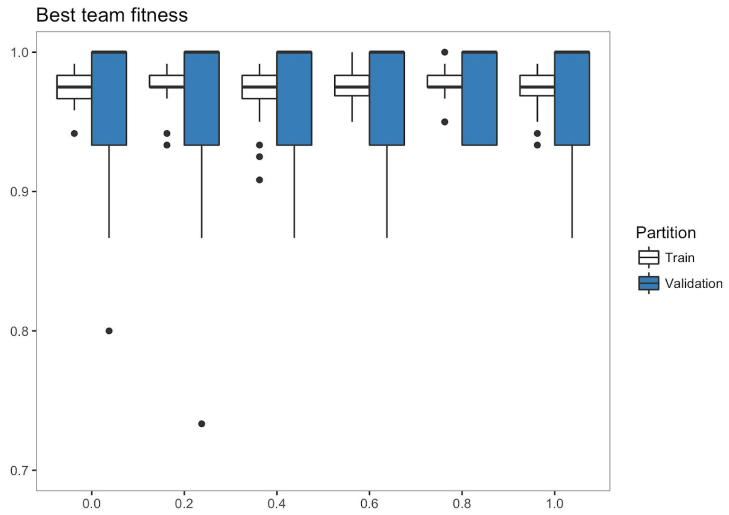

Figure 30: Train and validation best team final fitnesses for each CIR values experiment.

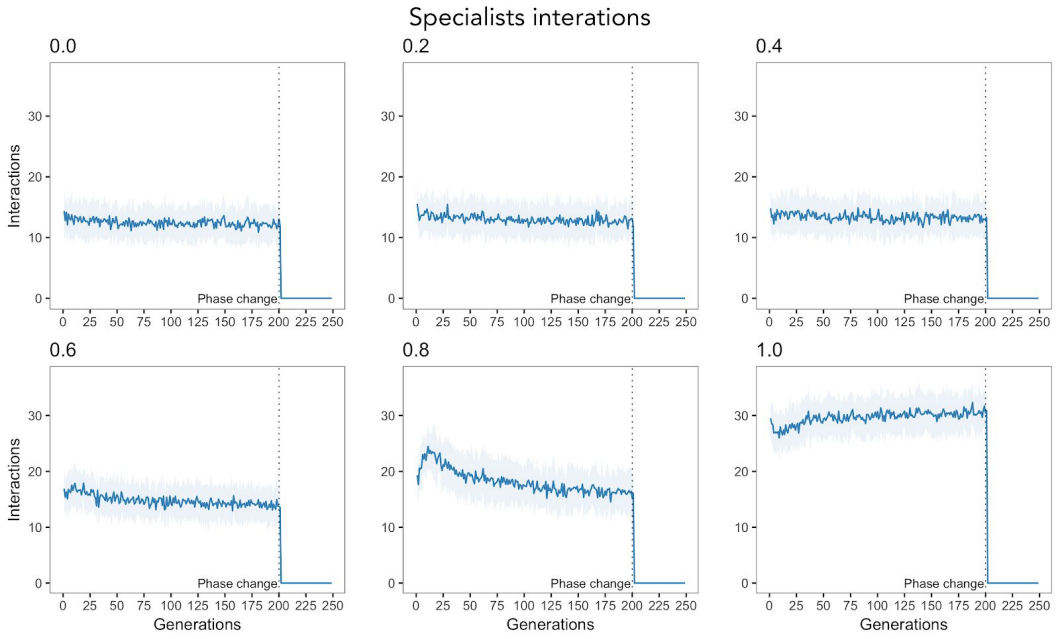

Figure 31: Specialists interactions mean and one standard deviation through GP evolution for CIR values experiment.

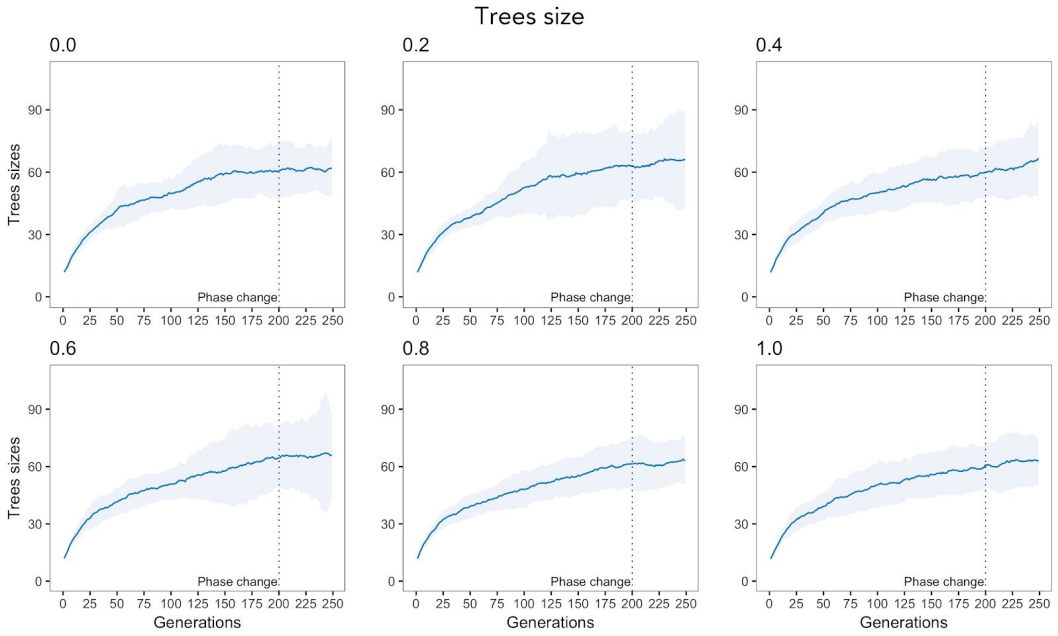

Figure 32: Trees sizes mean and one standard deviation through GP evolution for each CIR values experiment.

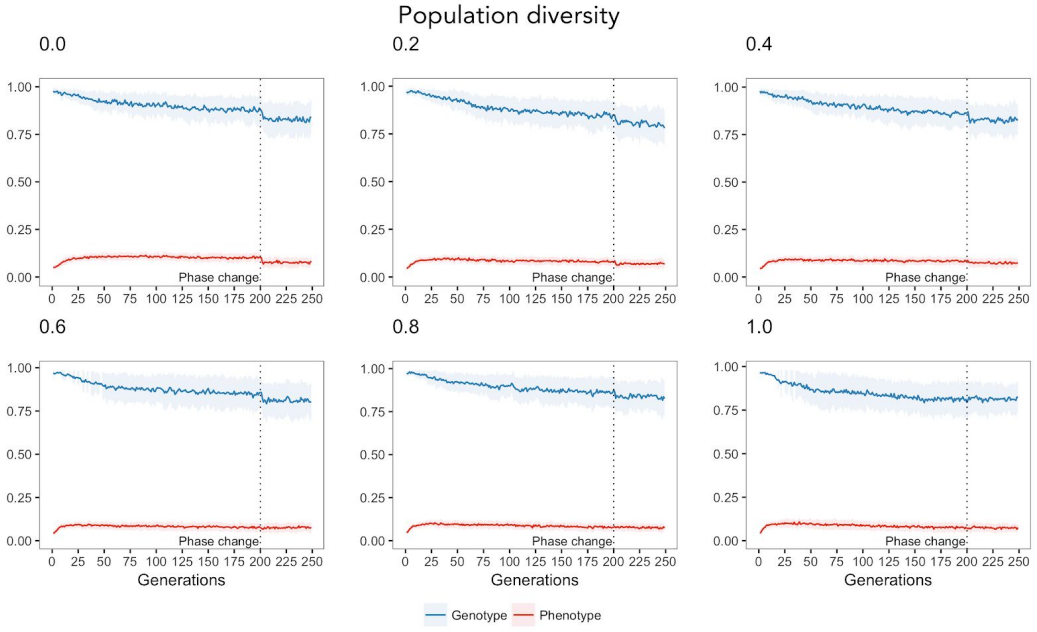

Figure 33: Trees sizes mean and one standard deviation through GP evolution for each CIR values experiment.

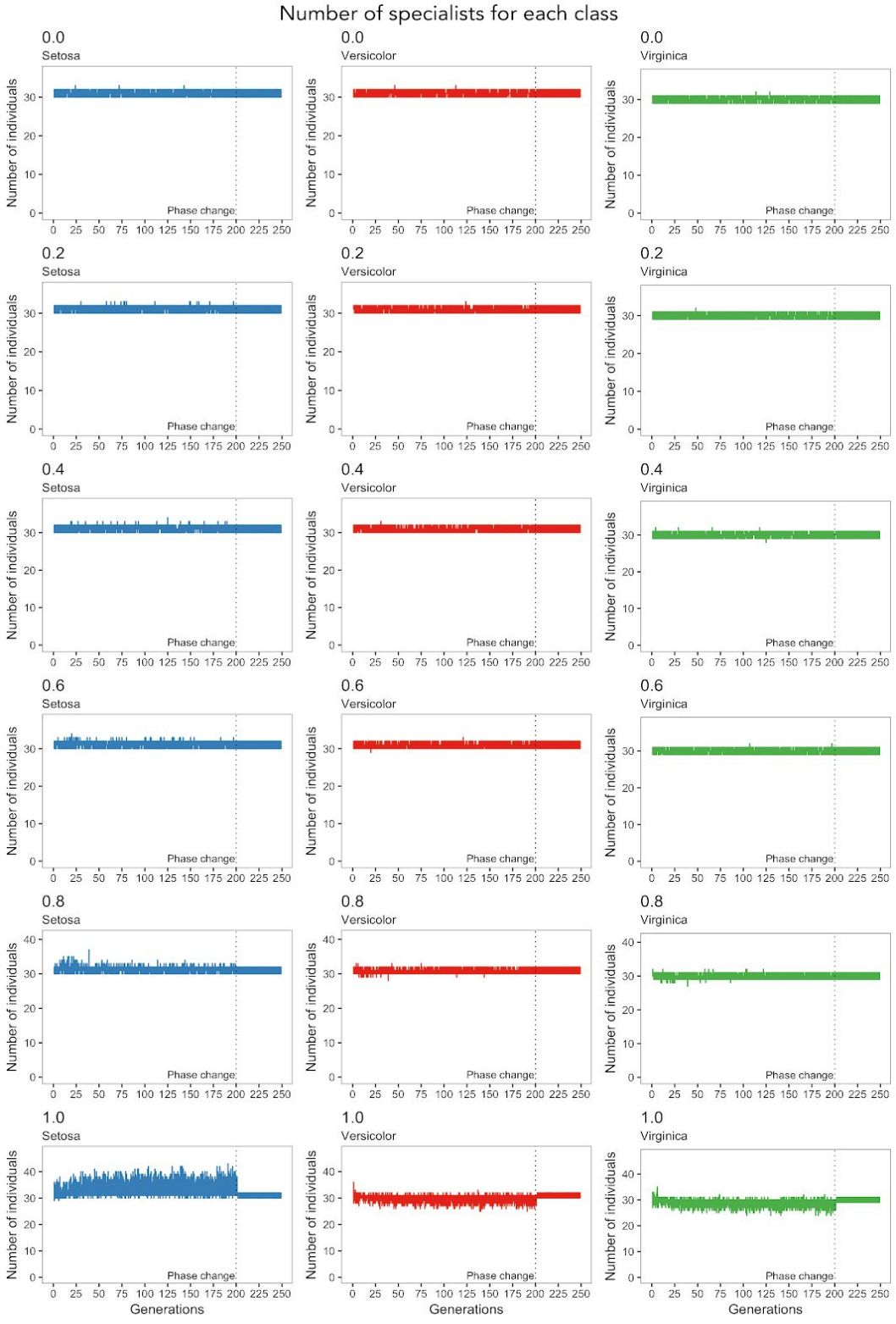

Figure 34: Number of specialised individuals in each class subpopulation through GP evolution for each CIR values experiment.

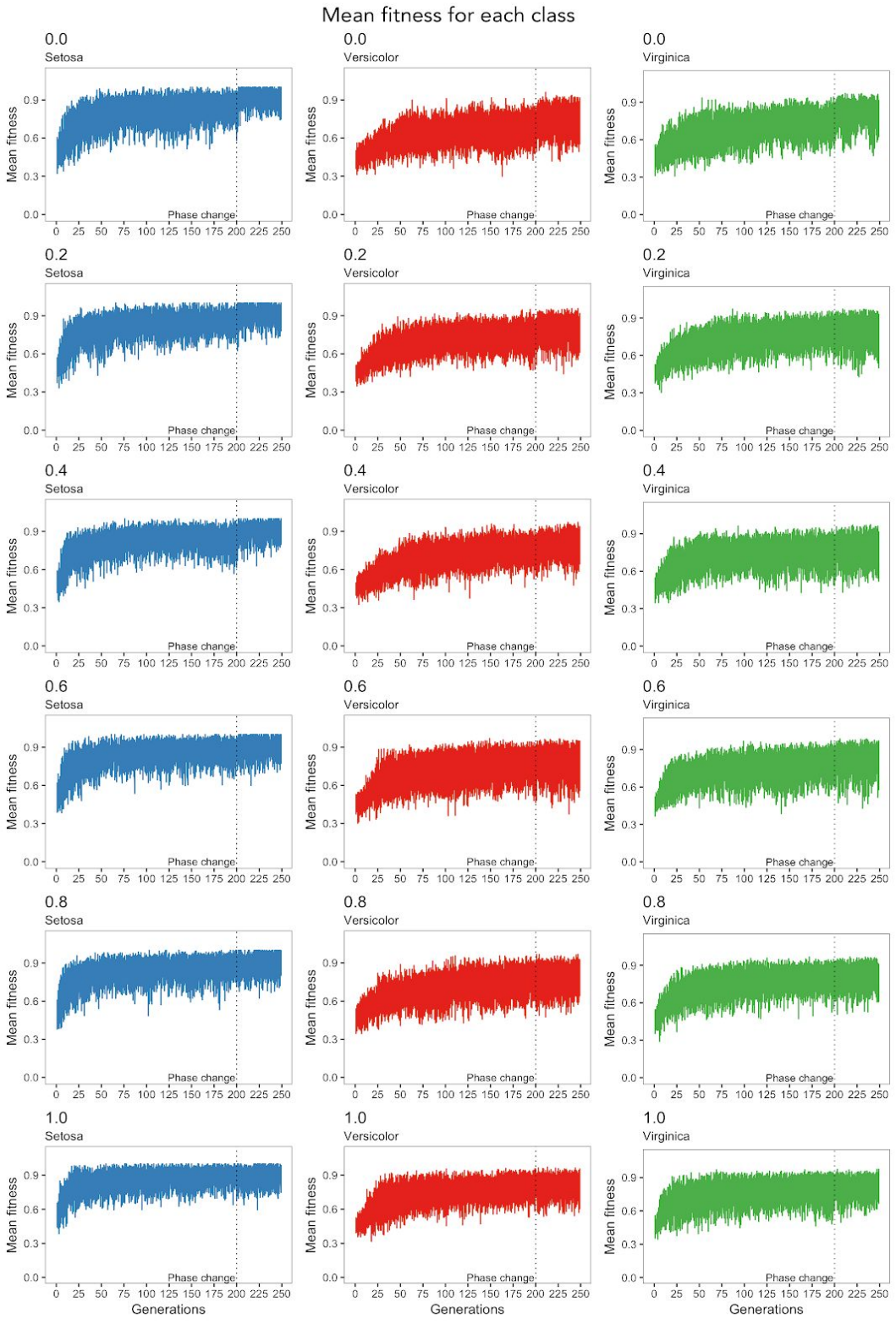

Figure 35: Mean of fitness of training partition in each class subpopulation through GP evolution for each CIR values experiment.

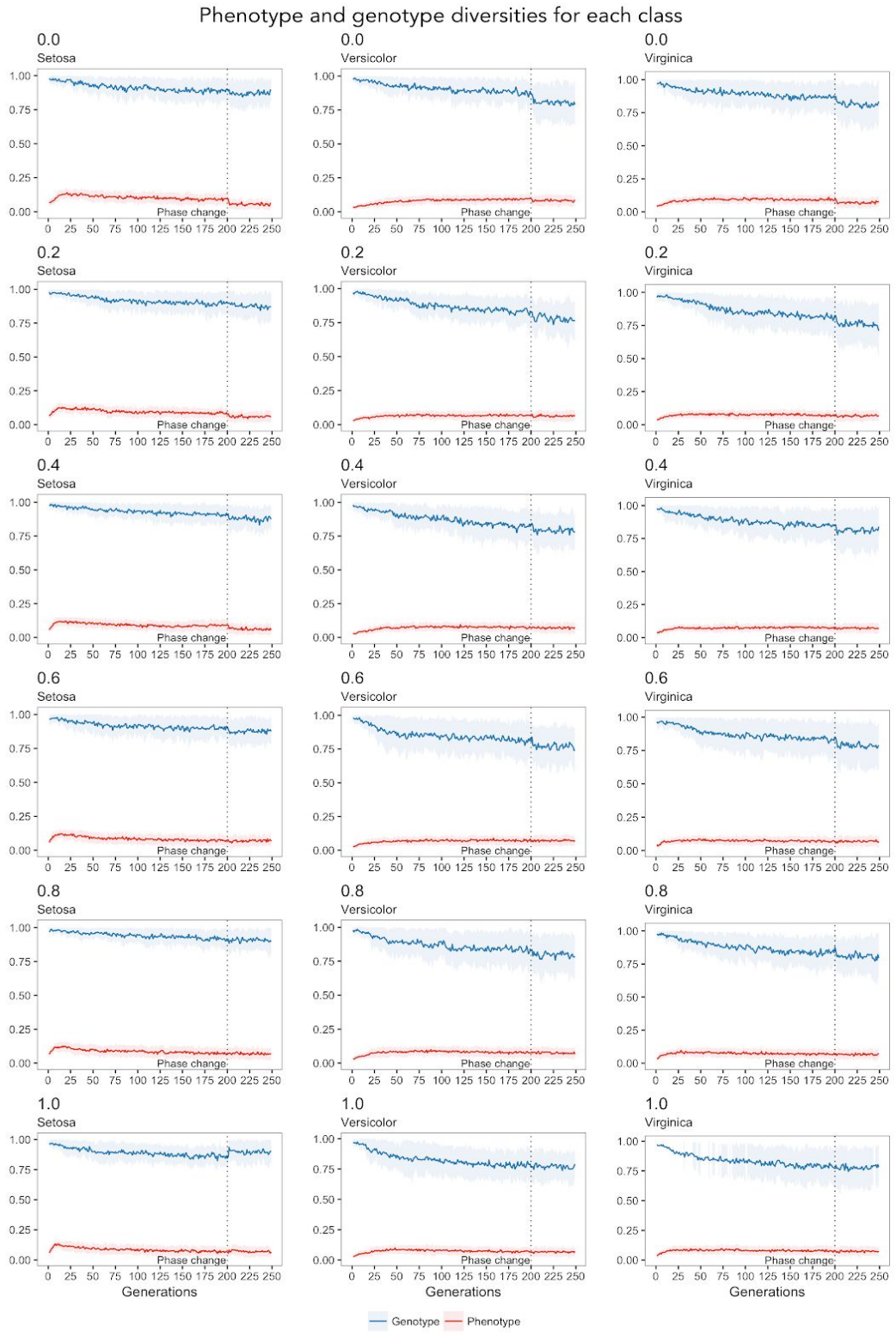

Figure 36: Genotype and phenotype diversities mean and one standard deviation for each class subpopulation through GP evolution for each CIR values experiment.

1.1.4.2. Rate decrease

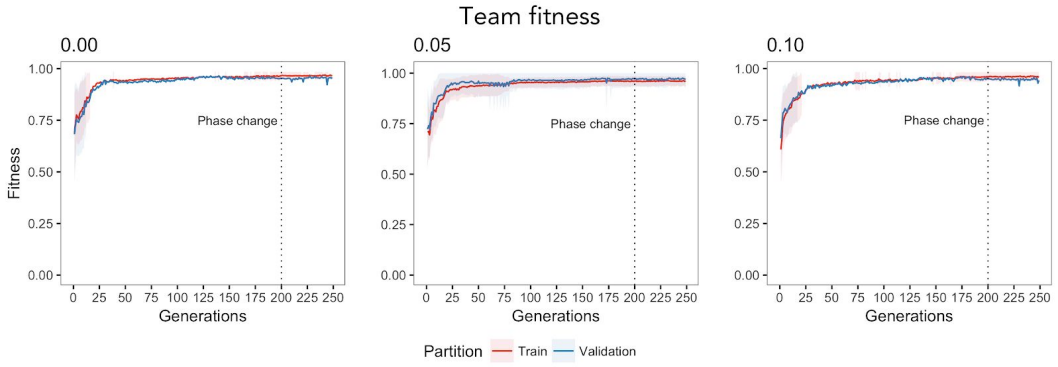

Figure 37: Mean and one standard deviation of train and validation team fitness through the GP evolution for different cooperation intensity rate decrease values.

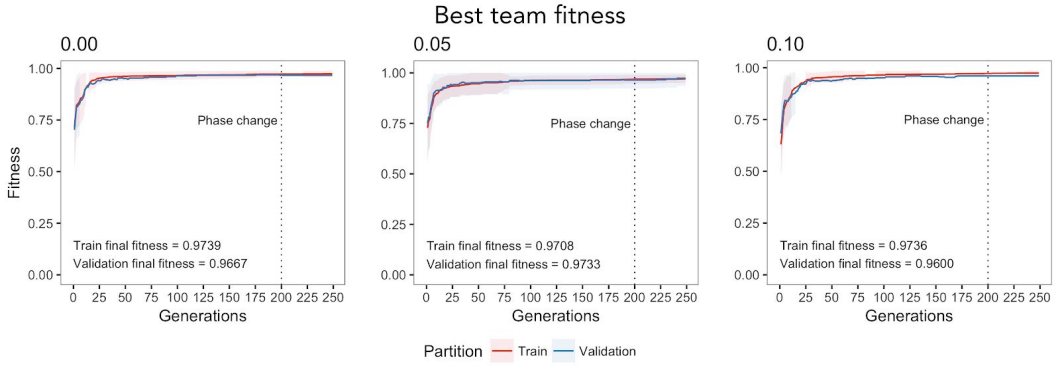

Figure 38: Mean and one standard deviation of train and validation best team fitness through the GP evolution for different cooperation intensity rate decrease values.

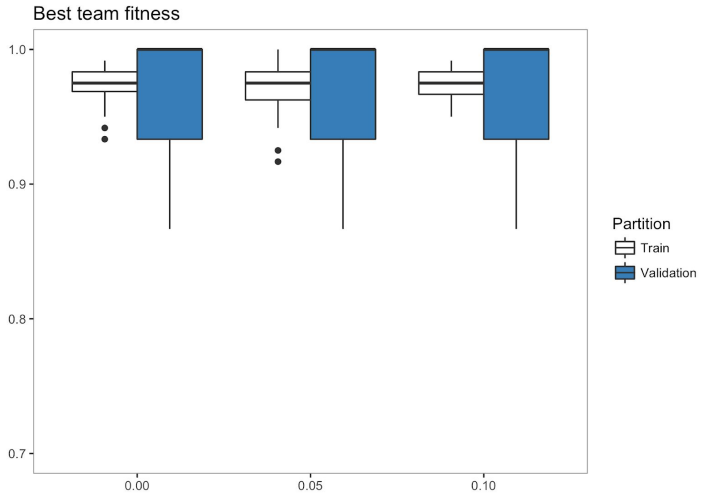

Figure 39: Train and validation best team final fitnesses for different cooperation intensity rate decrease values.

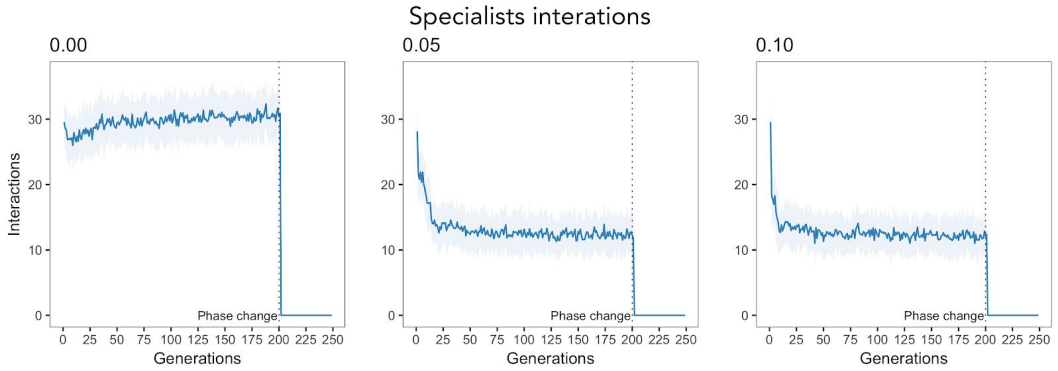

Figure 40: Specialists interactions mean and one standard deviation through GP evolution for different cooperation intensity rate decrease values.

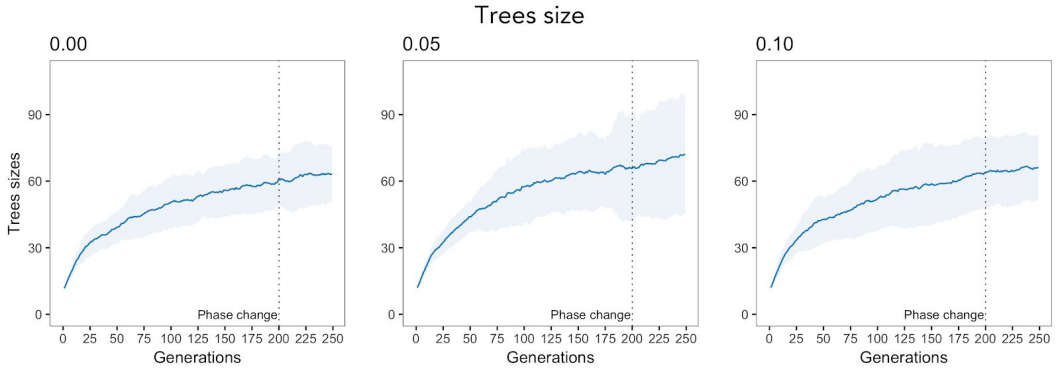

Figure 41: Trees sizes mean and one standard deviation through GP evolution for different cooperation intensity rate decrease values.

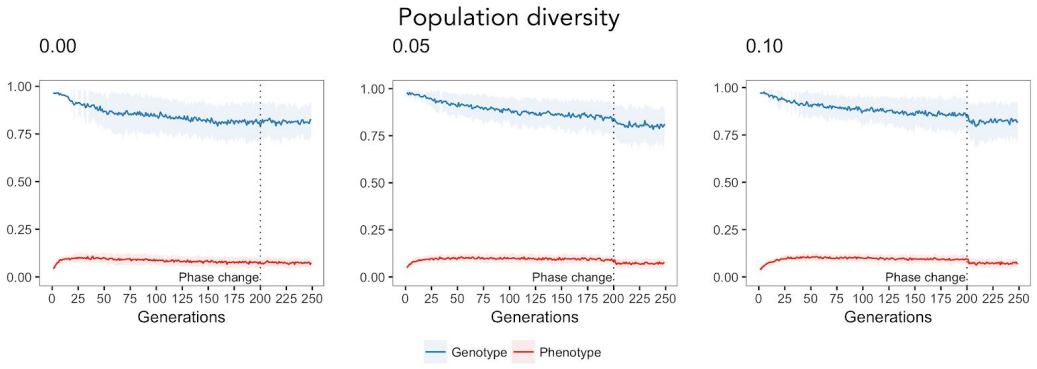

Figure 42: Trees sizes mean and one standard deviation through GP evolution for different cooperation intensity rate decrease values.

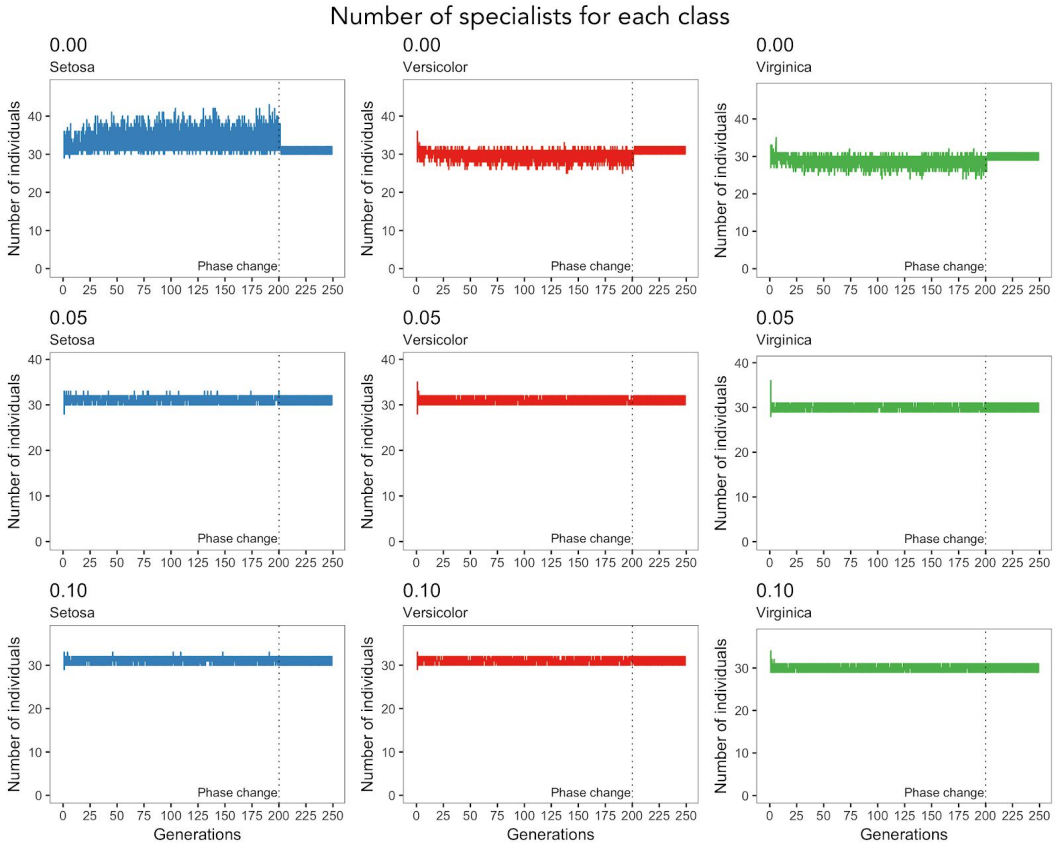

Figure 43: Number of specialised individuals in each class subpopulation through GP evolution for each different cooperation intensity rate decrease values.

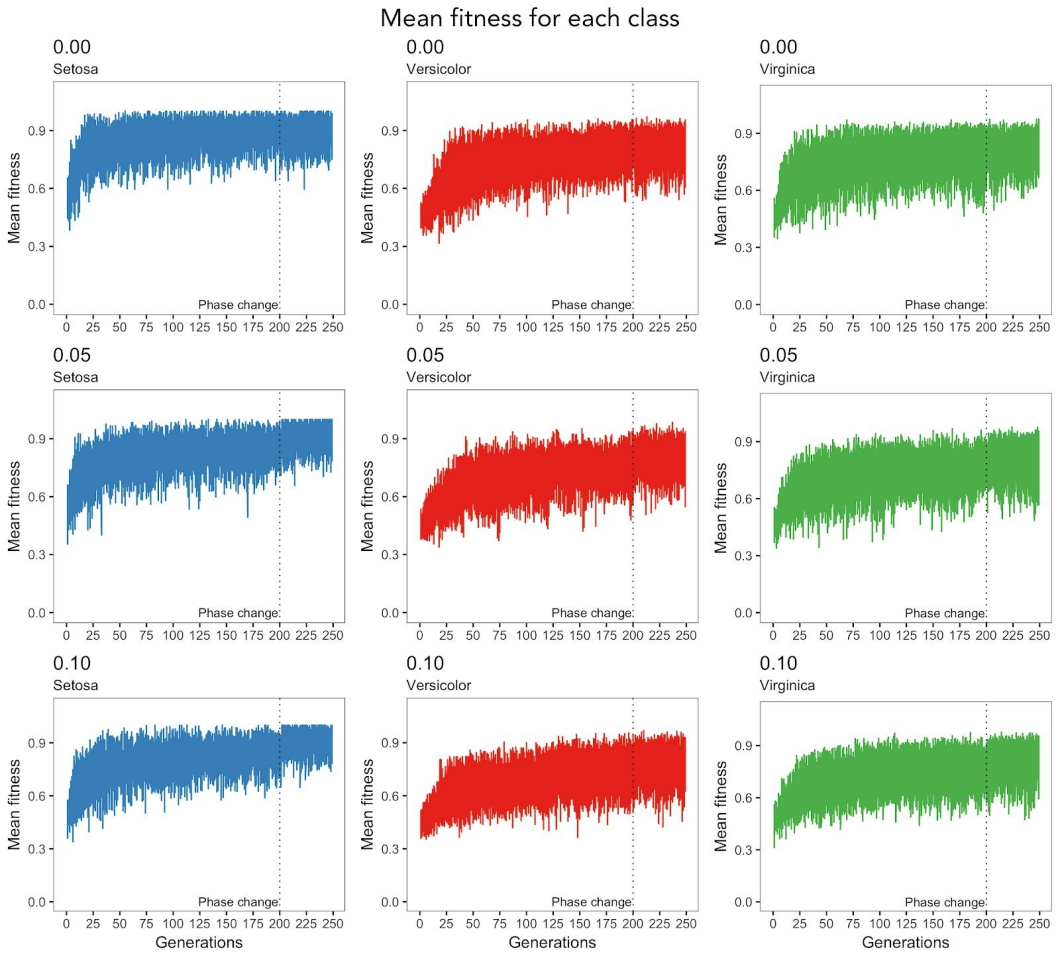

Figure 44: Mean of fitness of training partition in each class subpopulation through GP evolution for different cooperation intensity rate decrease values.

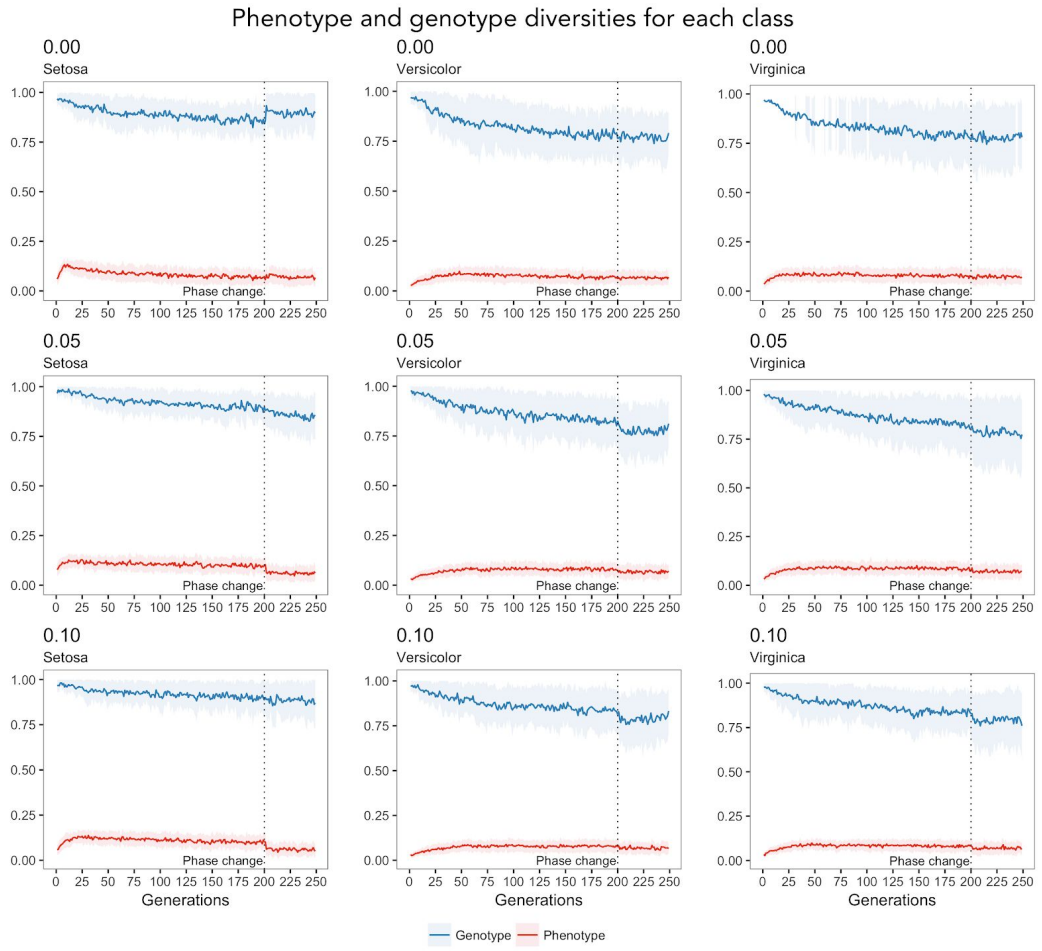

Figure 45: Genotype and phenotype diversities mean and one standard deviation for each class subpopulation through GP evolution for different cooperation intensity rate decrease values.

1.1.5. Team prediction

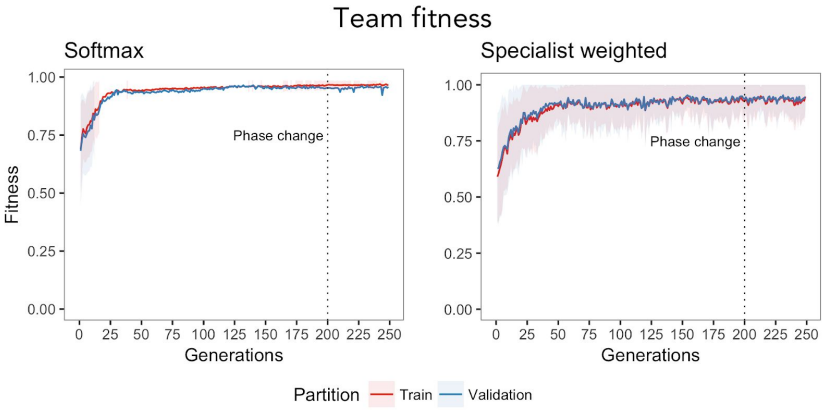

Figure 46: Mean and one standard deviation of train and validation team fitness through the GP evolution for each team prediction method.

Figure 47: Mean and one standard deviation of train and validation best team fitness through the GP evolution for each team prediction method.

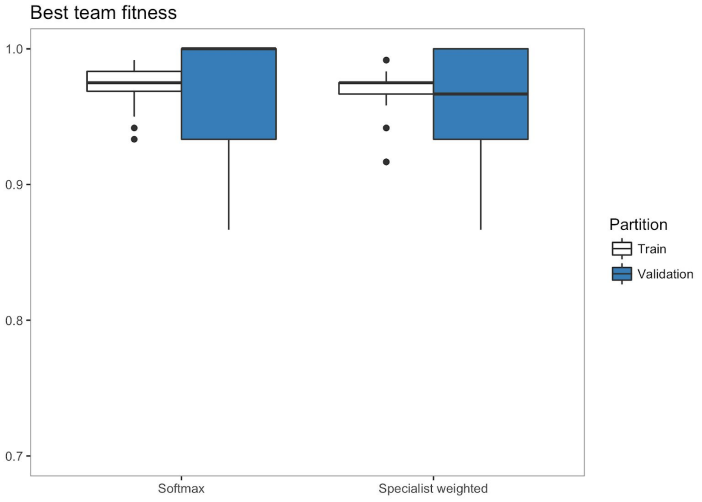

Figure 48: Train and validation best team final fitnesses for each team prediction method.

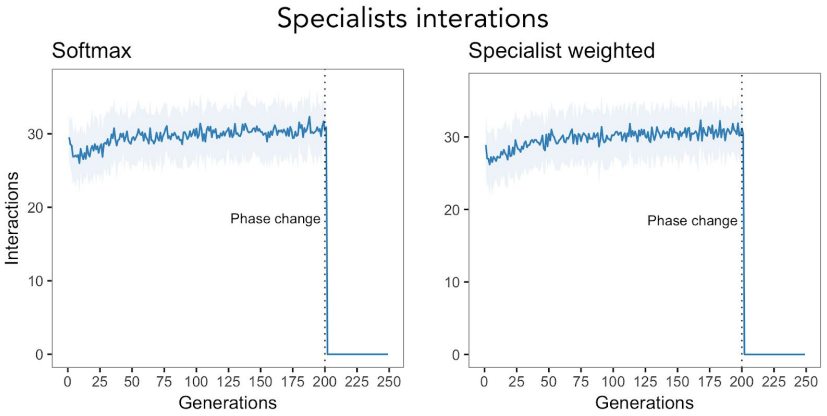

Figure 49: Specialists interactions mean and one standard deviation through GP evolution for each team prediction method.

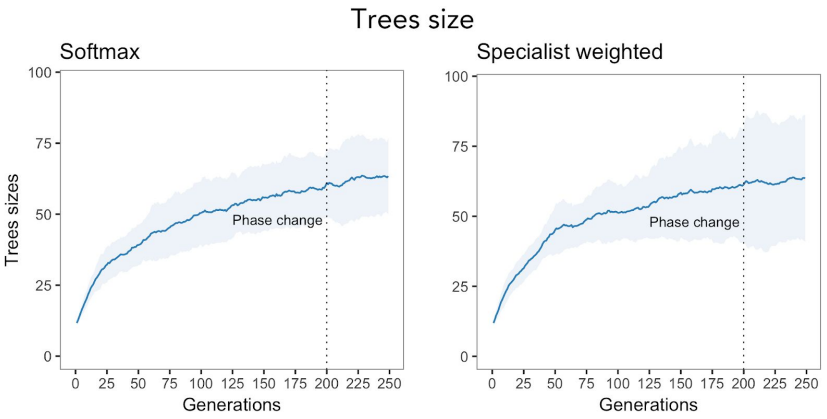

Figure 50: Trees sizes mean and one standard deviation through GP evolution for each team prediction method.

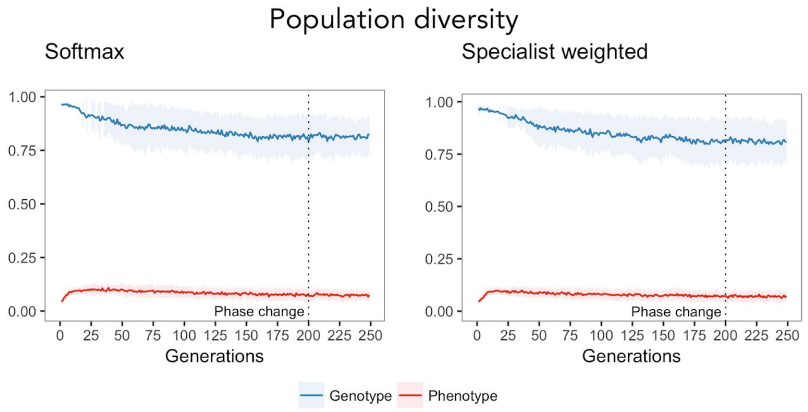

Figure 51: Trees sizes mean and one standard deviation through GP evolution for each team prediction method.

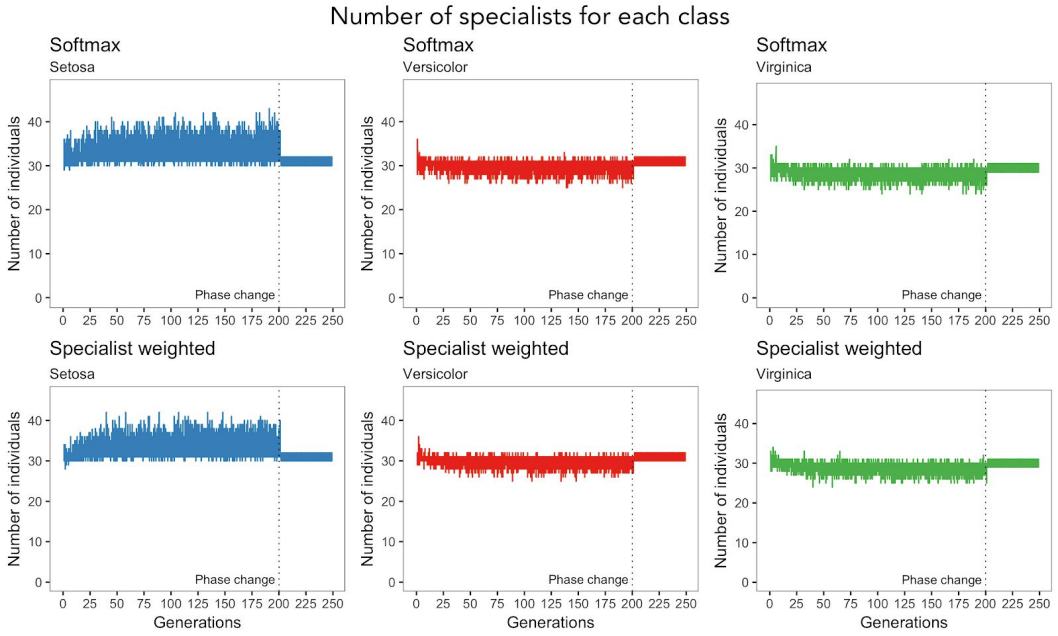

Figure 52: Number of specialised individuals in each class subpopulation through GP evolution for each team prediction method.

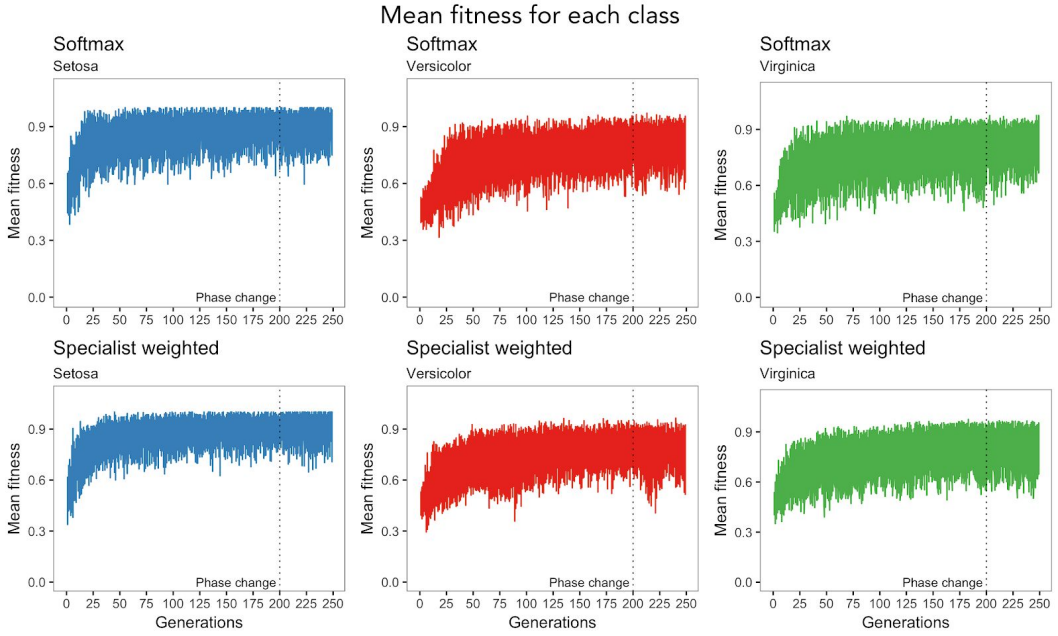

Figure 53: Mean of fitness of training partition in each class subpopulation through GP evolution for each team prediction method.

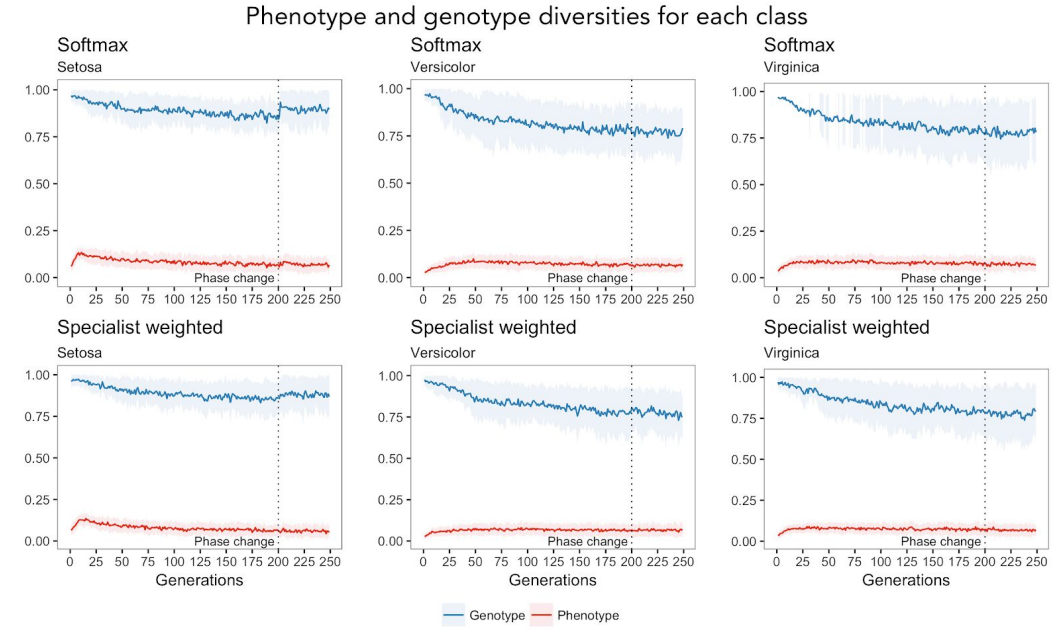

Figure 54: Genotype and phenotype diversities mean and one standard deviation for each class subpopulation through GP evolution for each team prediction method.

1.1.6. Teams mutation operator

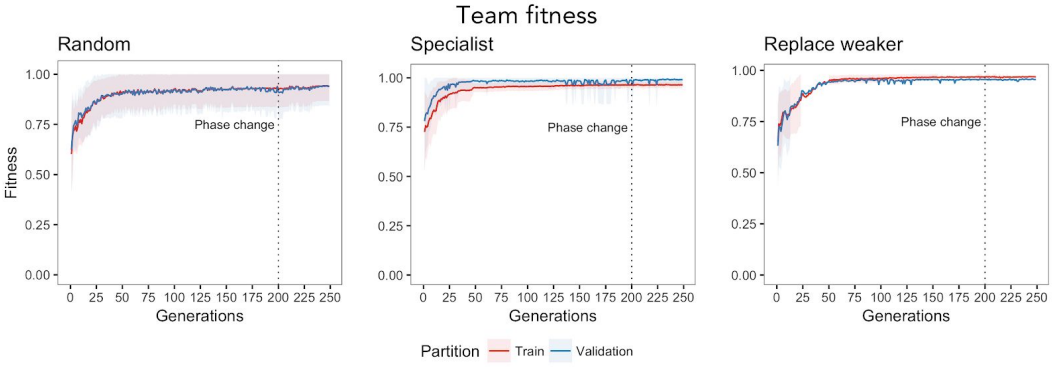

Figure 55: Mean and one standard deviation of train and validation team fitness through the GP evolution for each team mutation operator.

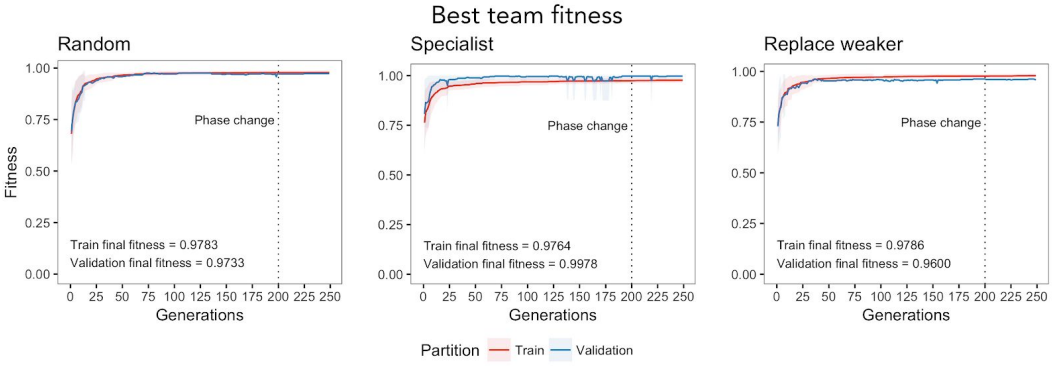

Figure 56: Mean and one standard deviation of train and validation best team fitness through the GP evolution for each team mutation operator.

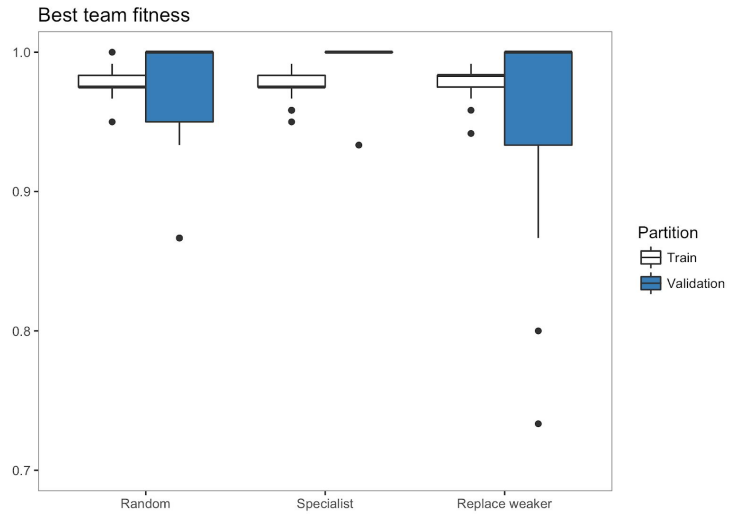

Figure 57: Train and validation best team final fitnesses for each team mutation operator.

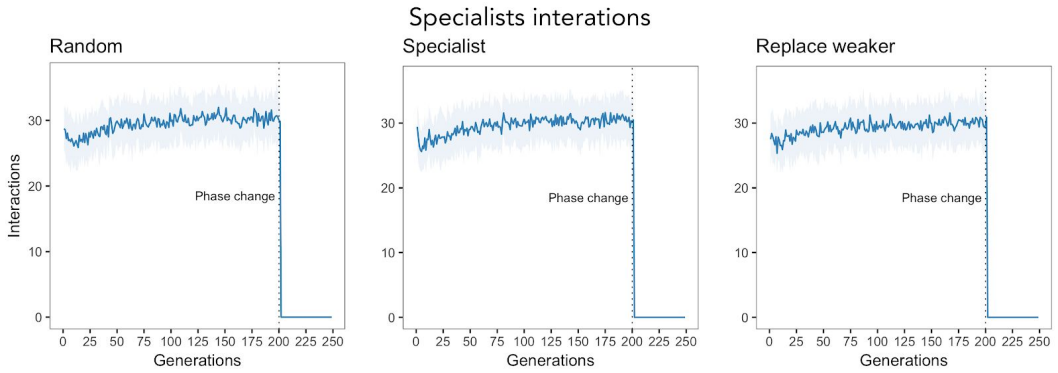

Figure 58: Specialists interactions mean and one standard deviation through GP evolution for each team mutation operator.

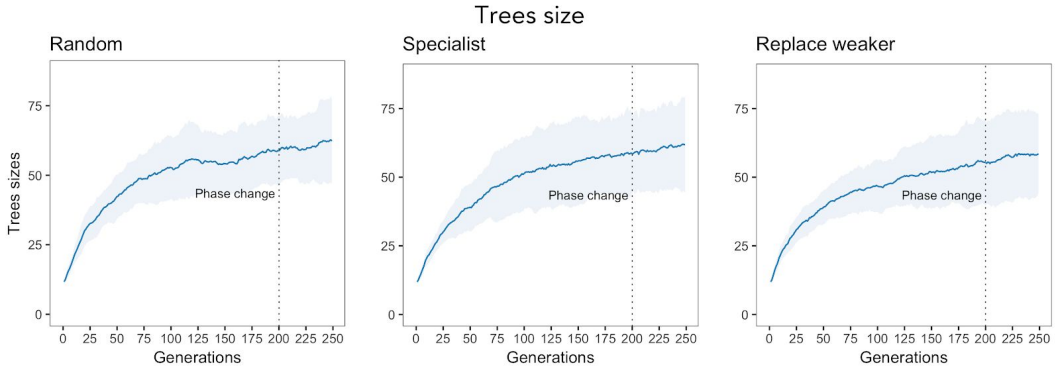

Figure 59: Trees sizes mean and one standard deviation through GP evolution for each team mutation operator.

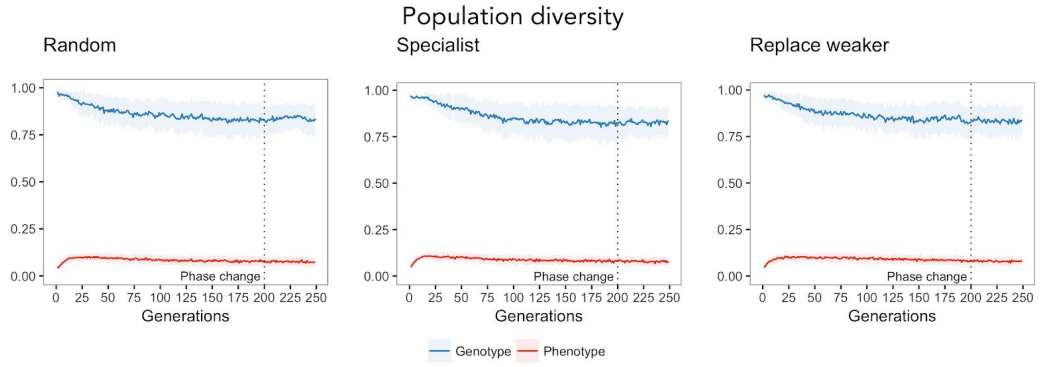

Figure 60: Trees sizes mean and one standard deviation through GP evolution for each team mutation operator.

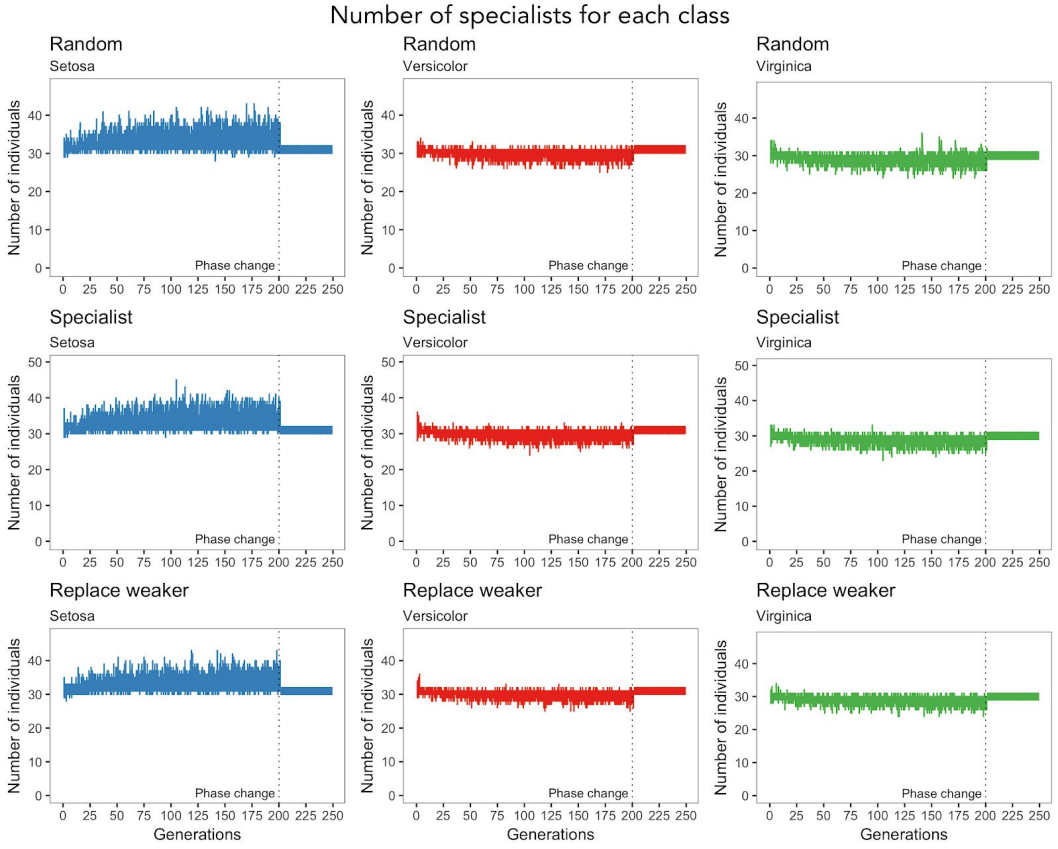

Figure 61: Number of specialised individuals in each class subpopulation through GP evolution for each team mutation operator.

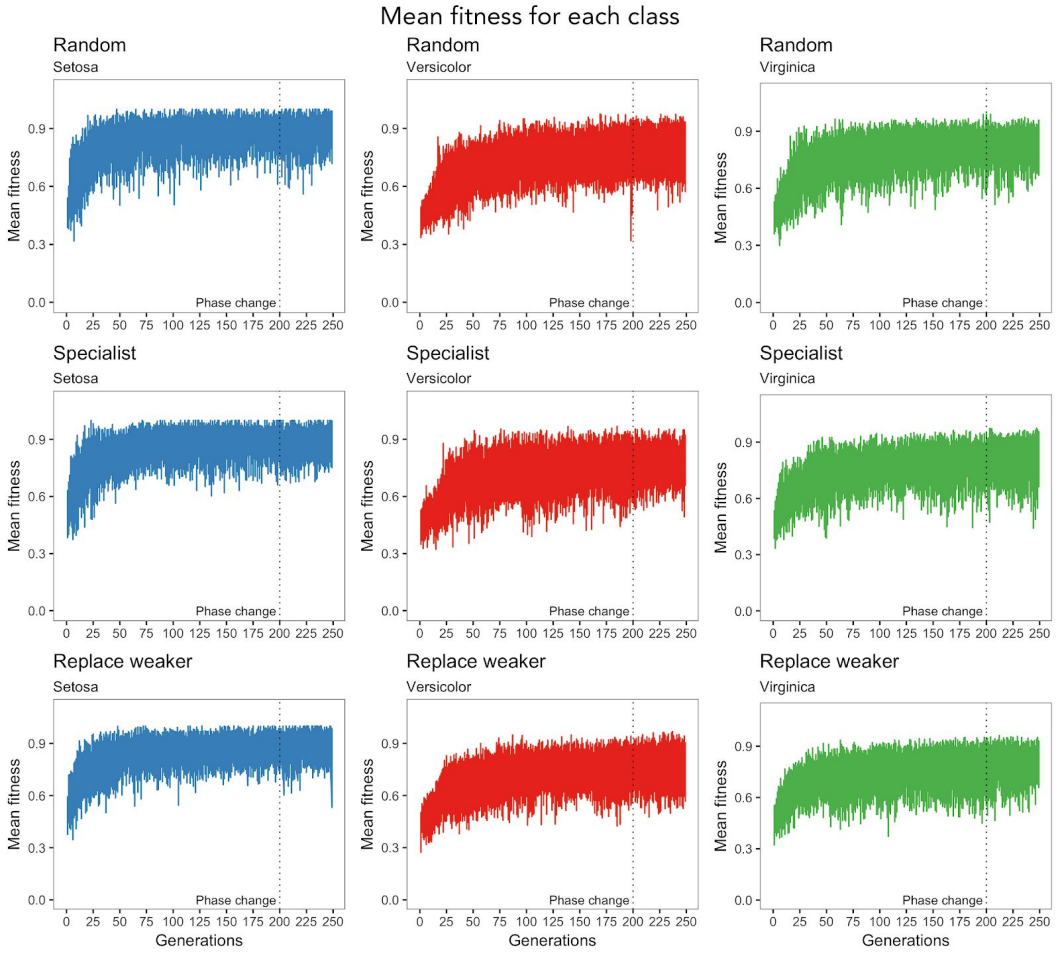

Figure 62: Mean of fitness of training partition in each class subpopulation through GP evolution foreach team mutation operator.

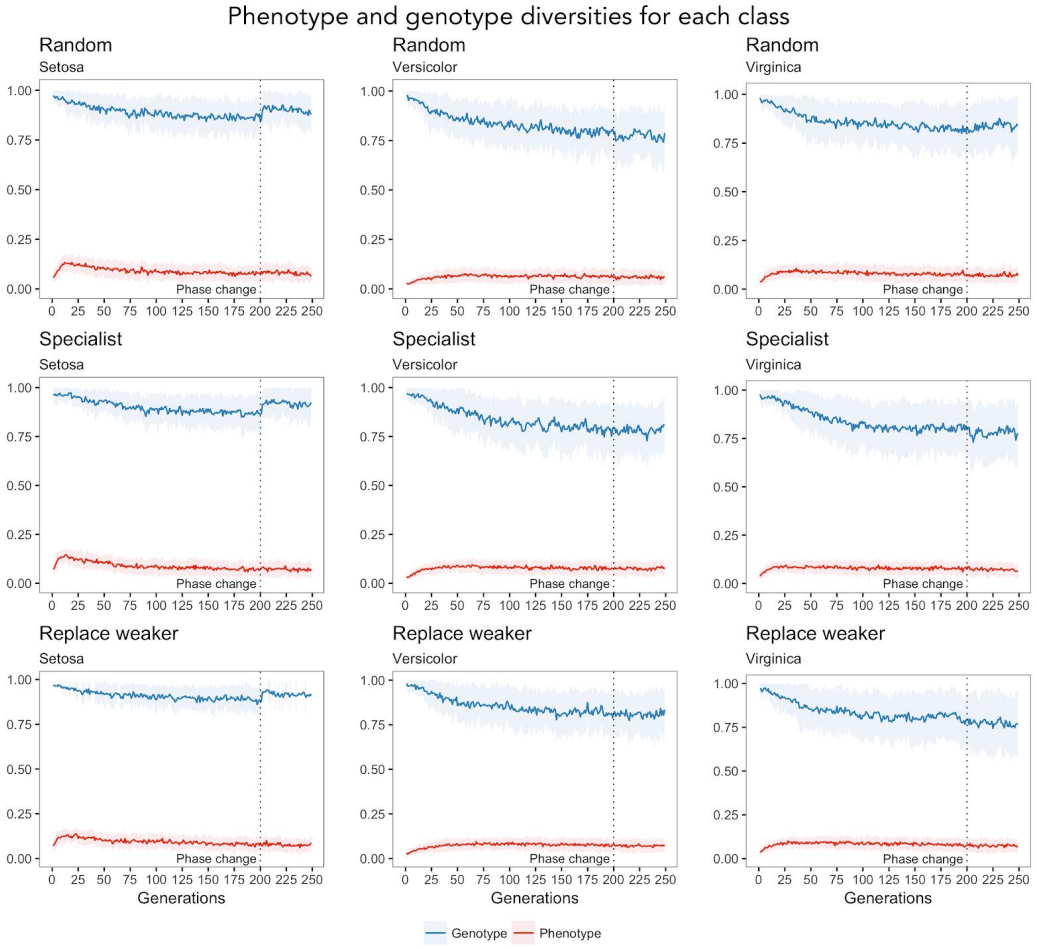

Figure 63: Genotype and phenotype diversities mean and one standard deviation for each class subpopulation through GP evolution for each team mutation operator.