SLUG: Feature Selection Using Genetic Algorithms and Genetic Programming

Nuno M. Rodrigues[1], João E. Batista[1], William La Cava[3], Leonardo Vanneschi[2] and Sara Silva[1]

[1] LASIGE Faculty of Sciences, University of Lisbon, Lisbon, Portugal
{nmrodrigues,jebatista,sara}@fc.ul.pt

[2] NOVA Information Management School (NOVA IMS), Universidade Nova de Lisboa, Campus de Campolide, 1070-312 Lisbon, Portugal
lvanneschi@novaims.unl.pt

[3] Boston Children's Hospital, Harvard Medical School, Boston, MA, USA
william.lacava@childrens.harvard.edu

This is the Author Peer Reviewed version of the following chapter/ conference paper published by Springer:

Rodrigues, N. M., Batista, J. E., La Cava, W., Vanneschi, L., & Silva, S. (2022). SLUG: Feature Selection Using Genetic Algorithms and Genetic Programming. In E. Medvet, G. Pappa, & B. Xue (Eds.), Genetic Programming: 25th European Conference, EuroGP 2022, Held as Part of EvoStar 2022, Madrid, Spain, April 20–22, 2022, Proceedings (pp. 68-84). (Lecture Notes in Computer Science; Vol. 13223). Springer. https://doi.org/10.1007/978-3-031-02056-8_5

FUNDING:

This work was supported by FCT, Portugal, through funding of LASIGE Research Unit (UIDB/00408/2020 and UIDP/00408/2020); MAR2020 program via project MarCODE (MAR-01.03.01-FEAMP-0047); projects BINDER (PTDC/CCI-INF/29168/2017), AICE (DSAIPA/DS/0113/2019), OPTOX (PTDC/CTA-AMB/30056/2017) and GADgET DSAIPA/DS/0022/2018). Nuno Rodrigues and João Batista were supported by PhD Grants 2021/05322/BD and SFRH/BD/143972/2019, respectively; William La Cava was supported by the National Library Of Medicine of the National Institutes of Health under Award Number R00LM012926.

SLUG: Feature Selection Using Genetic Algorithms and Genetic Programming

Nuno M. Rodrigues[1 [0000−0001−5312−8276]], João E. Batista[1 [0000−0002−2997−8550]], William La Cava[3 [0000−0002−1332−2960]], Leonardo Vanneschi[2 [0000−0003−4732−3328]], and Sara Silva[1 [0000−0001−8223−4799]]

[1] LASIGE, Faculty of Sciences, University of Lisbon, Portugal
[2] NOVA Information Management School (NOVA IMS), Universidade Nova de Lisboa, Campus de Campolide, 1070-312 Lisboa, Portugal
[3] Boston Children's Hospital, Harvard Medical School, Massachusetts, USA
{nmrodrigues,jebatista,sara}@fc.ul.pt
lvanneschi@novaims.unl.pt
william.lacava@childrens.harvard.edu

Abstract. We present SLUG, a method that uses genetic algorithms as a wrapper for genetic programming (GP), to perform feature selection while inducing models. This method is first tested on four regular binary classification datasets, and then on 10 synthetic datasets produced by GAMETES, a tool for embedding epistatic gene-gene interactions into noisy datasets. We compare the results of SLUG with the ones obtained by other GP-based methods that had already been used on the GAMETES problems, concluding that the proposed approach is very successful, particularly on the epistatic datasets. We discuss the merits and weaknesses of SLUG and its various parts, i.e. the wrapper and the learner, and we perform additional experiments, aimed at comparing SLUG with other state-of-the-art learners, like decision trees, random forests and extreme gradient boosting. Despite the fact that SLUG is not the most efficient method in terms of training time, it is confirmed as the most effective method in terms of accuracy.

Keywords: Feature Selection, Epistasis, Genetic Programming, Genetic Algorithms, Machine Learning

1 Introduction

Epistasis can generally be defined as the interaction between genes, and it is a topic of interest in molecular and quantitative genetics [7]. In machine learning (ML), several types of epistatic interactions have been studied. In evolutionary computation, epistasis has traditionally been interpreted as the interaction between characters, sets of characters or, generally speaking, parts of the chromosome representing solutions. This type of epistatic interaction has attracted the interest of researchers mainly because of its effect on fitness landscapes and, consequently, problem hardness. The topic has been studied since the early 90s (see for instance [8,30]), and one of the most popular outcomes of those studies was the NK-landscapes benchmark [2], in which the amount of epistasis is tunable by means of two parameters, N and K. This benchmark has been used in

several circumstances for testing the performance of genetic algorithm (GA) variants (see for instance [23,22,5,1,28,37], just to mention a few), and more recently it has also been extended to genetic programming (GP) [41]. An in-depth, although not very recent, survey of studies of epistasis in GA can be found in [31], while in [13] the effect of epistasis on the performance of GA is critically revised, highlighting the difficulty of GA in optimizing epistatic problems. In [19], epistasis was used to select the appropriate basis for basis change space transformations in GA, and in the same year [3] proposed a method to decipher the exact combinations of genes that trigger the epistatic effects, focusing on multi-effect and multi-way epistasis detection. Recently, a new benchmark was proposed [24] where epistasis-tunable test functions are constructed via linear combinations of simple basis functions. A different way of interpreting epistasis in ML is by studying the interactions between features in data. The problem of attribute interdependency is well known in ML. It has been studied in several approaches, using for instance several types of correlation [11] or mutual information [26].

In this paper, we tackle a rather different type of problem: we want to be able to deal with datasets where, among many variables, only a very limited number of them are useful and able to explain the target, and they must be used together for the model to be accurate. In other words, all the few "important" variables must be selected, while the many "confounding" ones must be left out. Furthermore, these few important variables are not necessarily correlated between each other, or have any other relationship of interdependency. This type of behavior can be observed, for instance, in some of Korn's benchmark problems proposed in [15], or in some medical problems, where finding epistasis can be crucial to identify the association between disease and genetic variants, and consequently be able to develop medical treatments and prevention [29]. It is of common intuition that, for problems characterized by such a typology of data, feature selection plays a crucial role, and the objective of this work is to propose a feature selection strategy that, integrated in a very natural way with the modelling algorithm, could be appropriate for working with epistatic datasets. The epistatic datasets studied in this paper have been generated using the GAMETES algorithm, introduced in [38], and have already been used in [16] as a benchmark to validate the M4GP classification method. Similar types of datasets have also been studied in [36], where a GP-based pipeline optimization tool (TPOT-MDR) was proposed to automatically design ML pipelines for bioinformatics problems. For tackling problems characterized by this type of data, Urbanowicz and colleagues recently presented RelieF-based feature selection [40], a unique family of filter-style feature selection algorithms that are sensitive to feature interactions and that can be applied to various types of problems, including classification and regression. In [16] this method has been coupled with M4GP, achieving state-of-the-art results on the GAMETES datasets.

Our proposal consists of using GA for feature selection. The idea, presented for instance in [17,4,21], is framed in a well established research track, and surveys can be found in [12,43]. With the proliferation of data and the consequent development of ML, the use of GA for feature selection increased in the last decade. Numerous recent contributions can be found, for instance, aimed at improving the method in presence of vast amounts of data [18,6], or applying the method in several different real-world scenarios, including medicine [42], economy [34], image processing [33] and sociology [10],

just to mention a few. However, we match GA with another evolutionary algorithm, Genetic Programming (GP), obtaining an integrated, and purely evolutionary, method that is able to perform feature selection and at the same time induce good models using the selected features. The GA part acts as a wrapper to the GP part, that is the learner. We call our approach SLUG, and compare it to both standard GP and other GP-based algorithms already used on the GAMETES datasets, such as M3GP [25] and M4GP [16]; we also compare it with other GA-wrapped ML classifiers that also perform feature selection, such as decision trees, random forests and XGBoost. In [35], the authors propose an opposite methodology to SLUG, were they GP for feature selection and GA for feature construction, in an iterative away as opposed to our wrapped approach.

2 SLUG

Our method, feature **S**e**L**ection **U**sing **G**enetic algorithms and genetic programming (SLUG), uses a cooperative approach between these two evolutionary algorithms, where the quality of each GA individual is assessed by running GP with the features selected by GA. A schematic showing the behavior of the full SLUG pipeline can be seen in Figure 1, with the evaluation of the individuals being detailed in Figure 2.

After splitting the data into training and test sets, a standard GA is applied to the training set (Figure 1). The individuals of this GA are binary strings of length equal to the number of features in the data. Each bit of the chromosome represents one feature, where 1 or 0 mean that the respective feature is selected or not, respectively. After initializing such a population, it is evaluated. The evaluation of each GA individual requires a complete run of standard GP, using the same training set as the GA but seeing only the features selected by the GA individual. The best fitness achieved in the GP run is the fitness of the GA individual (Figure 2). Once every GA individual has been evaluated, a new GA population is formed by applying selection and the genetic operators, and after a number of generations the GA finishes and returns both the chromosome with the best selected features and the GP model that achieved the best results when given those features (Figure 1). Finally, the best GP model is then evaluated using the GA selected features of the test dataset.

Naturally, the GP model does not have to use all the GA selected features, since it also performs its own feature selection. In fact, this is one of the strengths of SLUG for epistatic datasets. On the one hand, the number of useful features on the GAMETES datasets is so low that not even a method like GP, that is so well equipped with feature selection abilities, can isolate them from the numerous other ones. On the other hand, the GA only has to reduce the number of features that GP can potentially use, so its task is facilitated. In other words, the strength of SLUG is that the feature selection step does not need to be accurate: as long as the right features are among a reasonable number of selected ones, GP can do the rest of the job.

3 Data

We test our method on two distinct sets of problems. For the first set, we use four standard binary classification problems: HRT (Heart) [9]; ION (Ionosphere) [9], PRK

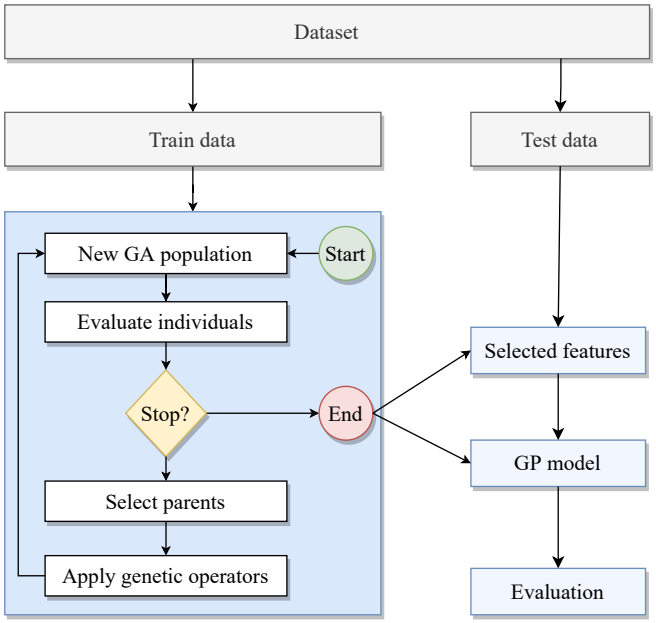

Fig. 1. A graphical representation of the SLUG pipeline.

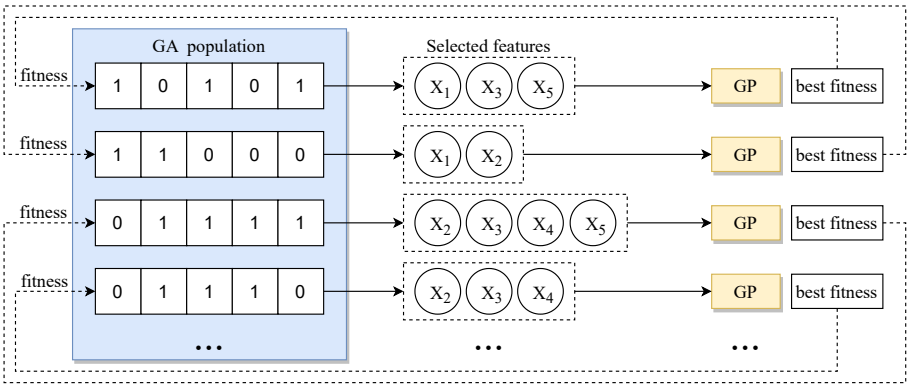

Fig. 2. Illustration of the way the GA individuals are evaluated by running GP with only the selected features. The best fitness of the GP run is the fitness of the respective GA individual.

(Parkinsons) [9] and SON (Sonar) [44]. Details regarding the composition of these datasets can be found in Table 1.

For the second set, we use GAMETES datasets, which are produced by a tool for embedding epistatic gene-gene interactions into noisy genetic datasets [39]. We use 10 different problems that vary according to two measures of difficulty: number of fea-

Table 1. Number of features, observations and negative/positive ratio on each dataset.

Datasets	HRT	ION	PRK	SON
Features	13	33	23	61
Observations	270	351	195	208
Neg/Pos Ratio	45/55	65/35	75/25	46/54

tures (10, 100, 1000) and signal-to-noise ratio (0.05, 0.1, 0.2, 0.4). For each problem, a two-way epistatic interaction is present that is predictive of the disease classification, but this is masked by the presence of confounding features and noisy disease classifications. Due to computational and time constraints, we did not perform experiments on all the possible combinations of number of features and signal-to-noise ratio. The 10 selected datasets are (with simplified names in the format $features\_ratio$): 10_005, 10_01, 10_02 and 10_04; 100_005, 100_01, 100_02 and 100_04; 1000_02 and 1000_04. All the gametes datasets are balanced.

4 Methods

Besides standard GA and standard GP, which are part of the SLUG method, we also compare our results with the following GP-based methods:

M3GP: M3GP stands for multidimensional multiclass GP with multidimensional populations [25]. Originally designed for multiclass classification, in M3GP each individual is composed of a mutable number of trees, also called dimensions, from which we extract a set of hyper-features that are then given to a classifier. Along with the standard crossover and mutation operators, M3GP includes an additional crossover, which swaps dimensions between individuals, and two additional mutations, which add/remove dimensions to/from an individual. The fitness of each individual is calculated by running a classifier on the hyper-feature space created by the trees of the individual. On the original implementation of M3GP, this is by default the Mahalanobis distance classifier.

M4GP: While the M3GP uses a tree-based structure for the individuals, M4GP, the successor of M3GP, uses a stack-based structure, which naturally provides support for multiple outputs. Regarding genetic operators, M4GP uses stack-based operators that are equivalent to the ones used by M3GP. For selection, M4GP uses lexicase selection, which out-preformed standard tournament selection, and age-fitness Pareto survival selection in experiments [16].

M4GP+EKF: Expert knowledge filter (EFK) is a preprocessing feature selection algorithm from the RelieF family [14]. In M4GP+EKF it is used to reduce the dataset to the top 10 features before giving it to the M4GP algorithm [16]. From now on we will call this variant M4GP-E.

As part of the discussion, we also present some results obtained by replacing the GP part of SLUG with other ML methods, namely, decision trees (DT), random forests (RF) and extreme gradient boosting, better known as XGBoost (XGB).

5 Experimental Setup

We run SLUG for 50 generations of the GA, using a population of 100 individuals. The GP populations also have 100 individuals, but they evolve for only 30 generations, which our initial experiments revealed to be sufficient to evaluate the quality of the selected features. GP uses the traditional binary arithmetic operators $[+, -, /, *]$ and no random constants. Fitness is the overall accuracy in the training set, measured after transforming the real-valued outputs of GP into class labels. The best fitness of each GP run is passed to the GA as the fitness of each individual, as explained in Section 2, and therefore the GA (and therefore SLUG) also uses the overall accuracy as fitness (as do all the other GP and non-GP methods used here). Both GA and GP select the parents of the next generation using tournaments of size 5. Regarding the genetic operators, GP uses the standard subtree crossover and mutation with 50% probability each. GA also uses standard crossover that swaps same-sized blocks between 2 chromosomes with probability of 70%, and standard mutation that performs bit-flip on the chromosome with probability of $1/n$ (where n is the population size) and each bit has probability of $1/m$ of being flipped (where m is the length of the chromosome, i.e., the number of features of the problem). Both GA and GP use some elitism: GP guarantees that the best individual of one generation survives into the next; GA does not guarantee the survival of the best chromosome from one generation to the next, to avoid diversity loss, but it keeps track and returns the best chromosome (and respective GP model) that was ever achieved during the entire run.

Standard GP, M3GP and both M4GP variants all use populations of 500 individuals evolving for 100 generations and, like SLUG, they all use tournaments of size 5. For more specific details on the M3GP and M4GP implementations and settings, the reader should consult Section 4 and the papers cited therein. The implementation of the GP methods will be available for download once the paper is accepted.

The STGP and M3GP implementations we use in this work can be found here [4].

Regarding the methods DT, RF and XGB mentioned in the discussion, we use the implementations provided by Scikit-learn [27]. We perform hyperparameter optimization by means of grid search with 5-fold cross-validation on the entire dataset, for each of the three methods. For DT we optimize the split criterion and maximum depth; for RF we optimize the split criterion, number of estimators and maximum depth; for XGB we optimize the learning rate, maximum depth and number of estimators. The GA runs with the exact same parameters as SLUG. In all cases, we randomly split the datasets 30 times, one for each run, into 70% training and 30% test.

6 Results

We measure the overall accuracy of the methods and present the results as boxplots (training and test) of the 30 runs and tables with the (test) medians. To assess the statistical significance of the results, we perform one-way non-parametric ANOVA analysis by means of Kruskal-Wallis with Holm correction, using 0.05 as the significance threshold. The Appendix contains the Holm-corrected p-values obtained in all the datasets.

[4] https://github.com/jespb/Python-STGP and https://github.com/jespb/Python-M3GP

6.1 Regular classification tasks

Taking into consideration the results presented in Table 2, Figure 3 and Appendix Table 4, we can see that our approach performs well, on par with the other GP methods such as M3GP and M4GP. Compared to the baseline of standard GP, SLUG performs better on both HRT and PRK datasets, and presents no significant differences on the remaining two. Regarding the M3GP and M4GP baselines, the results are also positive, with SLUG outperforming both methods on one problem, presenting no significant difference on two others, and being outperformed in the remaining problem. Lastly, regarding M4GP-E, this method outperforms SLUG in one problem, and no significant difference was found between them in the remaining problems. Finally, we could not help but notice one thing that appears to be different between SLUG and the other methods, that is the consistently lower dispersion of the results on training.

Table 2. Median test overall accuracy of the different methods on the non-gametes binary classification tasks. Best results for each problem are identified in **green**, more than one when there are no statistically significant differences.

	HRT	PRK	ION	SON
GP	0.778	0.831	**0.858**	0.698
M3GP	0.790	**0.881**	**0.873**	**0.786**
M4GP	0.784	**0.864**	**0.868**	**0.762**
M4GP-E	**0.802**	**0.873**	**0.854**	**0.738**
SLUG	**0.827**	**0.864**	**0.877**	0.730

6.2 Gametes classification tasks

Taking into consideration the results presented in Table 3, Figure 4 and Appendix Table 5, the first thing to notice is the fact that the standard GP baseline was one of the best methods on the 10-feature gametes problems. It outperformed both M4GP and M4GP-E on the 10_005 dataset, M4GP-E on 10_01, and all except SLUG on 10_02 [5]. We hypothesize that, on these easier problems, the exploration of different dimensional feature spaces that M3GP and M4GP perform is not helpful to the search, preventing the exploitation of better solutions.

Regarding our approach, the results were again highly positive, with SLUG invariably being one of the top performing methods in all problems. The GA of SLUG is able to filter out most, if not all, of the redundant features, which are then further filtered by the standard GP populations, also producing a ready to use model to apply to the problem.

On the 1000_04 dataset, SLUG produced results significantly worse than M4GP-E. We attribute this to the default parameterization of SLUG, which always uses very small populations of 100 individuals. Particularly in the GA, this is too small to appropriately explore the search space on the 1000-feature problems, and therefore the method is not fully capable of filtering out the redundant features. To confirm this hypothesis, we

[5] We performed 30 runs using the same total number of comparisons as SLUG using the STGP (10000 individuals and 1500 generations). With this, the median test accuracy achieved was 0.4982, while the best was 0.5348

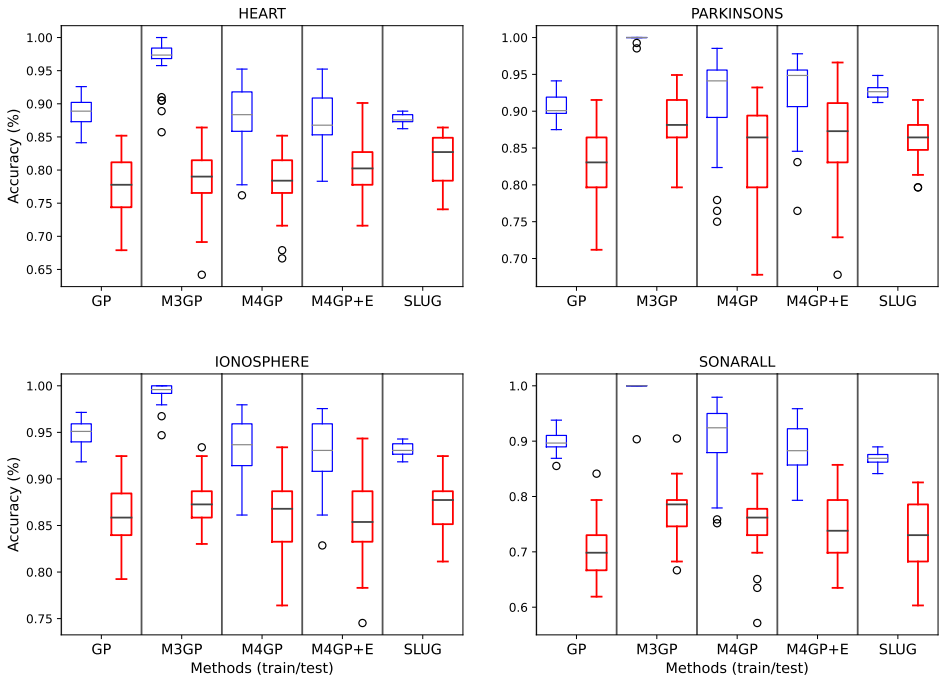

Fig. 3. Performance on the non-gametes binary classification datasets. Each plot contains, for each method, the results on the training (left) and test (right) sets.

ran SLUG with a larger GA population of 200 individuals. Although this is the double of the previous population size, it is still a very low number of individuals for such a large search space (however, further increasing the size of the population becomes computationally demanding, an issue that is discussed later). We named this variation SLUG Large (SLUG-L). As it can be seen on Figure 5, particularly on the 1000_04 dataset, SLUG-L is slightly improved, enough to be significantly better than the other solutions, and not significantly worse than M4GP-E.

Once again we notice that SLUG exhibits a much lower dispersion of results than most other methods (Figure 4), this time not only on training but also on test. This remarkable stability is shared by M4GP-E, although not so strongly.

7 Discussion

From the previous results we can state that SLUG is a powerful method that performs feature selection while inducing high-quality models. On the set of four regular problems, it was one of the best methods in three of them. On the set of 10 gametes problems, it was always one of the best methods, although it required a larger population size on one of them. On this one exception, one of the hardest problems, the other winner besides SLUG-L was M4GP-E. Published results on M4GP [16] had already shown that wrapping a feature selection method around a powerful classifier can improve the re-

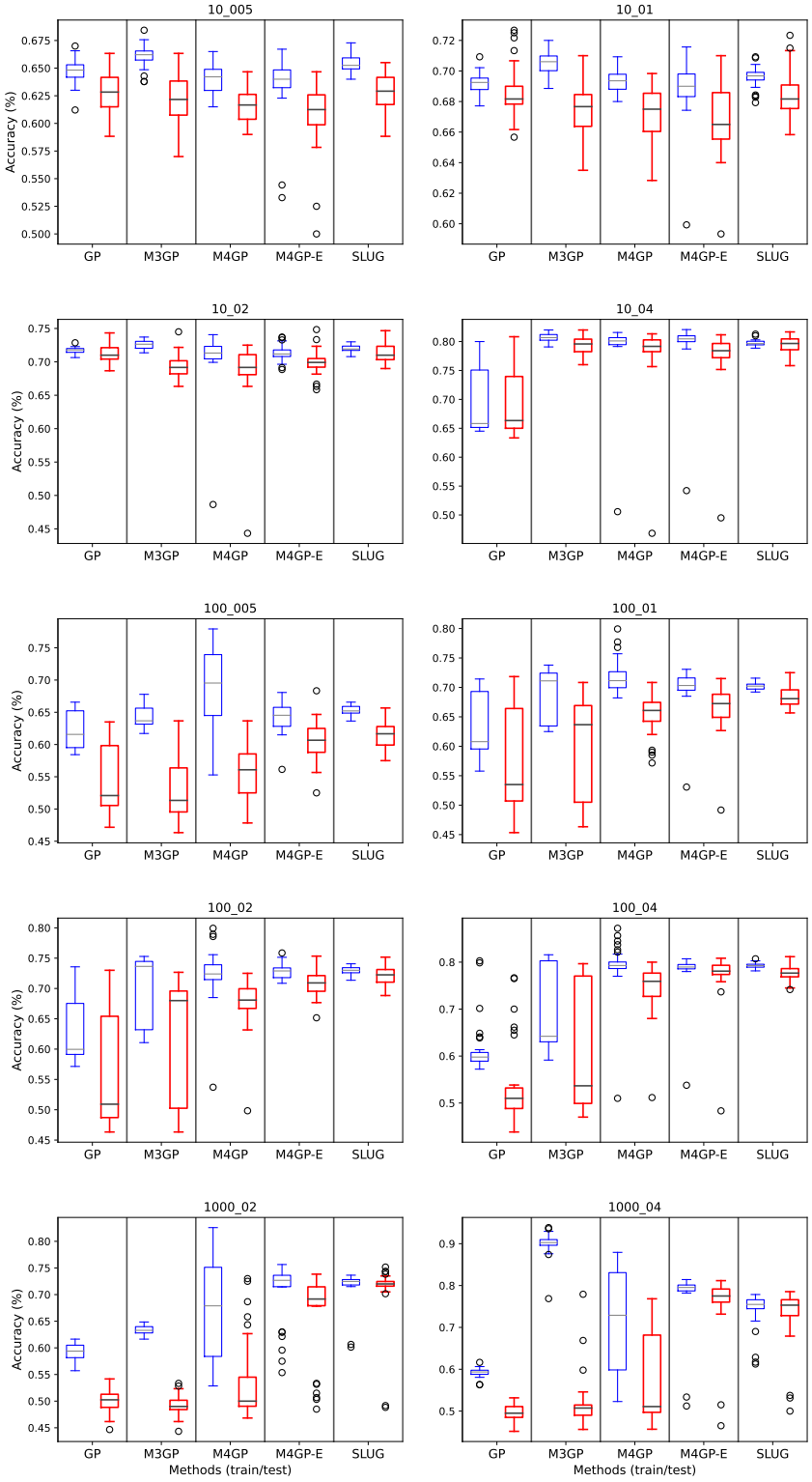

Fig. 4. Performance on the gametes datasets. Each plot contains, for each method, the results on the training (left) and test (right) sets.

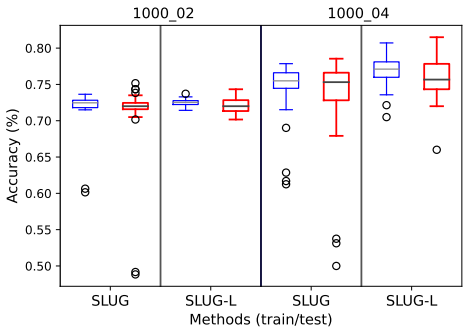

Fig. 5. Performance of different SLUG variants on the two higher dimensional gametes datasets. Each plot contains, for each variant, the results on the training (left) and test (right) sets.

Table 3. Median test overall accuracy of the different methods on the gametes tasks. Best results for each problem are identified in **green**, more than one when there are no statistically significant differences.

	10_005	10_01	10_02	10_04	100_005	100_01	100_02	100_04	1000_02	1000_04
GP	**0.628**	**0.682**	**0.710**	0.663	0.521	0.535	0.509	0.510	0.502	0.495
M3GP	**0.622**	**0.677**	0.692	**0.796**	0.513	0.637	0.680	0.537	0.490	0.507
M4GP	0.617	**0.675**	0.692	**0.792**	0.561	0.661	0.681	0.759	0.500	0.511
M4GP-E	0.613	0.665	0.699	**0.784**	**0.607**	**0.672**	0.709	**0.781**	0.692	**0.775**
SLUG	**0.629**	**0.682**	**0.710**	**0.797**	**0.617**	**0.681**	**0.722**	**0.777**	**0.720**	0.753
SLUG-L	-	-	-	-	-	-	-	-	**0.720**	**0.757**

sults significantly, and here we confirm that indeed, M4GP-E is often significantly better than M4GP. Reminding that SLUG is also the product of wrapping a feature selection method (GA) around a powerful classifier (GP), our results reconfirm the advantages of such an approach, since SLUG is very often significantly better than standalone GP.

Naturally, we are interested in searching for the best match between wrapper and learner, and we begin by exploring why SLUG performs so well; which of its parts is more important, the GA wrapper of the GP learner. On the one hand, we observe that M4GP is in general a stronger learner than GP; on the other hand, M4GP-E is not stronger than SLUG. Therefore, GA seems to be a better wrapper than the EKF used in M4GP-E, and the main responsible for the success of SLUG. While the combination of GA with M4GP seems like a promising match to explore in the future, for now we try to answer a simple question: is GA such a good wrapper that it can improve also the performance of other ML methods, arguably less powerful than the GP-based ones, like DT, RF and XGB? This question is not only academically interesting, but also important from a practical point of view. Two evolutionary algorithms nested in each other is never an efficient solution in terms of computational effort, so it is not a surprise that SLUG is... sluggish. Any of the three other mentioned ML methods runs much faster than GP, so wrapping GA around any of them could result in a much faster SLUG. Furthermore, like GP, these methods can also perform feature selection on their own, on

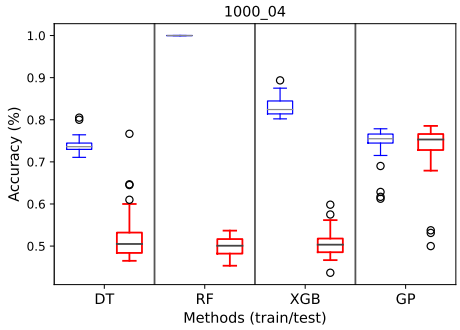

Fig. 6. Performance of different SLUG variants using DT, RF, XGB and GP (the original SLUG) on the gametes 1000_04 problem. Each plot contains, for each variant, the results on the training (left) and test (right) sets.

top of the preselection made by GA. Therefore, we experiment with alternative variants of SLUG where GP is replaced by DT, RF and XGB. The problem chosen to test these variants is the gametes 1000_04, coincidentally the one problem where SLUG-L was required because SLUG was not one of the best methods (see Section 6). We chose this particular problem because it has already been used in previous studies [16,32] where the standalone versions of DT, RF and XGB were unable to solve the problem.

The obtained results are shown in Figure 6 and reveal that, even when wrapped with GA, these methods are not able to solve the problem, and this means that GP is also essential for the success of SLUG. Since the other methods also perform feature selection, the reason why GP is essential is not clear, particularly after observing that in one of the 30 runs the DT method, which is undoubtedly the less powerful one, was able to obtain a high-quality model (highest outlier in test, Figure 6), and it did so after only 17 generations.

8 Conclusion and Future Work

We have presented SLUG, a method for feature selection using genetic algorithms (GA) and genetic programming (GP). SLUG implements a co-operative approach between these two evolutionary algorithms, where the quality of each GA individual is assessed by doing a GP run with the features selected by GA. The GA acts like a wrapper, selecting features for GP, the learner. At the end of the process, both the set of GA selected features and the best GP induced model are returned, and therefore SLUG comprises the entire pipeline from data preprocessing to predictive modeling. No efforts are put into the optimization of the model, as this is not the main purpose of our research.

We tested SLUG on four regular binary classification datasets, and on 10 synthetic datasets produced by GAMETES, a tool for embedding epistatic gene-gene interactions into noisy datasets. We compared the results of SLUG with the ones obtained by standard GP and other GP-based methods like M3GP and two different M4GP variants, one of them also wrapped by the EKF algorithm for feature selection. SLUG obtained the best results in practically all the problems, with special relevance for the good results

obtained on the epistatic datasets, whose difficulty was the driver for this research in the first place.

We discussed the merits and weaknesses of SLUG and the parts that compose it. Its slowness is its obvious limitation, as it requires considerable computational effort to run two evolutionary algorithms nested in each other. We experimented with alternative implementations, replacing the GP backbone of SLUG with faster methods like decision trees, random forests and XGBoost, all wrapped with GA for feature selection. However, even with tuned parameters, none of them was able to catch up with SLUG.

From the above, we conclude that SLUG is a powerful method that performs feature selection while inducing high-quality models, even without putting any efforts on model optimization. In the future, we intend to address the main limitation of SLUG, by reducing its computational demands and therefore making it less sluggish. Many other improvements and extensions are possible, like the ones described below.

The backbone of SLUG is currently standard GP, which is not appropriate for multiclass classification. However, it can be replaced by other methods. The ones we tried did not produce good models, however other options exist, including M3GP and M4GP themselves, which are some of the best GP-based multiclass classification methods available today. Replacing GP with M3GP would give us the added flexibility of being allowed to plug any learning algorithm to the pipeline to work with the hyper-features evolved by M3GP. Instead of GA+GP, we would have a GA+M3GP+$classifier$ pipeline, where GA preselects the features, M3GP uses them to build hyper-features tailored to $classifier$, and $classifier$ finally induces an optimized predictive model, with the added advantage that the classifier can be whatever method best suits the needs of the domain application. Naturally, the same rationale can be used for regression instead of classification.

Regarding the improvement of the wrapper, the main issue with GA is, and has always been, the delicate balance between exploration and exploitation, here with an intense concern regarding computational demands. On the one hand, we want to make GA converge faster to a good subset of selected features, also to save computational effort; on the other hand, it must be able to properly explore the search space, particularly on the most difficult higher dimensional problems, but without requiring large populations which would increase the computational time. We finalize by presenting some ideas on how to deal with this.

In order to accelerate convergence, GP (or any other backbone SLUG is using) could inform GA of what features are actually being used, from the ones preselected. In case the backbone does not perform feature selection itself, it can probably still inform what features are more important. This way, the GA could use more information from the learner than just the fitness achieved with each subset of features, increasing the cooperation between the two methods. It is reasonable to think that, in this case, the GA binary chromosomes would become real-valued ones, where each bit would now contain a sort of probability of selecting each feature, that the learner could use to build its own models. In order to promote the exploration of the search space without having to increase the population size, and particularly when adding measures for faster convergence, our idea is to use novelty search [20] on the GA in order to increase the bias towards yet unexplored subsets of features.

Acknowledgments

This work was supported by FCT, Portugal, through funding of LASIGE Research Unit (UIDB/00408/2020 and UIDP/00408/2020); MAR2020 program via project MarCODE (MAR-01.03.01-FEAMP-0047); projects BINDER (PTDC/CCI-INF/29168/2017), AICE (DSAIPA/DS/0113/2019), OPTOX (PTDC/CTA-AMB/30056/2017) and GADgET (DSAIPA/DS/0022/2018). Nuno Rodrigues and João Batista were supported by PhD Grants 2021/05322/BD and SFRH/BD/143972/2019, respectively; William La Cava was supported by the National Library Of Medicine of the National Institutes of Health under Award Number R00LM012926.

References

1. Aguirre, H.E., Tanaka, K.: Genetic algorithms on nk-landscapes: Effects of selection, drift, mutation, and recombination. In: Cagnoni, S., Johnson, C.G., Cardalda, J.J.R., Marchiori, E., Corne, D.W., Meyer, J.A., Gottlieb, J., Middendorf, M., Guillot, A., Raidl, G.R., Hart, E. (eds.) Applications of Evolutionary Computing. pp. 131–142. Springer Berlin Heidelberg, Berlin, Heidelberg (2003)
2. Altenberg, L.: B2.7.2. nk fitness landscapes. In: Handbook of Evolutionary Computation. p. B2.7:5—B2.7:10. IOP Publishing Ltd and Oxford University Press (1997)
3. Ansarifar, J., Wang, L.: New algorithms for detecting multi-effect and multi-way epistatic interactions. Bioinformatics **35**(24), 5078–5085 (06 2019). https://doi.org/10.1093/bioinformatics/btz463
4. Chaikla, N., Qi, Y.: Genetic algorithms in feature selection. In: IEEE SMC'99 Conference Proceedings. 1999 IEEE International Conference on Systems, Man, and Cybernetics (Cat. No.99CH37028). vol. 5, pp. 538–540 vol.5 (1999). https://doi.org/10.1109/ICSMC.1999.815609
5. Chan, K., Aydin, M., Fogarty, T.: An epistasis measure based on the analysis of variance for the real-coded representation in genetic algorithms. In: The 2003 Congress on Evolutionary Computation, 2003. CEC '03. vol. 1, pp. 297–304 Vol.1 (2003). https://doi.org/10.1109/CEC.2003.1299588
6. Chiesa, M., Maioli, G., Colombo, G.: Gars: Genetic algorithm for the identification of a robust subset of features in high-dimensional datasets. BMC Bioinformatics **21**(54) (2020). https://doi.org/https://doi.org/10.1186/s12859-020-3400-6
7. Cordell, H.J.: Epistasis: what it means, what it doesn't mean, and statistical methods to detect it in humans. Human Molecular Genetics **11**(20), 2463–2468 (10 2002). https://doi.org/10.1093/hmg/11.20.2463
8. Davidor, Y.: Epistasis variance: A viewpoint on ga-hardness. Foundations of Genetic Algorithms, vol. 1, pp. 23–35. Elsevier (1991). https://doi.org/https://doi.org/10.1016/B978-0-08-050684-5.50005-7
9. Dua, D., Graff, C.: UCI machine learning repository (2017), `http://archive.ics.uci.edu/ml`
10. García-Dominguez, A., Galván-Tejada, C.E., Zanella-Calzada, L.A., Gamboa-Rosales, H., Galván-Tejada, J.I., Celaya-Padilla, J.M., Luna-García, H., Magallanes-Quintanar, R.: Feature selection using genetic algorithms for the generation of a recognition and classification of children activities model using environmental sound. Mobile Information Systems **Article ID 8617430**, 12 pages (2020). https://doi.org/https://doi.org/10.1155/2020/8617430
11. Hall, M.A.: Correlation-based Feature Selection for Machine Learning. Ph.D. thesis (1999)

12. Hussein, F., Kharma, N., Ward, R.: Genetic algorithms for feature selection and weighting, a review and study. In: Proceedings of Sixth International Conference on Document Analysis and Recognition. pp. 1240–1244 (2001). https://doi.org/10.1109/ICDAR.2001.953980
13. Jafari, S., Kapitaniak, T., Rajagopal, K., Pham, V.T., Alsaadi, F.: Effect of epistasis on the performance of genetic algorithms. Journal of Zhejiang University-SCIENCE A **20** (11 2018). https://doi.org/10.1631/jzus.A1800399
14. Kononenko, I.: Estimating attributes: Analysis and extensions of relief. In: ECML (1994)
15. Korns, M.F.: Genetic programming symbolic classification: A study. In: Banzhaf, W., Olson, R.S., Tozier, W., Riolo, R. (eds.) Genetic Programming Theory and Practice XV. pp. 39–54. Springer International Publishing, Cham (2018)
16. La Cava, W., Silva, S., Danai, K., Spector, L., Vanneschi, L., Moore, J.H.: Multidimensional genetic programming for multiclass classification. Swarm and Evolutionary Computation **44**, 260–272 (2019). https://doi.org/https://doi.org/10.1016/j.swevo.2018.03.015
17. Lanzi, P.: Fast feature selection with genetic algorithms: a filter approach. In: Proceedings of 1997 IEEE International Conference on Evolutionary Computation (ICEC '97). pp. 537–540 (1997). https://doi.org/10.1109/ICEC.1997.592369
18. Lavine, B.K., White, C.G.: Boosting the performance of genetic algorithms for variable selection in partial least squares spectral calibrations. Appl. Spectrosc. **71**(9), 2092–2101 (Sep 2017)
19. Lee, J., Kim, Y.H.: Epistasis-based basis estimation method for simplifying the problem space of an evolutionary search in binary representation. Complexity. **Article ID 2095167**, 13 pages (2019)
20. Lehman, J., Stanley, K.O.: Exploiting open-endedness to solve problems through the search for novelty. In: Proceedings of the Eleventh International Conference on Artificial Life (Alife XI. MIT Press (2008)
21. Li, A.D., Xue, B., Zhang, M.: Multi-objective feature selection using hybridization of a genetic algorithm and direct multisearch for key quality characteristic selection. Information Sciences **523**, 245–265 (2020). https://doi.org/https://doi.org/10.1016/j.ins.2020.03.032
22. Mathias, K.E., Eshelman, L.J., Schaffer, J.D.: Niches in nk-landscapes. In: Martin, W.N., Spears, W.M. (eds.) Foundations of Genetic Algorithms 6, pp. 27–46. Morgan Kaufmann, San Francisco (2001). https://doi.org/https://doi.org/10.1016/B978-155860734-7/50085-8
23. Merz, P., Freisleben, B.: On the effectiveness of evolutionary search in high-dimensional nk-landscapes. In: 1998 IEEE International Conference on Evolutionary Computation Proceedings. IEEE World Congress on Computational Intelligence (Cat. No.98TH8360). pp. 741–745 (1998). https://doi.org/10.1109/ICEC.1998.700144
24. Mo, H., Li, Z., Zhu, C.: A kind of epistasis-tunable test functions for genetic algorithms. Concurrency and Computation: Practice and Experience **33**(8), e5030 (2021). https://doi.org/https://doi.org/10.1002/cpe.5030, e5030 cpe.5030
25. Muñoz, L., Silva, S., Trujillo, L.: M3gp - multiclass classification with gp. In: EuroGP (2015)
26. Nazareth, D.L., Soofi, E.S., Zhao, H.: Visualizing attribute interdependencies using mutual information, hierarchical clustering, multidimensional scaling, and self-organizing maps. In: 2007 40th Annual Hawaii International Conference on System Sciences (HICSS'07). pp. 53–53 (2007). https://doi.org/10.1109/HICSS.2007.608
27. Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., Duchesnay, E.: Scikit-learn: Machine learning in Python. Journal of Machine Learning Research **12**, 2825–2830 (2011)
28. Pelikan, M., Sastry, K., Goldberg, D.E., Butz, M.V., Hauschild, M.: Performance of evolutionary algorithms on nk landscapes with nearest neighbor interactions and tunable overlap. In: Proceedings of the 11th Annual Conference on Genetic and Evolutionary Computation. p.

851–858. GECCO '09, Association for Computing Machinery, New York, NY, USA (2009). https://doi.org/10.1145/1569901.1570018
29. Petinrin, O.O., Wong, K.C.: Protocol for epistasis detection with machine learning using genepi package. Methods in molecular biology **2212**, 291–305 (2021)
30. Reeves, C.R., Wright, C.C.: Epistasis in genetic algorithms: An experimental design perspective. In: Proceedings of the 6th International Conference on Genetic Algorithms. p. 217–224. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA (1995)
31. Rochet, S.: Epistasis in genetic algorithms revisited. Information Sciences **102**(1), 133–155 (1997). https://doi.org/https://doi.org/10.1016/S0020-0255(97)00017-0
32. Rodrigues, N.M., Batista, J.E., Silva, S.: Ensemble genetic programming. In: Hu, T., Lourenço, N., Medvet, E., Divina, F. (eds.) Genetic Programming. pp. 151–166. Springer International Publishing, Cham (2020)
33. Seo, K.K.: Content-based image retrieval by combining genetic algorithm and support vector machine. In: de Sá, J.M., Alexandre, L.A., Duch, W., Mandic, D. (eds.) Artificial Neural Networks – ICANN 2007. pp. 537–545. Springer Berlin Heidelberg, Berlin, Heidelberg (2007)
34. shik Shin, K., Lee, Y.J.: A genetic algorithm application in bankruptcy prediction modeling. Expert Syst. Appl. **23**, 321–328 (2002)
35. Smith, M.G., Bull, L.: Feature construction and selection using genetic programming and a genetic algorithm. In: Ryan, C., Soule, T., Keijzer, M., Tsang, E., Poli, R., Costa, E. (eds.) Genetic Programming. pp. 229–237. Springer Berlin Heidelberg, Berlin, Heidelberg (2003)
36. Sohn, A., Olson, R.S., Moore, J.H.: Toward the automated analysis of complex diseases in genome-wide association studies using genetic programming. In: Proceedings of the Genetic and Evolutionary Computation Conference. p. 489–496. GECCO '17, Association for Computing Machinery, New York, NY, USA (2017). https://doi.org/10.1145/3071178.3071212
37. Tinós, R., Whitley, D., Chicano, F.: Partition crossover for pseudo-boolean optimization. In: Proceedings of the 2015 ACM Conference on Foundations of Genetic Algorithms XIII. p. 137–149. FOGA '15, Association for Computing Machinery, New York, NY, USA (2015). https://doi.org/10.1145/2725494.2725497
38. Urbanowicz, R., Kiralis, J., Sinnott-Armstrong, N., *et al.*: GAMETES: a fast, direct algorithm for generating pure, strict, epistatic models with random architectures. BioData Mining **5**(16) (2012). https://doi.org/https://doi.org/10.1186/1756-0381-5-16
39. Urbanowicz, R.J., Kiralis, J., Sinnott-Armstrong, N.A., Heberling, T., Fisher, J.M., Moore, J.H.: Gametes: a fast, direct algorithm for generating pure, strict, epistatic models with random architectures. BioData Mining **5**, 16 – 16 (2012)
40. Urbanowicz, R.J., Meeker, M., La Cava, W., Olson, R.S., Moore, J.H.: Relief-based feature selection: Introduction and review. Journal of Biomedical Informatics **85**, 189–203 (2018). https://doi.org/https://doi.org/10.1016/j.jbi.2018.07.014
41. Vanneschi, L., Castelli, M., Manzoni, L.: The k landscapes: A tunably difficult benchmark for genetic programming. In: Proceedings of the 13th Annual Conference on Genetic and Evolutionary Computation. p. 1467–1474. GECCO '11, Association for Computing Machinery, New York, NY, USA (2011). https://doi.org/10.1145/2001576.2001773
42. Wutzl, B., Leibnitz, K., Rattay, F., Kronbichler, M., Murata, M., Golaszewski, S.M.: Genetic algorithms for feature selection when classifying severe chronic disorders of consciousness. PLOS ONE **14**(7), 1–16 (07 2019). https://doi.org/10.1371/journal.pone.0219683
43. Xue, B., Zhang, M., Browne, W.N., Yao, X.: A survey on evolutionary computation approaches to feature selection. IEEE Transactions on Evolutionary Computation **20**(4), 606–626 (2016). https://doi.org/10.1109/TEVC.2015.2504420
44. Zhang, S.: sonar.all-data (2018), `https://www.kaggle.com/ypzhangsam/sonaralldata`

Appendix

Table 4. Holm corrected p-values using Kruskal-Wallis for the regular classification problems.

HRT	GP	M3GP	M4GP	M4GP-E	SLUG
GP	—	0.9922	1.1556	0.2843	*0.0027*
M3GP	0.9922	—	0.6884	0.8486	*0.0490*
M4GP	1.1556	0.6884	—	0.6599	*0.0124*
M4GP-E	0.2843	0.8486	0.6599	—	0.7852
SLUG	**0.0027**	**0.0490**	**0.0124**	0.7852	—

PRK	GP	M3GP	M4GP	M4GP-E	SLUG
GP	—	*0.0000*	0.7658	0.0627	*0.0441*
M3GP	**0.0000**	—	0.0597	0.4882	0.1015
M4GP	0.7658	0.0597	—	0.8317	0.7966
M4GP-E	0.0627	0.4882	0.8317	—	0.7220
SLUG	**0.0441**	0.1015	0.7966	0.7220	—

ION	GP	M3GP	M4GP	M4GP-E	SLUG
GP	—	0.3297	2.2914	1.6595	0.3457
M3GP	0.3297	—	0.9674	0.3734	0.9704
M4GP	2.2914	0.9674	—	1.7345	1.1531
M4GP-E	1.6595	0.3734	1.7345	—	0.3530
SLUG	0.3457	0.9704	1.1531	0.3530	—

SON	GP	M3GP	M4GP	M4GP-E	SLUG
GP	—	*0.0000*	*0.0010*	*0.0154*	0.3311
M3GP	**0.0000**	—	0.2653	0.3660	**0.0232**
M4GP	**0.0010**	0.2653	—	0.6295	0.2414
M4GP-E	**0.0154**	0.3660	0.6295	—	0.4270
SLUG	0.3311	*0.0232*	0.2414	0.4270	—

Table 5. Holm corrected p-values using Kruskal-Wallis for the gametes problems.

10_005	GP	M3GP	M4GP	M4GP-E	SLUG
GP	—	1.1109	0.0673	**0.0107**	0.9234
M3GP	1.1109	—	0.8996	0.2383	0.5035
M4GP	0.0673	0.8996	—	0.7535	*0.0433*
M4GP-E	*0.0107*	0.2383	0.7535	—	*0.0117*
SLUG	0.9234	0.5035	**0.0433**	**0.0117**	—

10_01	GP	M3GP	M4GP	M4GP-E	SLUG
GP	—	0.2129	0.0701	**0.0194**	0.5589
M3GP	0.2129	—	1.3918	0.5030	0.4243
M4GP	0.0701	1.3918	—	0.9644	0.1472
M4GP-E	*0.0194*	0.5030	0.9644	—	*0.0241*
SLUG	0.5589	0.4243	0.1472	**0.0241**	—

10_02	GP	M3GP	M4GP	M4GP-E	SLUG
GP	—	**0.0000**	**0.0017**	**0.0021**	1.4559
M3GP	*0.0000*	—	0.7673	0.4077	*0.0001*
M4GP	*0.0017*	0.7673	—	0.8110	*0.0020*
M4GP-E	*0.0021*	0.4077	0.8110	—	*0.0109*
SLUG	1.4559	**0.0001**	**0.0020**	**0.0109**	—

10_04	GP	M3GP	M4GP	M4GP-E	SLUG
GP	—	*0.0000*	*0.0000*	*0.0000*	*0.0000*
M3GP	**0.0000**	—	0.8301	0.1779	1.3901
M4GP	**0.0000**	0.8301	—	0.3962	1.3785
M4GP-E	**0.0000**	0.1779	0.3962	—	0.0975
SLUG	**0.0000**	1.3901	1.3785	0.0975	—

100_005	GP	M3GP	M4GP	M4GP-E	SLUG
GP	—	0.3509	0.3911	*0.0012*	*0.0001*
M3GP	0.3509	—	*0.0213*	*0.0000*	*0.0000*
M4GP	0.3911	**0.0213**	—	*0.0000*	*0.0000*
M4GP-E	**0.0012**	**0.0000**	**0.0000**	—	0.3713
SLUG	**0.0001**	**0.0000**	**0.0000**	0.3713	—

100_01	GP	M3GP	M4GP	M4GP-E	SLUG
GP	—	0.9823	*0.0148*	*0.0060*	*0.0001*
M3GP	0.9823	—	0.0599	*0.0072*	*0.0000*
M4GP	**0.0148**	0.0599	—	0.2304	*0.0006*
M4GP-E	**0.0060**	**0.0072**	0.2304	—	0.0796
SLUG	**0.0001**	**0.0000**	**0.0006**	0.0796	—

100_02	GP	M3GP	M4GP	M4GP-E	SLUG
GP	—	0.3664	*0.0040*	*0.0000*	*0.0000*
M3GP	0.3664	—	0.2970	*0.0001*	*0.0000*
M4GP	**0.0040**	0.2970	—	*0.0005*	*0.0000*
M4GP-E	**0.0000**	**0.0001**	**0.0005**	—	*0.0365*
SLUG	**0.0000**	**0.0000**	**0.0000**	**0.0365**	—

100_04	GP	M3GP	M4GP	M4GP-E	SLUG
GP	—	0.0728	*0.0000*	*0.0000*	*0.0000*
M3GP	0.0728	—	0.0539	*0.0000*	*0.0001*
M4GP	**0.0000**	0.0539	—	*0.0015*	*0.0084*
M4GP-E	**0.0000**	**0.0000**	**0.0015**	—	0.1311
SLUG	**0.0000**	**0.0001**	**0.0084**	0.1311	—

1000_02	GP	M3GP	M4GP	M4GP-E	SLUG	SLUG-L
GP	—	0.4726	0.5740	*0.0000*	*0.0000*	*0.0000*
M3GP	0.4726	—	0.0718	*0.0000*	*0.0000*	*0.0000*
M4GP	0.5740	0.0718	—	*0.0000*	*0.0000*	*0.0000*
M4GP-E	**0.0000**	**0.0000**	**0.0000**	—	*0.0003*	*0.0000*
SLUG	**0.0000**	**0.0000**	**0.0000**	**0.0003**	—	0.9174
SLUG-L	**0.0000**	**0.0000**	**0.0000**	**0.0000**	0.9174	—

1000_04	GP	M3GP	M4GP	M4GP-E	SLUG	SLUG-L
GP	—	0.3560	0.0676	*0.0000*	*0.0000*	*0.0000*
M3GP	0.3560	—	0.2035	*0.0000*	*0.0000*	*0.0000*
M4GP	0.0676	0.2035	—	*0.0000*	*0.0000*	*0.0000*
M4GP-E	**0.0000**	**0.0000**	**0.0000**	—	**0.0114**	0.1650
SLUG	**0.0000**	**0.0000**	**0.0000**	*0.0114*	—	0.3762
SLUG-L	**0.0000**	**0.0000**	**0.0000**	0.1650	0.3762	—