Genetic Programming and Evolvable Machines (2019) 20:127–137
https://doi.org/10.1007/s10710-018-9338-z

LETTER

Automated discovery of test statistics using genetic programming

Jason H. Moore[1] · Randal S. Olson[1] · Yong Chen[1] · Moshe Sipper[1,2]

Received: 15 June 2018 / Revised: 2 September 2018 / Published online: 10 October 2018

Abstract
The process of developing new test statistics is laborious, requiring the manual development and evaluation of mathematical functions that satisfy several theoretical properties. Automating this process, hitherto not done, would greatly accelerate the discovery of much-needed, new test statistics. This automation is a challenging problem because it requires the discovery method to know something about the desirable properties of a good test statistic in addition to having an engine that can develop and explore candidate mathematical solutions with an intuitive representation. In this paper we describe a genetic programming-based system for the automated discovery of new test statistics. Specifically, our system was able to discover test statistics as powerful as the *t* test for comparing sample means from two distributions with equal variances.

Keywords Genetic programming · Statistics · Optimization · *t* test

1 Introduction

Test statistics such as the *t* test and Chi square test of independence summarize experimental or observational data and, when coupled with a decision rule, can be used as evidence to accept or reject a null hypothesis. Although we use a number of different statistics in daily practice they are all limited by assumptions that must be true for their distribution and derived *p* value to be valid. For example, the two-sample *t* test assumes that the variances between the two groups of data are equal. Deriving a *t* test that produces a valid *p* value when the variances are not equal is an unsolved problem

Jason H. Moore and Randal S. Olson contributed equally to this paper.

✉ Moshe Sipper
sipper@gmail.com

1 Institute for Biomedical Informatics, Perelman School of Medicine, University of Pennsylvania, Philadelphia, PA 19104, USA

2 Department of Computer Science, Ben-Gurion University, Beer-Sheva 84105, Israel

in statistics. The number of unsolved problems is growing due to the increasing diversity of research questions that coincide with the increasing size and complexity of data that are enabled by technology. Unfortunately, the process of developing new test statistics is laborious, requiring the manual development and evaluation of mathematical functions that satisfy theoretical properties such as being unbiased, having low variance (efficient), and capturing the relevant information contained in the data (sufficient) [1]. Determining these properties requires the test statistic to have a known probability density function (PDF) that can be evaluated using differential and integral calculus. Despite the many advances in applied statistics and data science, the development process for new test statistics has not yet been automated using computational methods. Automation would greatly accelerate the discovery of new test statistics that are very much needed in the era of big data. This would in turn accelerate scientific discovery and research translation.

Artificial intelligence (AI) has shown promise for automating several human-driven processes, such as object detection in images, speech recognition, game playing, financial trading, and more. The deep neural networks that are often in current-day use are purely pattern recognition engines that differentiate signal from noise in big data. The development of test statistics is a more challenging problem because it requires the optimization system to know something about the desirable properties of a good test statistic in addition to having an engine that can develop and explore candidate mathematical solutions with an intuitive representation. Very few fundamental mathematical problems have been tackled by automated systems to date (one prominent example being [2], who applied an evolutionary algorithm to the discovery of particular algebraic terms).

The goal of the present study was to develop an evolutionary system for the automated discovery of new test statistics. There were three important challenges that needed to be addressed to accomplish this objective. First, we needed an engine for generating mathematical candidates for test statistics, in our case using available array-based operators in a modern programming language with a data structure that is easy for the computer to manipulate. Second, we needed a set of evaluation criteria that are general enough to allow the computer to generate innovative solutions while specific enough to satisfy human statistical objectives without directing the computer to predetermined outcomes. Third, we needed a system that could tinker with candidate test statistics as a mathematician would, by making small changes and by interchanging functional modules to create new solutions. We describe here a genetic programming (GP) solution to this problem.

We believe the significance of the results presented in this letter lies beyond their preliminary nature, in that we tap into an entirely novel application field for evolutionary computation.

2 Methods

2.1 Representation of test statistics

The raw materials or building blocks of new test statistics include mathematical functions such as addition or square root, constants such as the sample size, and array-based operations on the data such as sum, maximum, mean, median, and standard deviation. We chose to develop our discovery system in the popular programming language Python, giving us access to an extensive library of scalar and array functions in NumPy and SciPy. NumPy is a scientific computing library in Python, focused on n-dimensional arrays of data while SciPy includes a wide range of modules for mathematics, science, and engineering. Functions in these libraries serve as the building blocks for candidate test statistics. We chose to represent candidate solutions in the computer using binary expression trees where the nodes of the tree are mathematical functions or operators and the leaves of the tree represent experimental or observational variables.

2.2 Objective criteria

The key to implementing an evolutionary approach to the discovery of test statistics is to articulate the objective criteria that are important to human statisticians. We chose here to focus on four very general criteria to allow the system to be innovative. First, we want a test statistic to have a low rate of type I errors. These occur when the null hypothesis is true but is rejected. Second, we want a statistic to have good power under the alternative hypothesis (power is the probability that the test correctly rejects the null hypothesis when a specific alternative hypothesis is true). Third, we want a statistic to be invariant to the scale of the data, making it generalizable. Finally, we would like a statistic to be as simple as possible, thus making it easy to understand and implement. Our target discovery system must be able to consider these four criteria simultaneously in a multi-objective framework. These general criteria allow the system to be innovative in a way that might not occur if more specific mathematical or statistical constraints—such as requiring a probability density function—were specified. This process opens the door to the discovery of novel parametric and nonparametric test statistics.

2.3 Genetic programming

The final piece of an evolutionary system for automated discovery of test statistics is a framework for making changes in candidate test statistics with selection of increasingly better solutions. We chose to implement genetic programming (GP) as our evolutionary framework because it is ideally suited to working with binary expression trees and their manipulation through a mutation operator for changing nodes and leaves in the trees and a recombination operator that simply swaps modules of mathematical functions and variables between different candidate solutions with fitness-based selection and learning [3–5]. This evolutionary approach is also ideally suited for multi-objective Pareto optimization that can balance the four fitness criteria we outlined above. Here, we combined the type I error rate and the power additively into a single objective. This yielded three objectives used in the Pareto optimization. Genetic programming has been extensively applied to classification and regression problems but not to the design and discovery of new test statistics. We used the Distributed Evolutionary Algorithms in Python (DEAP) software package for this study because it is open-source and has the desired functionality including Pareto optimization [6].

2.4 The problem

Our goal was to apply this GP system to discover test statistics as powerful as the *t* test, a commonly used statistic for comparing sample means from two distributions with equal variances. Equation (1) below shows the two-sample *t* test statistic. Here, x bar is the sample mean, S^2 is the variance, and N is the sample size. The subscripts on each indicate the sample number. Thus, the numerator of the *t* test is the difference of the sample means and the denominator is the standard error of the differences between the means. This yields a t distribution under the null hypothesis whose shape is determined be degrees of freedom equal to the two sample sizes added together minus two.

In these experiments, we asked whether the GP system is capable of discovering test statistics similar in power to the *t* test (Eq. 1) when presented with our general evaluation criteria.

$$t = \frac{\bar{x}_1 - \bar{x}_2}{\sqrt{\frac{S_1^2}{N_1} + \frac{S_2^2}{N_2}}} \tag{1}$$

To answer this question, we first simulated data drawn from two normal distributions, each with different means (0 and 1, 0 and 2, or 0 and 4) but the same variances (standard deviations of 1, 2, or 4, respectively). A total of 30 data sets with sample size of n = 100 were simulated for each set of means. These data are used for the evaluation criteria to represent data consistent with the alternate hypothesis of a difference in means. Next, we permuted each of these 30 data sets to create pairs of distributions consistent with the null hypothesis that the data were drawn from the same distribution with equal means and variances. Finally, we simulated data under the alternate hypothesis such that as the difference in the means increased (0 and 10 or 0 and 100) the equal variances also increased (10 or 100, respectively). A total of 30 datasets were simulated for each set of means and variances. These data were used for the third criterion to evaluate the scale invariance. Finally, we implemented an evaluation criterion that encourages smaller models, to incentivize the GP system to explore smaller models while at the same time being able to create diversity by considering larger models.

2.5 Experimental design

To quantify the objective criteria in Sect. 2.2, we used the following fitness function. When we evaluate an individual, we provide the aforementioned pairs of distributions to the evolved test statistic to generate the test statistic scores for the pairs. Next, we compute a Gaussian kernel density estimate (KDE) of the test statistics from the null distribution pairs (the null distribution is the probability distribution of the test statistic when the null hypothesis is true). The KDE allows us to measure the probability of a test statistic value appearing in the null distribution.

For the first objective (low false positive rate), we take the evolved test statistic values computed from the null distributions across each set of 30 data sets and measure the probability of them occurring in the null distribution. In this case, higher probabilities are considered better because it entails that the null test statistic values are distributed together around a single value.

For the second objective (high power), we take the evolved test statistic values computed from the distribution pairs with differences in means and measure the probability of them occurring in the null distribution. In this case, lower probabilities are considered better because it entails that the test statistic values, when there is a difference in means, fall outside the null distribution. We note that we combined objectives one and two into a single fitness component to limit the multi-objective search space, as the objectives are highly related.

For the third objective (scale invariance), we used the evolved test statistic values from the distribution pairs with means of 0 and 1, 0 and 10, and 0 and 100. As these distribution pairs are the exact same but with a multiplier of 1, 10, and 100, respectively, a test statistic that is scale invariant should produce the exact same test statistic values for these distributions. Thus, for this objective we considered lower sums of differences between the test statistic pairs to be better.

For the fourth objective (simplicity), we used the number of primitives in the GP tree as the measure of complexity. The number of primitives in the GP tree directly correlates to the complexity of the function; thus, GP trees with fewer primitives are considered better.

Using the GP system with these evaluation data and criteria, we ran 30 unique replicate runs (i.e., with different random seeds) with a population size of 1000 candidate test statistics for 1000 GP generations. We saved the entire Pareto front of test statistics at the end of every replicate run and manually inspected every test statistic on the final Pareto fronts. Each test statistic on the final Pareto front from one of the 30 independent runs represents a tradeoff between the different objective criteria according to NSGA2 [7]. The details of the GP parameter settings are provided in Table 1.

3 Results

Across *all* 30 replicate runs, the GP system discovered test statistics that *had a fitness that was equal to or better than the t test*. Figure 1 shows the binary expression tree for the two-sample *t* test. The left side of the tree is the numerator representing

Table 1 GP parameter settings

Parameter	Value
Population size	1000
Initialization method	50%: Randomly generate full trees with leaf depth of 4 50%: Randomly generate partial trees with a minimum leaf depth of 1 and maximum leaf depth of 4
Function set	Array-wise operations: sqrt, square, abs, mean, median, min, max, std, var, size, sum, standard error, add float Float operations: add, subtract, multiply, divide, square, sqrt, abs Array + float operations: subtract float, multiply float, divide float
Terminal set	x_1 and x_2 (the two sample distributions), x_1 bar and x_2 bar (means), v_1 and v_2 (variances), N_1 and N_2 (samples sizes)
Generations	1000
Selection	Select top 1000 individuals according to NSGA2 Pareto ranking
Crossover rate	50%, 1-point crossover
Mutation rate	50%, replace random sub-tree with random sub-tree with leaf depth of 2

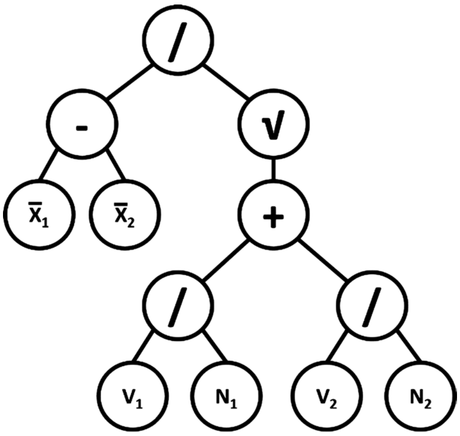

Fig. 1 The two-sample *t* test equation represented as a binary expression tree. The vector of sample values is represented by X, sample means by X bar, variances by V, and sample sizes by N for samples one and two

the difference in the sample means. The right side of the tree is the denominator representing the standard error of the means across the two samples. We present this here to allow a direct comparison with the binary expression trees for three different test statistics discovered by the GP system (Fig. 2). Note that the test statistics discovered by GP include many of the same functions but are mathematically simpler. This tendency to simplicity is likely due to the complexity objective in the fitness function.

Figure 3 shows scatterplots of the test statistic values for the *t* test and those generated by one of the Pareto-optimal GP-based solutions across the simulated data used in the fitness function. Note the linear relationship suggesting that the newly discovered test statistics captures much of the information provided by the *t* test. This was true across all the solutions generated by GP. Thus, the GP system

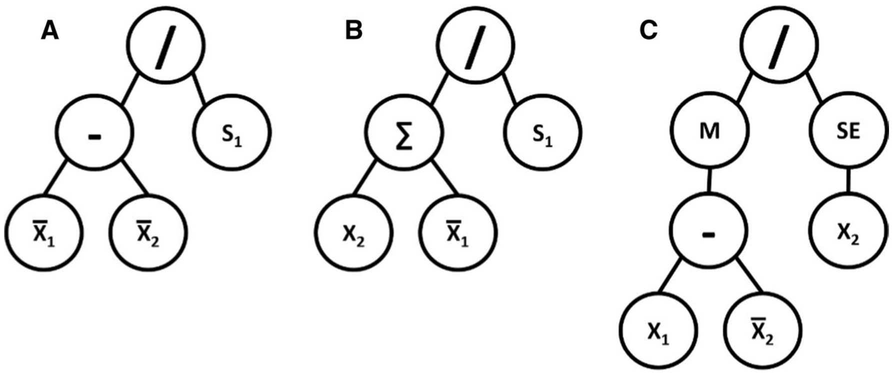

Fig. 2 Three GP-generated test statistics represented as binary expression trees. The vector of sample values is represented by X, sample means by X bar, standard deviations by S, standard error by SE, and median by M

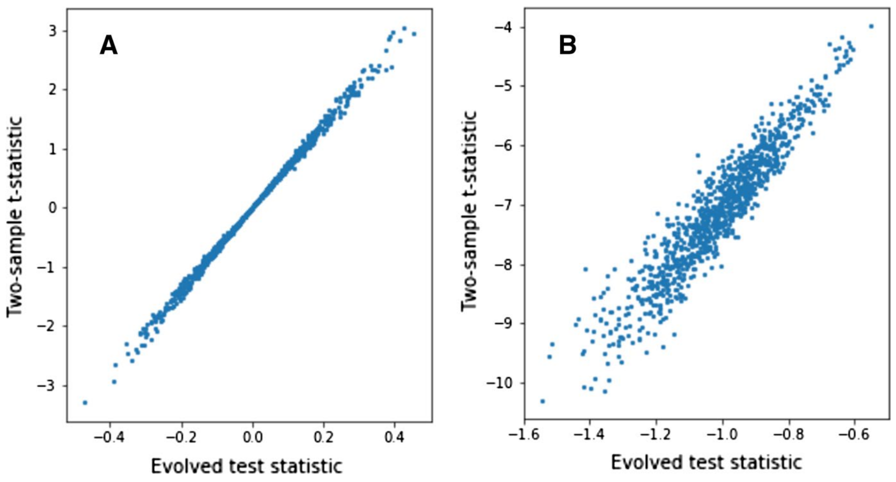

Fig. 3 Scatterplots of the linear relationship of the values of the evolved test statistic from Fig. 2a with the values from the *t* test applied to data simulated under the null hypothesis (**a**) and the alternate hypothesis (**b**)

is finding approximations of the *t* test with equal or greater fitness based on our fitness objectives.

Figure 4 shows the distribution of values for the evolved test statistic from Fig. 2a under the null hypothesis (orange) and alternate hypotheses (blue and green). Note that the test statistic is centered at zero under the null hypothesis. This is the same as the *t* test. The test statistic values are negative or positive depending on whether mean one is greater or less than mean two in the simulation. Both are considered in our fitness function.

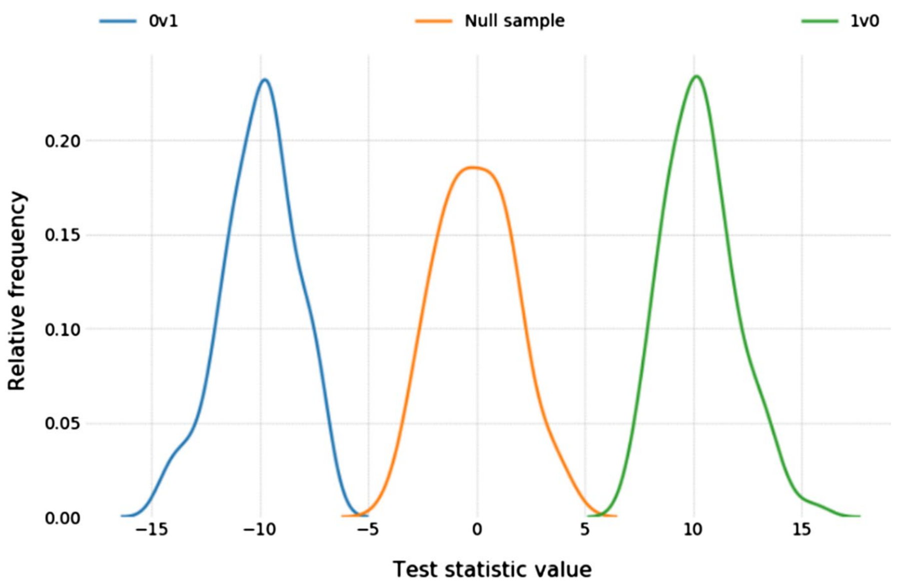

Fig. 4 Smoothed histograms summarizing the distribution of evolved test statistic values under the null hypothesis (orange) and alternate hypotheses (blue and green) (Color figure online)

Figure 5 shows the distribution of values for the evolved test statistic from Fig. 2a under the alternate hypotheses (blue) across means of 0 and 1, 0 and 10, and 0 and 100. Note that the center of the distributions is approximately the same regardless

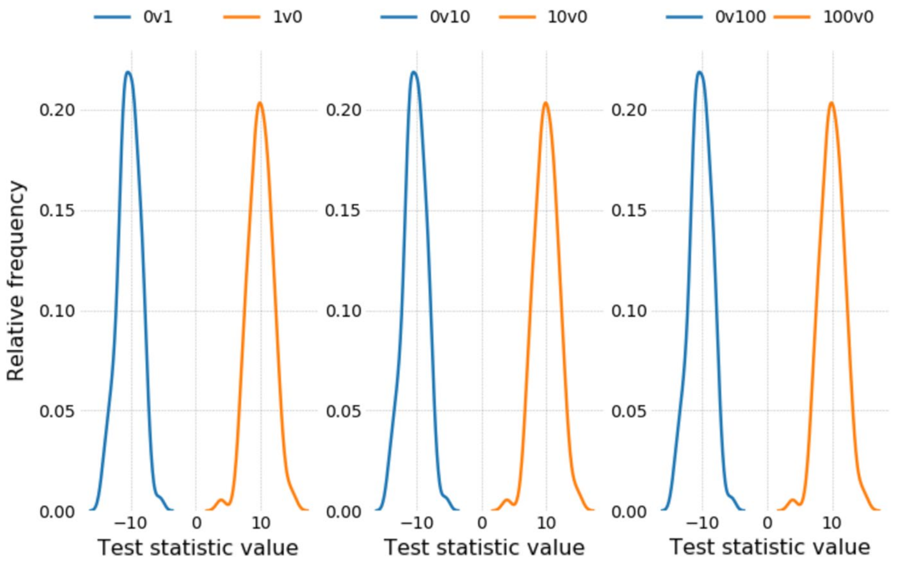

Fig. 5 Smoothed histograms summarizing the distribution of evolved test statistic values under the null hypothesis (orange) and alternate hypotheses (blue) (Color figure online)

of the difference in means. This satisfies the scale-invariance objective of the fitness function—a desirable property of all test statistics.

These results indicate that GP routinely generates test statistics for the comparison of means from two distributions with equal variances that have low type I error, good power, scale-invariance, and that are simple. Further, the evolved test statistics resemble the *t* test suggesting that the GP system finds functions that capture much of the same information.

4 Discussion and concluding remarks

We have introduced here a computational framework for the automated discovery of test statistics using genetic programming. Our approach starts with a list of scalar and vector functions such as arithmetic operators, square root, mean, median, and standard deviation that are often used in the manual construction of test statistics. We represented the candidate test statistics as binary expression trees and then used GP as the discovery engine. The key to this framework is the fitness function that is more complex than many typical GP methods. We specified that good test statistics should have a low false positive rate (i.e., type I error), a high power, scale-invariance, and should be as simple as possible. These four objectives were implemented in a Pareto optimization framework for evaluation and selection within the GP system. We then applied this framework to the discovery of test statistics for comparing sample means from distributions with equal variances. The evaluation of the type I error and power of each candidate model require the use of data simulated under the null hypothesis of no differences between the sample means and the alternate hypothesis that the means are different. Our results show that in each of the replicate runs the GP system was able to generate test statistics that had fitness values as good as or better than the *t* test that is the widely accepted and applied solution to this problem. Further, our GP-generated test statistics were linearly related to the *t* test and tended to be much simpler. We conclude that GP is suited to the automatic generation of test statistics and should be extended and applied to unsolved test statistic problems in statistics.

It is important to note that our GP implementation did not find the exact equation for the *t* test on the final Pareto front across the 30 replicate runs. There are several possible reasons for this. First, we used complexity as an objective. This put pressure on the GP system to find simple solutions that contained many of the same functions as the *t* test but perhaps not all. Second, we purposefully left the GP fitness function general to encourage innovation. In other words, we did not want to guide it toward finding the *t* test. There is a mathematical component to developing test statistics that we did not explore here. When developing a test statistic by hand it is necessary to use differential and integral calculus to evaluate the bias, efficiency, and sufficiency of the functions. We propose that our GP approach be used for discovery with the evaluative calculus steps being saved at the end for fine-tuning of the discovered functions.

There are several limitations of the current study that could be explored in future work. First, the fitness function is computationally complex given that simulated data is used to evaluate the type I error and power of the solutions. This complexity will only increase as more complex test statistics are tackled. There may be mathematical solutions that could speed up the fitness function. Second, our fitness function with its four objectives was developed to approximate the process that humans use when evaluating a test statistic. There may be other important objectives to consider. For example, the efficiency of a test statistic is an important theoretical objective that is a measure of the number of observations in a sample that are needed to maximize its power. Of course, balancing more than three or four objectives in a Pareto optimization framework can be challenging. Finally, our approach uses simulated data in the fitness function. This means that the distribution of the data must be known. However, this does not preclude the development of nonparametric tests.

The discovery of test statistics for comparing two means is just the start. There are many unsolved problems in statistics that could benefit from our GP framework. For example, Cox [8] outlines 22 unsolved problems from 1984. Some of these remain challenges today. For example, in cancer genomic studies using DNA methylation data, it has been found that the methylation measurements of the cancer and normal groups could differ not only in means but also in variances [9, 10]. To account for these data characteristics, we have previously developed an extension of the *t* test using U-statistics theory [11] and mixture models [12]. These extensions were not trivial and could have been greatly accelerated using genetic programming. In addition, there are a number of interesting GP methods that have been developed that might be useful to enhance our strategy. One such approach is layered GP that adds new functions to candidate solutions instead of swapping them between expression trees using crossover [13]. This is appealing because test statistics are often modular in nature. For example, the ability of the GP to add a useful function such as standard error to a statistic could accelerate the discovery process. This could even be done in a probabilistic way to prioritize commonly used modules from parametric statistics.

The need for new statistics is exploding as new technologies give us new data with unique characteristics that yield new scientific questions. There is no question that data and experimental designs are changing at a rate that exceeds that of mathematical statisticians. In fact, the number of statisticians that actively develop new test statistics is decreasing as trainees opt for more exciting and lucrative fields such as data science, where the demand for the application of statistical methods and machine learning is exploding. This is the right time to explore artificial intelligence methods for assisting statisticians with the automated development of test statistics.

Acknowledgements This work was supported by National Institutes of Health (USA) Grants LM012601, AI116794, and DK112217. We would like to thank the reviewers for the thoughtful suggestions.

References

1. G. Casella, R.L. Berger, *Statistical Inference* (Duxbury Press, Pacific Grove, 2001)
2. L. Spector, D.M. Clark, I. Lindsay, B. Barr, J. Klein. Genetic programming for finite algebras, in *Proceedings of the 10th Annual Conference on Genetic and Evolutionary Computation*. (ACM, New York, 2008), pp. 1291–1298

3. J.R. Koza, *Genetic Programming: On the Programming of Computers by Means of Natural Selection* (MIT Press, Cambridge, 1992)
4. R. Poli, W.B. Langdon, N.F. McPhee, *A Field Guide to Genetic Programming* (Lulu Enterprises, UK Ltd, 2008)
5. M. Sipper, W. Fu, K. Ahuja, J.H. Moore, Investigating the parameter space of evolutionary algorithms. BioData Min. **11**, 2 (2018)
6. F.-A. Fortin, F.-M.D. Rainville, M.-A. Gardner, M. Parizeau, C. Gagné, DEAP: evolutionary algorithms made easy. J. Mach. Learn. Res. **13**, 2171–2175 (2012)
7. K. Deb, A. Pratap, S. Agarwal, T. Meyarivan, A fast and elitist multiobjective genetic algorithm: NSGA-II. IEEE Trans. Evol. Comput. **6**, 182–197 (2002)
8. D.R. Cox, Present position and potential developments: some personal views: design of experiments and regression. J. R. Stat. Soc. Ser. A Gen. **147**, 306–315 (1984)
9. K. Gervin, M. Hammerø, H.E. Akselsen, R. Moe, H. Nygård, I. Brandt, H.K. Gjessing, J.R. Harris, D.E. Undlien, R. Lyle, Extensive variation and low heritability of DNA methylation identified in a twin study. Genome Res. **21**, 1813–1821 (2011)
10. K.D. Hansen, W. Timp, H.C. Bravo, S. Sabunciyan, B. Langmead, O.G. McDonald, B. Wen, H. Wu, Y. Liu, D. Diep, E. Briem, K. Zhang, R.A. Irizarry, A.P. Feinberg, Increased methylation variation in epigenetic domains across cancer types. Nat. Genet. **43**, 768–775 (2011)
11. Y. Chen, Y. Ning, C. Hong, S. Wang, Semiparametric tests for identifying differentially methylated loci with case-control designs using Illumina arrays. Genet. Epidemiol. **38**, 42–50 (2014)
12. C. Hong, Y. Chen, Y. Ning, S. Wang, H. Wu, R.J. Carroll, Plemt: a novel pseudolikelihood based em test for homogeneity in generalized exponential tilt mixture models. J. Am. Stat. Assoc. **112**, 1393–1404 (2017)
13. D. Medernach, J. Fitzgerald, R.M.A Azad, C. Ryan. A new wave: a dynamic approach to genetic programming, in *Proceedings of the Genetic and Evolutionary Computation Conference 2016*. (ACM, New York, 2016), pp. 757–764