Genetic Programming and Evolvable Machines manuscript No.
(will be inserted by the editor)

Unimodal optimization using a genetic-programming-based method with periodic boundary conditions

Rogério C. B. L. Póvoa · Adriano S. Koshiyama · Douglas M. Dias · Patrícia L. Souza · Bruno A. C. Horta

Received: date / Accepted: date

Abstract This article describes a new genetic-programming-based optimization method using a multi-gene approach along with a niching strategy and periodic domain constraints. The method is referred to as Niching MG-PMA, where MG refers to multi-gene and PMA to parameter mapping approach. Although it was designed to be a multimodal optimization method, recent tests have revealed its suitability for unimodal optimization. The definition of Niching MG-PMA is provided in a detailed fashion, along with an in-depth explanation of two novelties in our implementation: the feedback of initial parameters and the domain constraints using periodic boundary conditions. These ideas can be potentially useful in terms of other optimization techniques. The method is tested on the basis of the CEC'2015 benchmark functions. Statistical analysis shows that Niching MG-PMA performs similarly to the winners of the competition even without any parametrization towards the benchmark, indicating that the method is robust and applicable to a wide range of problems.

R. C. B. L. Povóa
Department of Electrical Engineering, Pontifical Catholic University of Rio de Janeiro, Rio de Janeiro, 22451 900, Brazil
Apparat AS, Bergen, Hordaland, 5058, Norway
E-mail: rogerio.povoa@apparat.no

A. S. Koshiyama
Department of Computer Science, University College London, Gower Street, London, WC1E 6BT, UK
E-mail: a.koshiyama@cs.ucl.ac.uk
ORCID: http://orcid.org/0000-0001-7536-1503

D. M. Dias
Department of Electronics and Telecommunications, Faculty of Engineering, Rio de Janeiro State University, Rio de Janeiro, 20550 900, Brazil
E-mail: douglas.dias@uerj.br
ORCID: http://orcid.org/0000-0002-1783-6352

P. L. Souza
LabSem/CETUC, Pontifical Catholic University of Rio de Janeiro, Rio de Janeiro, 22451 900, Brazil
E-mail: plustoza@cetuc.puc-rio.br

B. A. C. Horta
Institute of Chemistry, Federal University of Rio de Janeiro, Rio de Janeiro, 21941 901, Brazil
E-mail: bruno@iq.ufrj.br
ORCID: http://orcid.org/0000-0002-3952-1474

Keywords Optimization · Evolutionary computation · Genetic programming · Niching methods · Periodic boundary conditions · Parameter mapping approach

1 Introduction

Genetic programming (GP) [22] is a type of evolutionary algorithm in which the solution space is defined by a set of computer programs that evolve in a competitive way through the application of bio-inspired operators and selection criteria. Commonly applied for symbolic regression [29], classification [10] and search based software engineering [14, 13, 23, 24, 50], GP has interesting approaches such as Genetic Semantic Programming with Local Search (GSGP-LS) [6], a hybrid method combining geometric semantic [34] and a local strategy to optimize the parameters that characterize the individuals at each generation. Nevertheless, GP has rather few implementations and benchmark studies in the field of numerical optimization.

Despite the popularity and efficiency of several Evolutionary Algorithms (EAs) for numerical optimization, such as Genetic Algorithms (GAs) [16], Particle Swarm Optimization (PSO) [9], and Differential Evolution (DE) [46], GP allows for the interpretability and understanding of the solution of a problem through the analysis of its programs, which enables new insights and discoveries. Some promising GP-based optimization methods have been reported in this field: (i) Cartesian Genetic Programming (CGP) [48, 33]; (ii) Embedded Cartesian Genetic Programming (ECGP) [48]; (iii) Positional Cartesian Genetic Programming (PCGP) [49], (iv) Parameter Mapping Approach (PMA) [40]; (v) Optimization by Genetic Programming (OGP) [21]; and (vi) Multimodal Genetic Programming (MMGP) [51]. Although these methods are promising, only a few comparative analysis on benchmark functions have been performed and they are starting to be used in some different applications like machine learning [41, 45] and digital circuits [3].

PMA and Multi-Gene-PMA (MG-PMA as we call it in this paper, but the authors originally used the acronym OGP) are optimization methods that use GP to map initial parameters into optimal input parameters for an objective function. In 2013, Koshiyama [21] showed the performance of MG-PMA for certain benchmark functions without the need to specify the domain of those functions. This advantage allows the use of such method for solving problems with unknown domain. In 2018, a preliminary study was performed by us [39], applying a niching MG-PMA method for multimodal optimization (MMO) problems. More specifically, the 20 multimodal functions from the 2017 benchmark of the IEEE CEC Special Session on Niching Methods for Multimodal Optimization [26] were used to compare Niching MG-PMA with the best methods of the CEC competition.

In summary, our results revealed no significant difference between Niching MG-PMA and other considered methods. If one takes into account that the other methods were highly tuned to perform well on this benchmark [26], the results indicate the potential of Niching MG-PMA for general and real-world application problems. Interestingly, our previous work [39] (only about multimodal problems) also showed that this method performs best for the first five functions (p1 to p5) of the benchmark, which are the functions with the lowest number of solutions. This led us to consider that Niching MG-PMA, in principle designed for multimodal problems, could perform even better on unimodal optimization problems. Therefore, the objective of the present work is twofold: (*i*) provide a formal and more complete definition of the method that was only superficially introduced by us in 2018 [39]; and (*ii*) evaluate the performance of the method on unimodal optimization problems using the well known unimodal benchmark functions of the CEC 2015 competition [28].

This paper is organized as follows: Section 2 presents the niching multi-gene parameter mapping approach (Niching MG-PMA) and the methods used for its development (MG-PMA and the feedback of the parameter values); Section 3 shows our recent normalization technique used to follow domain constraints based on the periodic boundary conditions (PBC); Section 4 shows the set of unimodal benchmark functions and the configurations used in the experiments of this work; Section 5 presents the results; and Section 6 presents closing remarks and suggestions for future work.

2 Multi-gene genetic programming for numerical optimization

2.1 Parameter Mapping Approach

PMA [40] is a GP-based method designed for parameter optimization and tuning (POT) problems. POT methods are used to perform the tuning or the optimization of these parameters, for which at least one initial set of parameters is required. These methods then produce a set of adapted parameter values. In PMA, a population of GP programs represents a set of possible mapping functions that transforms initial parameters into adapted ones. These adapted values are submitted to an objective (evaluation) function, resulting in a value that defines the quality of the solution. GP iteratively improves such functions by mapping initial parameters until (sub)optimal values are found. Figure 1 presents this POT approach by PMA, where a GP program maps an array of initial parameter values (r_1, r_2, r_3) into an array of adapted ones (P_1, P_2, P_3), considering a problem for which the solution is the point $(0,0,0)$. Over the generations (from G_0 to G_n), the GP program evolves to a better solution; the nearest point to $(0,0,0)$.

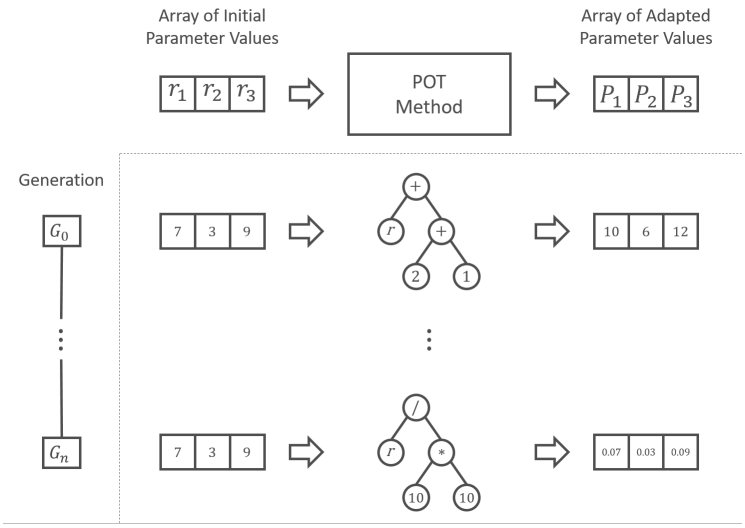

Fig. 1 Transformation performed by PMA in a typical POT method.

The variables of these mapping functions are represented by the initial parameters. As variables of a mathematical function can assume more than one value, the representation of these parameters cannot be only an array as in Figure 1. The initial parameters are represented in PMA by a matrix **M** containing n rows and m columns, where n is the dimensionality of the objective function and m is a user-defined number at the beginning of the optimization. Each mapping function generated by GP uses a random column as its variables and the constants are made available by GP constants (also called ephemeral random constant by Koza [22]). These constants are initialized using a random number generator at generation zero and remain unaltered during the evolutionary process. Figure 2 illustrates this process. Results found by Pujol and Poli [40] suggested that PMA is as effective as well established methods in the literature, such as GA, PSO and DE.

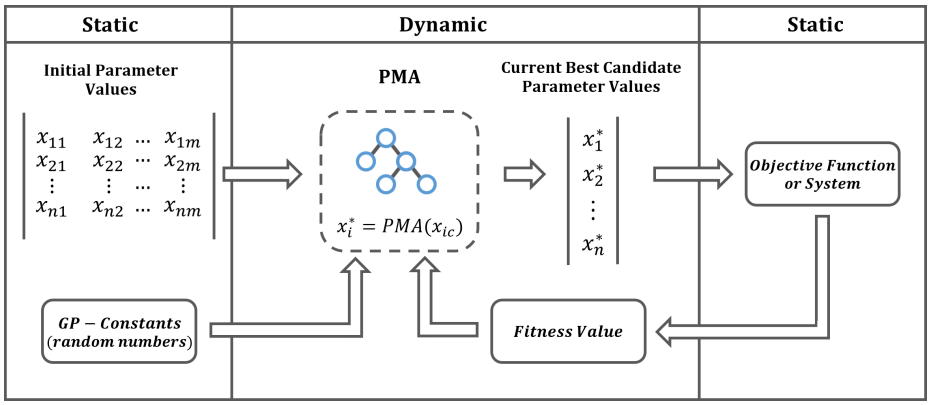

Fig. 2 Schematic representation of the PMA procedure. The aim of PMA is to find a program that maps the initial parameter values (x_{ic}) into the optimal parameter values (x_i^*) for a given system (c is a column of initial parameter values, selected by each tree/program). The current best candidate parameters correspond to adapted values from the mapping of the initial parameters. The fitness value indicates how well a program performs this mapping. As GP, PMA also utilizes random numbers as constants in its individuals. During the evolution process, the initial parameter values, the GP-constants and the objective function are always the same (static) while PMA functions, candidates parameters values and the corresponding fitness values change (dynamic).

2.2 Multi-gene parameter mapping approach

In PMA, the parameters to be optimized are not individually encoded, but mapped by a single function. The method works fairly well for trivial problems, as shown by Pujol and Poli [40]. For example, consider $X_i^* = 0$ for all i, the optimal parameters for $f(.)$. Thus, if an individual encodes the function $\text{PMA}(x_i) = x_i - x_i$, the optimization problem is solved. However, when the set of optimal solutions is not trivial, PMA tends to face more difficulties. Consider a second example in which $X_i^* = 1 + r_i$, where $r_i \sim \text{Unif}(-1,1)$. In this case, it is more difficult to find a $\text{PMA}(X_i)$ function that solves the optimization problem. This is discussed in details by Pujol and Poli [40].

In view of further improving GP-based optimization algorithms, this work presents a generalization of the PMA method, where a set of equations is responsible for certain pa-

rameters rather than a single one being responsible for all. This is achieved through multi-gene genetic programming (MGGP) [15], a generalization of the canonical GP, in which most components are similar, but MGGP denotes an individual as a complex tree structure, also called genes. Each tree structure can be considered as a partial solution to the problem, and the final output results from the linear combination of them. From this point on, the generalization of PMA, introduced in the present work, will be referred to as multi-gene parameter mapping approach (MG-PMA). Preliminary studies with early implementations of the multi-gene approach to PMA were performed. The results were compared to those obtained by GA, PSO, DE and PMA [21] and applied to optimize the parameters of a spiking neural network for clustering problems [45]. There the method was referred to as optimization by genetic programming (OGP). However, since it was in fact a generalization of PMA, the acronym MG-PMA seems more appropriate as it acknowledges the previous PMA method. This work recommends the usage of OGP for referring to the emerging field of GP-based optimization methods.

Figure 3 depicts the main differences between a single tree and a multi-tree approach. While PMA represents all parameters to be optimized by a single function (Figure 3.a), MG-PMA separates the optimization problem into k parts, where k is the number of equations of MGGP. For example, for $k = 4$, MG-PMA assigns to each equation the responsibility of searching for a quarter of the solution. Figure 3.b shows the function found by MG-PMA to solve a hypothetical optimization problem. In the limiting case, when k equals the number n of parameters to be optimized ($k = n$), MG-PMA assigns to each parameter an equation that receives an arbitrary value and returns it transformed by the decoded equation in the individual. By contrast, when $k = 1$, MG-PMA becomes PMA. Note that, in the limiting case ($k = n$), it is not mandatory to include initial parameters values (x_{ij}, as in Figure 2.13) in the terminal set of MGGP, but the GP constants are still required. Since each function is only responsible for one parameter of the objective function, a mapped value of one parameter can be represented by a result of arithmetical combinations of constants. In this case, MG-PMA searches a function of constants for each parameter.

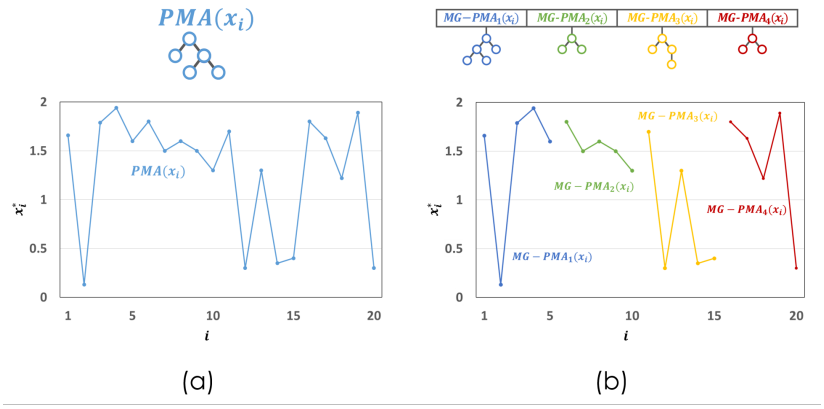

Fig. 3 Representation of individuals in PMA and MG-PMA. (a) PMA represents all parameters to be optimized by a single function. (b) MG-PMA partitions the optimization problem into k parts. In this example, for $k = 4$, MG-PMA assigns to each of the four equations the responsibility of optimizing a quarter of the problem. The optimal parameters are represented by points on the graphs.

In general, the pseudo-code of MG-PMA for any objective function is described as in Algorithm 1. Similarly to PMA, MG-PMA starts by setting the initial parameters matrix $\mathbf{M}_{n_x m}$. Then, MG-PMA specifies the population parameters (e.g., population size and number of generations) and genetic operators (e.g., crossover rate and mutation rate), which are commonly defined in a GP execution. Once they are established, an additional parameter k defines the number of trees encoded in an individual. The function $SplitInitialParameters$ (see Algorithm 2) uses the initial parameters and k as input to define how the initial parameters should be partitioned into the number of trees in MG-PMA. Partitioning is done so that each tree receives the same number of initial parameters. If it is not possible to evenly divide the parameters, the first trees receive more parameters. For example, for $n = 5$ and $k = 5$ each tree receives an initial parameter, while for $n = 5$ and $k = 3$, the first two trees receive two initial parameters while the last tree receives only one initial parameter.

```
Data: Objective Function
Output: Optimal Parameters
- InitialParameterValues ← Initialize();
- GPConstants ← Initialize();
- Set parameters for MG-PMA;
- Set k ;
- t ← 0 ;
- SplitInitialParameters(InitialParameters, k);
- Start Population(t);
- Fit ← Evaluate(Population(t));
while Stop criterion ≠ FALSE do
    t ← t + 1;
    Population(t) ← Select(Population(t − 1));
    Population(t) ← ApplyOperators(Population(t));
    Fit ← Evaluate(Population(t));
end
```

Algorithm 1: Pseudocode of MG-PMA for a generic objective function.

```
Input  : Initial Parameters and Parameter k
- MaxParametersPerTree ← ceil(CountRows(InitialParameters)/k);
- Parts[MaxParametersPerTree, k] ← Create segments with k parts;
- NumberParametersParts[k] ← All elements equal to 0;
- CountParameters ← 0;
- IndexParameters ← 1;
while CountParameters < CountRows(InitialParameters) do
    for CurrentPart ← 1 to k do
        if CountParameters < CountRows(InitialParameters) then
            NumberParametersParts[CurrentPart] ← NumberParametersParts[CurrentPart] + 1;
            CountParameters ← CountParameters + 1;
        else
            break;
        end
    end
end
for CurrentPart ← 1 to k do
    for Index ← 1 to NumberParametersParts[CurrentPart] do
        Parts(Index, CurrentPart) ← InitialParamenters(IndexParameters);
        IndexParameters ← IndexParameters + 1;
    end
end
```

Algorithm 2: Pseudocode of the `SplitInitialParameters` algorithm: division of the initial parameters into k trees.

A population is then randomly created, consisting of individuals with k functions each. Each individual selects a random column of the initial parameters matrix so as to simulate the "variables" of the mathematical equations which, in turn, are evaluated by the objective function. After the evaluation of the entire population, it is verified whether the stopping criterion is satisfied. If so, then the current population is returned, otherwise the algorithm enters in a loop that is only interrupted when the stop criterion is reached. During the loop, three operations are performed: selection, application of operators and evaluation. The first one chooses, based on some heuristic methods (e.g., roulette and tournament) [38], the entities of the following population. Subsequently, the function $ApplyOperators$ (see Algorithm 3) is called to apply elitism and one of these four genetic operators: (i) mutation; (ii) direct reproduction; (iii) high-level crossover (it exchanges the genes of two individuals to generate two new individuals [39]); (iv) or low-level crossover (similar to canonical GP [22]). The frequency of use of these genetic operators is determined by the set of parameters at the beginning of MG-PMA. Finally, this new population is re-evaluated and once again it is verified whether the stopping criterion has been reached or not. If yes, then the current population is returned, otherwise the whole process is repeated.

```
Input  : Population
Output: New Population
- NewPopulation ← ApplyElitism(Population);
while Size(NewPopulation) ≠ Size(Population) do
    Parent1 ← SelectOneIndividual(Population);
    Flip ← Random Number Between 0 and 1;
    if Flip < Mutation Probability then
        NewPopulation ← ApplyMutation(Parent1);
    else if Flip < Sum of Mutation and Direct Reproduction Probabilities then
        NewPopulation ← ApplyDirectReproduction(Parent1);
    else
        Parent2 ← SelectOneIndividual(Population);
        FlipCrossover ← Random Number Between 0 and 1;
        if FlipCrossover < High Level Crossover Probability then
            NewPopulation ← ApplyHighLevelCrossover(Parent1,Parent2);
        else
            NewPopulation ← ApplyLowLevelCrossover(Parent1,Parent2);
        end
    end
end
```

Algorithm 3: Pseudocode of `ApplyOperators` algorithm: MG-PMA genetic operators.

2.3 Feedback of the parameter values

One of the most damaging problems in overall optimization is the premature convergence of the objective function to a local minimum (or maximum) [32]. Several factors can cause this problem: lack of genetic diversity, poorly adjusted control parameters, poorly established stopping criterion and complexity of a problem for a particular optimization method. Since MG-PMA works by mapping initial values into adapted ones, it is natural that for some problems, this mapping may be very complex or even impossible. The MG-PMA approach with feedback aims not only at mapping initially fixed values but also at updating these initial values dynamically. As GP trees represent functions, using updated values can result into a completely different solution, which favors to escape from a local optimum.

This update of the initial values is made from time to time during the evolution process through a control parameter or when the optimization converges to a certain value for many generations (determined by the user also by a control parameter). When one of these criteria is met, one column of the initial values matrix is replaced by the adjusted values, generated by the best individual of the current generation. This approach was developed hoping that larger steps in the search space are obtained, so as to avoid premature convergence allowing more randomization to the optimization process. We have performed preliminary tests which showed this improvement by pointing to better solutions without premature convergence. However, not sufficient statistical tests were performed to prove this hypothesis. Figure 4 illustrates the MG-PMA with feedback process.

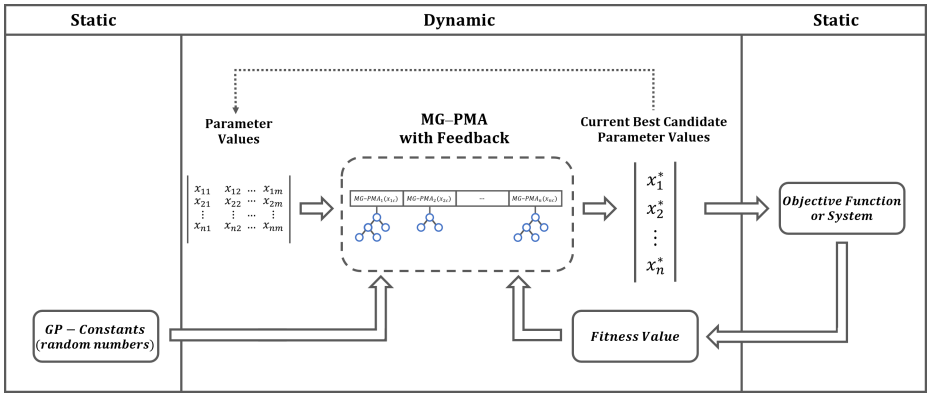

Fig. 4 Schematic representation of MG-PMA with feedback procedure. MG-PMA with feedback is a linear combination of programs that maps the initial parameter values (x_{ic}) into the optimal parameter values (x_i^*) for a given system (where i varies from 1 to n and c varies from 1 to m). The current best candidate parameters correspond to adapted values from the mapping of the parameter values. The fitness value indicates how well a program performs this mapping. As GP, MG-PMA with feedback also utilizes random numbers as constants in its individuals. From time to time the parameter values are updated to those of the best candidate. During the evolution process, the GP-constants and the object function are always the same (static) while the parameter values, MG-PMA functions, candidates parameters values and the corresponding fitness values change (dynamic).

This replacement of the parameter values by adapted values is performed from the first column ($c = 1$ in Figure 4) to the last column ($c = m$ in Figure 4). After replacing all the columns, the MG-PMA with feedback repeats the process of choosing to update the initial values from the first to the last column.

2.4 Niching multi-gene parameter mapping approach

Niching multi-gene parameter mapping approach (Niching MG-PMA) is an MMO method that aims at exploring the search space while maintaining population diversity, using MG-PMA with feedback, clearing procedure [37] and local optimization.

Following the concepts of the clearing procedure, Niching MG-PMA uses the idea of dominant individuals in subpopulations [37]. Each subpopulation is formed by individuals that are within the clearing radius and only winners are selected for the application of the

genetic operators. In Niching MG-PMA, all individuals who are not winners are removed from the population and new individuals are generated from the winners coming from the application of the genetic operators.

Another important characteristic of Niching MG-PMA is that it is a memetic algorithm that combines global with local optimization. The objective of the local optimization is to refine the global search since after applying the clearing method only the winner of a given region of the search space will be in that area. Therefore, local minimization/maximization should be sufficient to bring that winner close to a local (or global) minimum/maximum. The frequency of the local optimization is an input parameter but, based on our previous work, there is an indication that the best results are obtained when local optimization is applied at every generation.

We performed preliminary tests in an attempt to find the global optimum of the benchmark functions using only local optimization algorithms. This approach often resulted in the stagnation of the optimization by finding local optima, and, then, it was necessary to combine algorithms of local and global optimization so that the global optimum could be found.

2.4.1 Niching MG-PMA Procedure

Algorithm 4 presents the pseudocode of Niching MG-PMA. Similarly to MG-PMA, Niching MG-PMA starts with the initialization of the initial parameters and GP constants, and the configuration of the Niching MG-PMA parameters, which are: the number of trees (k), the frequency of updating initial parameters ($feedbackFrequency$) and the frequency of local optimization ($localOptimizationFrequency$).

```
Data: Objective Function
Output: Multiple Optimal/Sub-Optimal Solutions
- InitialParameterValues ← Initialize();
- GPConstants ← Initialize();
- Set parameters for Niching MG-PMA;
- Set k, feedbackFrequency, localOptimizationFrequency;
- t ← 0 ;
- AllIndividuals ← an empty vector;
- SplitInitialParameters(InitialParameters, k);
- Start Population(t);
- Fit ← Evaluate(Population(t));
while Stop criterion ≠ FALSE do
    t ← t + 1;
    Population(t) ← Niching(Population(t − 1));
    if mod(t,localOptimizationFrequency) then
        Population(t) ← LocalOptimization(Population(t));
    end
    Population(t) ← Select(Population(t));
    AllIndividuals and Population(t) ← ApplyOperators(Population(t));
    Fit ← Evaluate(Population(t));
    if mod(t, feedbackFrequency) then
        InitialParameterValues ← Feedback(Population(t));
    end
end
- MultipleSolutions ← CheckSolutions(AllIndividuals);
```

Algorithm 4: Pseudocode of Niching MG-PMA for a generic objective function.

The allocation of the initial parameters to the trees is performed in the same way as in MG-PMA (function `SplitInitialParameters` presented in Algorithm 2). The population

is initialized and the individuals are selected according to the function `Select` for application of the genetic operators (function `ApplyOperators` in the Algorithm 3), generating new individuals that are also evaluated.

Until the stopping criterion is met, the `Niching` function identifies the winners and removes individuals that are within a distance shorter than the clearing radius (relative to dominant individuals) from the population. If the current generation number is a multiple of the *localOptimizationFrequency* parameter, the function `LocalOpt` locally optimizes the position of the winners in the search space using the quasi-Newton algorithm and the gradients are estimated by finite differences [19]. The selection process and the application of the genetic operators are then performed, generating new individuals for the following population. Niching MG-PMA evaluates these individuals and, if the current generation is a multiple of the $feedbackFrequency$ parameter, the `Feedback` function updates the initial parameters, replacing one of the columns of the initial parameters matrix by the best individual of the current generation.

Once the stopping criterion is met, the function `CheckSolutions` searches for optimal and sub-optimal solutions among all the winners found during the optimization process.

3 Domain Constraints with Periodic Boundary Conditions

Metaheuristics such as GA, PSO and DE follow the domain constraints of the objective function that are imposed by limiting the search range of each parameter entering the objective function. GP and its variations are not subjected to this limitation because the parameters are generated by computer programs. In PMA [40], Pujol and Poli used the following linear transformation :

$$ADAPT = LOWER + OUT \times (UPPER - LOWER) \tag{1}$$

where *ADAPT* is the adapted value of the parameter, *LOWER* is the lower limit of the search range, *UPPER* is the upper limit of the search range and *OUT* is the output of the GP tree [40]. However, this transformation does not work for every type of function, as, for instance, when the function is periodic with more than one solution.

Our preliminary work [39] introduced another way to normalize the output of the GP trees by applying periodic boundary conditions (PBC) in the parameter space. In this work, we present PBC in more detail. This approach was inspired by the geometric PBC method used by the molecular modelling community [25] in order to remove surface effects and to minimize finite-size effects in molecular simulations. The domain-constraint technique proposed here is a box-constrained, considering that the objective function is inside a "box" that is replicated in all dimensions. In molecular simulations the box is in fact a 3D construction, but the dimensionality of the box, in the present context, is given by the dimensionality of the objective function. A parameter value generated by a GP tree may be outside the central box (i.e. in one of its periodic images). The PBC method translates that value, placing it back into the central box through multiple sums or subtractions of the box size. Equation 2 presents the transformation for a value outside the limits of the objective function:

$$EDGE = \begin{cases} LOWER\ , OUT < LOWER \\ UPPER\ , OUT > UPPER \end{cases}$$

$$N_{\mathrm{BOX}} = \frac{|OUT - EDGE|}{SIZE_{\mathrm{BOX}}}$$

$$ADAPT = \begin{cases} UPPER - (N_{\text{BOX}} - \lfloor N_{\text{BOX}} \rfloor) \times SIZE_{\text{BOX}} & , OUT < LOWER \\ LOWER + (N_{\text{BOX}} - \lfloor N_{\text{BOX}} \rfloor) \times SIZE_{\text{BOX}} & , OUT > UPPER \end{cases} \quad (2)$$

where *ADAPT* is the adapted value of the parameter, *EDGE* is the nearest edge from the output of the GP tree, *LOWER* is the lower limit of the search range, *UPPER* is the upper limit of the search range, *OUT* is the output of the GP tree, N_{BOX} is the number of boxes away from the domain, and $SIZE_{\text{BOX}}$ is the box size along this dimension.

Figure 5 shows an example of the normalization based on the PBC applied to a point outside of the domain constraints of a multimodal function. Figure 5.a shows this point outside the objective function domain (green dot, position $(16,0)$) and in Figure 5.b it is put back into the domain (black dot, position $(-4,0)$) from the replicated images of this function.

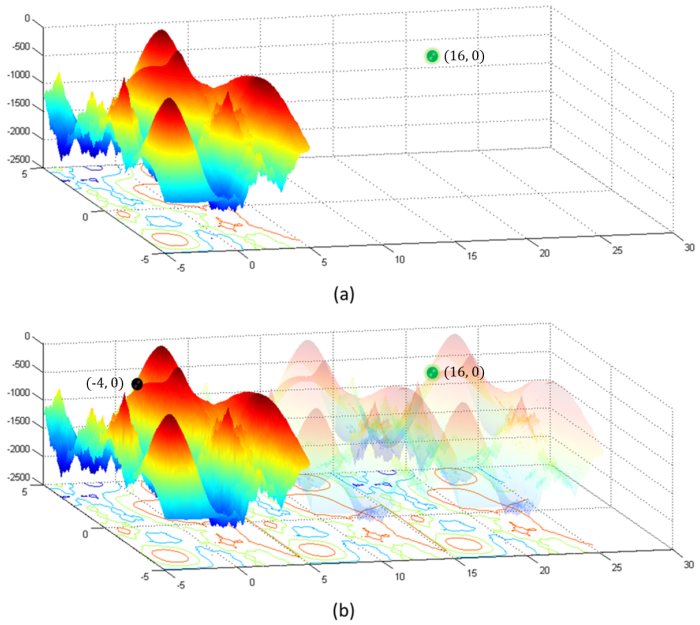

Fig. 5 Example of the normalization, based on the PBC applied to an output (green dot, position $(16,0)$) of a GP tree outside of the domain constraints. The linear transformation replicates images of the box and identifies where the outside dot should be (black dot, position $(-4,0)$). The parameter range determines the size of the box.

As the domain constraint with PBC is applied to the output value of the GP trees, this normalization is directly applied at the input values of the objective function. Thus, the domain constraint with PBC is a box-constrained and it is limited to be used in this type of problem with different metaheuristic methods.

4 Benchmark and Configuration

As mentioned in Section 1, the results of the preliminary investigation [39] using Niching MG-PMA and PBC for multimodal functions were compared to those obtained with well-established parameter optimization algorithms showing, within the statistical significance analysis, a similar performance to that of them all. Importantly, Niching MG-PMA outperformed the other methods when applied to the functions with fewer solutions (p1 to p5). Based on this result, we claim that this method could be competitive for unimodal optimization. The main purpose of this work is to verify this claim.

Niching MG-PMA is tested using unimodal benchmark functions of the CEC'2015 competition [28], presented in Table 1. It is highly important to emphasize that the method and configuration tested herein correspond to exactly the same method and configuration of the previous work [39], denoted therein by Niching MG-PMA L1, where L1 implies that local minimization was performed at every generation. Not a single parameter was tuned to improve the performance on the present benchmark (see Table 2). The search for the minimum was interrupted when the fitness of the best individual was lower than 10^{-8}, as established by the competition [28].

Table 1 Benchmark functions proposed for CEC'2015 competition. The table presents the name of each function, the number of the parameters to be optimized and the global minimum.

Abbreviation	Function Name	Number of Parameters	Global Minimum
c_1	Rotated High Conditioned Elliptic Function	10	100
c_2	Rotated Cigar Function	10	200
c_3	Shifted and Rotated Ackley's Function	10	300
c_4	Shifted and Rotated Rastrigin's Function	10	400
c_5	Shifted and Rotated Schwefel's Function	10	500
c_6	Hybrid Function 1 ($N = 3$)	10	600
c_7	Hybrid Function 2 ($N = 4$)	10	700
c_8	Hybrid Function 3 ($N = 5$)	10	800
c_9	Composition Function 1 ($N = 3$)	10	900
c_{10}	Composition Function 2 ($N = 3$)	10	1,000
c_{11}	Composition Function 3 ($N = 5$)	10	1,100
c_{12}	Composition Function 4 ($N = 5$)	10	1,200
c_{13}	Composition Function 5 ($N = 5$)	10	1,300
c_{14}	Composition Function 6 ($N = 7$)	10	1,400
c_{15}	Composition Function 7 ($N = 10$)	10	1,500

For this test, the population was equal to 100 and the tournament size equal to 2 for all runs. As established by the competition, 51 independent runs were performed for each benchmark function and the maximum number of evaluations before time-out was 10^5. The number of iterations for each optimization has been set for a maximum of 100 evaluations. Since the number of evaluations is fixed and local optimization is applied to each generation, the number of generations varies in each simulation, without exceeding the pre-established value of number of evaluations. The number of evaluations for the first generation is equal to the population size. For the other generations, the number of evaluations in each generation can be calculated by adding the number of local minimization steps for each winner and the number of new individuals generated from the winners. The total number of evaluations is found by summing the number of evaluations of all generations. The optimization process

Table 2 General configuration of MG-PMA.

Parameters	Values
Decimal GP constants rate	90%
Integer GP constants rate	0%
GP constants range	$[-1,+1]$
Low-level crossover rate	65%
Mutation rate	30%
Direct reproduction rate	5%
High-level crossover rate	50%
Elitism rate	1%
Lexicographic pressure [30]	Yes

runs while this total is less than the parameter that corresponds to the maximum number of evaluations.

For initialization and modification of GP trees, we used the mathematical operators *plus*, *minus*, *times* and *protected division*. The use of protected division is a GP open issue [35]. Thus, we opted for an empirical choice based on the idea of introns, where non-effective operations do not affect the optimization process [36,5]: for $a = b/c$, if $c = 0$ then $a = 0$. The remaining configuration are detailed in Table 2.

Niching MG-PMA was implemented using elements of GPTIPS 2 (Genetic Programming & Symbolic Data Mining Platform for MATLAB) library of MGGP [44]. This library allows the use of more than one mutation operator during optimization. As described in Table 2, the probability of applying the mutation operator in the experiments was 30% (value defined through preliminary analysis presenting low sensitivity). From the moment at which the mutation operator was used, it was possible to perform one of the three types of mutation [44] with the following probabilities: ordinary sub-tree mutation (90%) [15,44]; switch of an input terminal to another one randomly selected (5%); and Gaussian perturbation of a randomly selected constant (5%, with the standard deviation of the Gaussian used equal to 10%) – the parameter values for these three types of mutation are the default values of the GPTIPS 2 library [44]. The initial population was generated using the ramped half-and-half method [22] and the PBC normalization was used during the evolution process.

Different approaches can be found in the literature to determine the value of the niche radius [27]: (i) use a single uniform niche radius [20,8]; (ii) use a variable niche radius [47, 4,11]; and (iii) avoid specifying the niche radius [47,31]. The variable niche radius approach was used in this work. Equation 3 presents the calculation of the niche radius R_g, which is the same approach used in preliminary tests for Niching MG-PMA [39]. The value of R_g is changed according to the population, avoiding very large or very small sizes.

$$R_g = \frac{median(AllDistances_g)}{10} \tag{3}$$

where R_g is the niche radius for generation g and $AllDistances_g$ are all the distances between the individuals of the population in generation g.

Niching MG-PMA was compared with the winners of the CEC'2015 competition: SPS-L-SHADE-EIG [12], MVMO-SH [2], LSHADE-ND [43], DEsPA [42].

5 Results

Table 3 and Figure 6 present the mean best fitness values (MBFVs) for Niching MG-PMA and the best algorithms of the competition CEC'2015. It is possible to notice that SPS-L-SHADE-EIG (the winner of the competition [12]) was able to find the global solution for 8 out of the 15 functions considered. Both LSHADE-ND and MVMO-SH found 6 solutions, whereas both Niching MG-PMA and DEsPA found 5. In the cases, where the methods do not find the global solution, the relative deviations of the solution with respect to the global optima can also be analyzed and the methods can be compared in terms of the statistical significance of the results. In general, Niching MG-PMA performed similarly to the winning algorithms of the competition. For the function c_5 and c_{11}, Niching MG-PMA presented inferior results (greater MBFV). For the functions c_4, c_7-c_8 and c_{10} Niching MG-PMA was also slightly worse, showing a small but still noticeable difference in terms of MBVF. Considering the functions c_3, c_9, c_{13}, c_{14} and c_{15}, Niching MG-PMA performed better than DEsPA and for c_{14} better than LSHADE-ND. It is important to highlight that these methods were highly parametrized [2,12,42,43] to provide good performance, specifically considering the rules and nature of the functions of the CEC'2015 benchmark, whereas Niching MG-PMA was not. This is confirmed by the fact that we used the same configuration as in our preliminary investigation [39].

Table 3 Mean of the best fitness values (MBFV) for all benchmark functions (c_1-c_{15}) of the competition CEC'2015 found by SPS-L-SHADE-EIG, DEsPA, LSHADE-ND, MVMO-SH and Niching MG-PMA algorithms.

Benchmark Function	SPS-L-SHADE-EIG	DEsPA	LSHADE-ND	MVMO-SH	Niching MG-PMA
c_1	1.000×10^2	1.000×10^2	1.000×10^2	1.000×10^2	1.000×10^2
c_2	2.000×10^2	2.000×10^2	2.000×10^2	2.000×10^2	2.000×10^2
c_3	3.200×10^2	1.970×10^3	3.200×10^2	3.200×10^2	3.200×10^2
c_4	4.010×10^2	4.036×10^2	4.030×10^2	4.020×10^2	4.150×10^2
c_5	5.152×10^2	5.519×10^2	5.070×10^2	5.118×10^2	7.38×10^2
c_6	6.000×10^2	6.016×10^2	6.004×10^2	6.014×10^2	6.310×10^2
c_7	7.000×10^2	7.003×10^2	7.000×10^2	7.000×10^2	7.010×10^2
c_8	8.000×10^2	8.002×10^2	8.003×10^2	8.003×10^2	8.160×10^2
c_9	1.000×10^3	1.006×10^3	1.000×10^3	1.000×10^3	1.000×10^3
c_{10}	1.217×10^3	1.008×10^3	1.217×10^3	1.217×10^3	1.270×10^3
c_{11}	1.100×10^3	1.195×10^3	1.102×10^3	1.104×10^3	1.330×10^3
c_{12}	1.300×10^3	1.301×10^3	1.301×10^3	1.301×10^3	1.300×10^3
c_{13}	1.300×10^3	1.318×10^3	1.300×10^3	1.300×10^3	1.300×10^3
c_{14}	1.500×10^3	1.709×10^3	4.335×10^3	1.500×10^3	1.500×10^3
c_{15}	1.600×10^3	1.705×10^3	1.600×10^3	1.600×10^3	1.600×10^3

Table 4 shows the results of Aligned Friedman's [7] and Iman-Davenport's [18] tests, along with the Holm's procedure [17] based on the results listed in Table 3. These tests were carried out using the KEEL software [1]. It can be seen that the average rank of SPS-L-SHADE-EIG was the lowest (30.767) of all used algorithms.

Typically, in most statistical analysis, the value of 0.05 for the p-value is used as the threshold for significance [7]. If the p-value is less than 0.05, the null hypothesis that there is no difference between the configurations tested is rejected and it is concluded that there is a significant difference between these configurations. If the p-values are greater than 0.05, it is not possible to conclude that there is a significant difference.

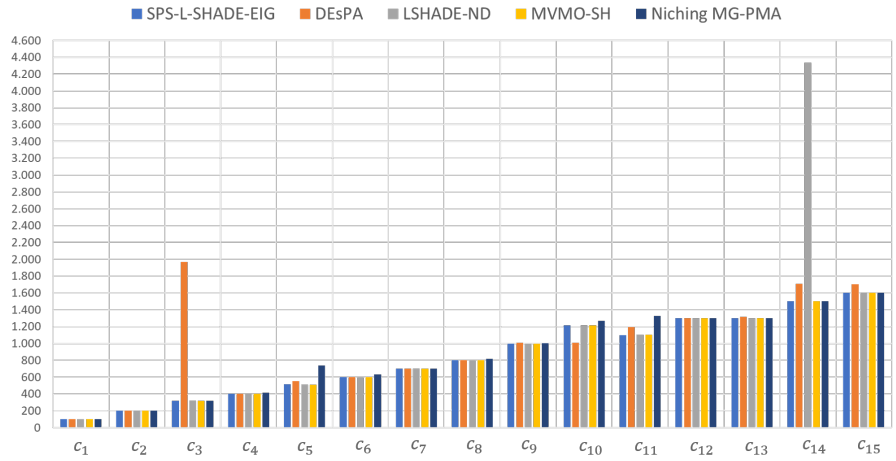

Fig. 6 Chart of the mean of the best fitness values (MBFV) for all benchmark functions (c_1-c_{15}) of the competition CEC'2015 found by SPS-L-SHADE-EIG, DEsPA, LSHADE-ND, MVMO-SH and Niching MG-PMA algorithms.

Table 4 Aligned Friedman's and Iman-Davenport's tests, and Holm's procedure for pairwise comparison between SPS-L-SHADE-EIG, DEsPA, LSHADE-ND, MVMO-SH and Niching MG-PMA algorithms. The reference rank (R_0) corresponds to the rank of the best algorithm ($i = 0$); in this case SPS-L-SHADE-EIG.

i	Algorithm	Rank
4	Niching MG-PMA	47.400
3	DEsPA	44.400
2	LSHADE-ND	36.167
1	MVMO-SH	31.267
0	SPS-L-SHADE-EIG	30.767

Test	p-value
Aligned Friedman	0.022

Algorithm	$z = (R_0 - R_i)/SE$	p-value	Holm	Reject?
Niching MG-PMA	2.090	0.037	0.013	No
DEsPA	1.713	0.087	0.017	No
LSHADE-ND	0.679	0.497	0.025	No
MVMO-SH	0.063	0.950	0.050	No

Both tests (Aligned Friedmans and Iman-Davenports) found a p-value less than 0.05 (p-value = 0.022), so the hypothesis that there is no difference between the compared algorithms is rejected. As the average rank of SPS-L-SHADE-EIG is the lowest, it was selected as the control model for the Holm's test. In Table 4, z represents the test statistic for Holms test, where the ranks R_i of all models are compared to that of the control model R_0, normalized by the standard error SE. This is used to compute the p-value, which is compared to the Holm level. If the p-value is lower than that level, the hypothesis of equality of ranks is rejected. In that way, it is possible to conclude that there is no significant difference between Niching MG-PMA and the best algorithms of the competition CEC'2015.

6 Conclusions

In a recent conference paper, we introduced a new GP-based optimization method using a multi-gene approach along with a niching strategy and periodic domain constraints. The method was referred to as Niching MG-PMA and was tested in the context of the 20 multimodal functions from the 2017 benchmark of the IEEE CEC Special Session on Niching Methods for Multimodal Optimization [26]. To our delight, without any specific optimization targeting the benchmark, statistical analysis revealed no significant difference between Niching MG-PMA and the considered methods. Also, worth noting was the fact that the method behaved significantly better for functions with a lower number of global optima, potentially indicating that it could be even more suitable for unimodal optimization.

The present work describes the Niching MG-PMA method in a more detailed fashion, providing an in-depth explanation of the feedback and the periodic domain constraint routines. The latter ideas can be potentially useful in terms of other optimization techniques.

In order to verify our claim that Niching MG-PMA could be a promising method for unimodal optimization, the method was tested on the basis of the competition CEC'2015 benchmark functions. Statistical analysis showed that Niching MG-PMA performs similarly as the winners (SPS-L-SHADE-EIG [12], MVMO-SH [2], LSHADE-ND [43] and DEsPA [42]) of the competition even without any specific parametrization to the benchmark, indicating that the method is robust and applicable to a wide range of problems.

These results are very promising, indicating that Niching MG-PMA can be applied to real problems. Moreover, because it is a generalization of PMA [40], its principle is to map starting points to optimal parameters of an objective function. This allows us to recommend its use to problems where suboptimal solutions are already known and can be further optimized (e.g. structural optimization of molecular structures). Besides, Niching MG-PMA is a GP algorithm which allows the interpretation of its solutions. Thus, the mapping of known solutions to optimal solutions can be studied, allowing new insights or discoveries in the application areas. It is important to observe the computational cost, since the the need of evaluation of GP expressions leads to longer Niching MG-PMA execution time, burden of the implicit mapping principle since the development of PMA [40].

Finally, the present investigation evidences the potential of GP in the field of numerical optimization, as well as the possibility of using multimodal strategies in unimodal optimization problems. New possibilities of application and research appear with the results obtained in this work are: (i) the application of Niching MG-PMA to real problems; (ii) the analysis of using protected division as a mathematical operator in GP trees (e.g. other types of definitions as for $a = b/c$, if $c = 0$ then $a = 1$ [22]); and (iii) an in-depth statistical analysis of both the local minimization and the feedback procedure on the method. Despite the surprisingly good results obtained without parameter tuning, demonstrating the robustness of the algorithm with respect to parameters, future work may improve the understanding of Niching MG-PMA's parameter robustness.

Acknowledgements This work was supported by the Brazilian fundations CNPq and by the Rio de Janeiro state agency FAPERJ (grant numbers E-26/203.198/2016 and E-26/010.002420/2016). This work was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Finance Code 001. We also acknowledge the support of Núcleo Avançado de Computação de Alto Desempenho (NACAD/COPPE/UFRJ), and Sistema Nacional de Processamento de Alto Desempenho (SINAPAD).

References

1. Alcalá-Fdez, J., Fernández, A., Luengo, J., Derrac, J., García, S.: KEEL data-mining software tool: Data set repository, integration of algorithms and experimental analysis framework. Multiple-Valued Logic and Soft Computing **17**(2-3), 255–287 (2011). URL http://www.oldcitypublishing.com/journals/mvlsc-home/mvlsc-issue-contents/mvlsc-volume-17-number-2-3-2011/mvlsc-17-2-3-p-255-287/
2. Awad, N., Ali, M.Z., Reynolds, R.G.: A differential evolution algorithm with success-based parameter adaptation for cec2015 learning-based optimization. In: 2015 IEEE Congress on Evolutionary Computation (CEC), pp. 1098–1105 (2015). DOI 10.1109/CEC.2015.7257012
3. Babu, K.S., Balaji, N.: Approximation of digital circuits using cartesian genetic programming. In: 2016 International Conference on Communication and Electronics Systems (ICCES), pp. 1–6 (2016). DOI 10.1109/CESYS.2016.7889978
4. Bird, S., Li, X.: Adaptively choosing niching parameters in a PSO. In: Proceedings of the 8th Annual Conference on Genetic and Evolutionary Computation, GECCO '06, pp. 3–10. ACM, New York, NY, USA (2006). DOI 10.1145/1143997.1143999
5. Brameier, M., Banzhaf, W.: A comparison of linear genetic programming and neural networks in medical data mining. IEEE Transactions on Evolutionary Computation **5**(1), 17–26 (2001). DOI 10.1109/4235.910462
6. Castelli, M., Trujillo, L., Vanneschi, L., Silva, S., Z-Flores, E., Legrand, P.: Geometric semantic genetic programming with local search (2015). DOI 10.1145/2739480.2754795
7. Derrac, J., García, S., Molina, D., Herrera, F.: A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms. Swarm and Evolutionary Computation **1**(1), 3 – 18 (2011). DOI 10.1016/j.swevo.2011.02.002
8. Dick, G.: Automatic identification of the niche radius using spatially-structured clearing methods. In: IEEE Congress on Evolutionary Computation, pp. 1–8 (2010). DOI 10.1109/CEC.2010.5586085
9. Eberhart, R., Kennedy, J.: A new optimizer using particle swarm theory. In: Micro Machine and Human Science, 1995. MHS '95., Proceedings of the Sixth International Symposium on, pp. 39–43 (1995). DOI 10.1109/MHS.1995.494215
10. Espejo, P.G., Ventura, S., Herrera, F.: 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 (2010). DOI 10.1109/TSMCC.2009.2033566
11. Gan, J., Warwick, K.: Dynamic niche clustering: a fuzzy variable radius niching technique for multimodal optimisation in gas. In: Proceedings of the 2001 Congress on Evolutionary Computation (IEEE Cat. No.01TH8546), vol. 1, pp. 215–222 vol. 1 (2001). DOI 10.1109/CEC.2001.934392
12. Guo, S.M., Tsai, J.S.H., Yang, C.C., Hsu, P.H.: A self-optimization approach for l-shade incorporated with eigenvector-based crossover and successful-parent-selecting framework on cec 2015 benchmark set. In: 2015 IEEE Congress on Evolutionary Computation (CEC), pp. 1003–1010 (2015). DOI 10.1109/CEC.2015.7256999
13. Harman, M., Jia, Y., Langdon, W.B., Petke, J., Moghadam, I.H., Yoo, S., Wu, F.: Genetic improvement for adaptive software engineering (keynote). In: Proceedings of the 9th International Symposium on Software Engineering for Adaptive and Self-Managing Systems, SEAMS 2014, pp. 1–4. ACM, New York, NY, USA (2014). DOI 10.1145/2593929.2600116
14. Harman, M., Langdon, W.B., Weimer, W.: Genetic programming for reverse engineering. In: 2013 20th Working Conference on Reverse Engineering (WCRE), pp. 1–10 (2013). DOI 10.1109/WCRE.2013.6671274
15. Hinchliffe, M., Hiden, H., McKay, B., Willis, M., Tham, M., Barton, G.: Modelling chemical process systems using a multi-gene genetic programming algorithm. In: J.R. Koza (ed.) Late Breaking Papers at the Genetic Programming 1996 Conference Stanford University July 28-31, 1996, pp. 56–65. Stanford Bookstore, Stanford University, CA, USA (1996). URL http://www.genetic-programming.org/gp96latebreaking.html. (print)
16. Holland, J., et al.: Adaptation in Natural and Artificial Systems. MIT Press, Cambridge (1975)
17. Holm, S.: A simple sequentially rejective multiple test procedure. Scandinavian Journal of Statistics **6**, 65–70 (1979)
18. Iman, R.L., Davenport, J.M.: Approximations of the critical region of the friedman statistic. Communications in Statistics - Theory and Methods **9**(6), 571–595 (1980). DOI 10.1080/03610928008827904. URL https://doi.org/10.1080/03610928008827904
19. Iott, J., Haftka, R., Adelman, H., Center, L.R.: Selecting Step Sizes in Sensitivity Analysis by Finite Differences. NASA technical memorandum. National Aeronautics and Space Administration, Scientific and Technical Information Branch (1985). URL https://books.google.no/books?id=KggDAAAAIAAJ
20. Jelasity, M., Dombi, J.: GAS, a concept on modeling species in genetic algorithms. Artificial Intelligence **99**(1), 1 – 19 (1998). DOI 10.1016/S0004-3702(97)00071-4

21. Koshiyama, A.S., Dias, D.M., Abs da Cruz, A.V., Pacheco, M.A.C.: Numerical optimization by multi-gene genetic programming. In: Proceedings of the 15th Annual Conference Companion on Genetic and Evolutionary Computation, GECCO '13 Companion, pp. 145–146. ACM, New York, NY, USA (2013). DOI 10.1145/2464576.2464651
22. Koza, J.R.: Genetic Programming: On the Programming of Computers by Means of Natural Selection. MIT Press, Cambridge, MA, USA (1992)
23. Langdon, W.B.: Genetic Improvement of Software for Multiple Objectives, pp. 12–28. Springer International Publishing, Cham (2015). DOI 10.1007/978-3-319-22183-0_2
24. Langdon, W.B., Lam, B.Y.H., Modat, M., Petke, J., Harman, M.: Genetic improvement of gpu software. Genetic Programming and Evolvable Machines **18**(1), 5–44 (2017). DOI 10.1007/s10710-016-9273-9
25. Leach, A.R.: Molecular Modelling: Principles and Applications. Pearson Education. Prentice Hall (2001). URL https://books.google.com.br/books?id=kB7jsbV-uhkC
26. Li, X., Engelbrecht, A., Epitropakis, M.G.: Benchmark functions for cec'2013 special session and competition on niching methods for multimodal function optimization. Tech. rep., Evolutionary Computation and Machine Learning Group, RMIT University, Australia (2013)
27. Li, X., Epitropakis, M.G., Deb, K., Engelbrecht, A.: Seeking multiple solutions: An updated survey on niching methods and their applications. IEEE Transactions on Evolutionary Computation **21**(4), 518–538 (2017). DOI 10.1109/TEVC.2016.2638437
28. Liang, J.J., Qu, B.Y., Suganthan, P.N., Chen, Q.: Problem definitions and evaluation criteria for the cec2015 competition on learning-based real parameter single objective optimization. Tech. Rep. 201411A, Computational Intelligence Laboratory, Zhengzhou University, Zhengzhou, China (2014)
29. Liu, Y., Cheng, Z.L., Xu, J., Yang, J., Wang, Q.W.: Improvement and validation of genetic programming symbolic regression technique of silva and applications in deriving heat transfer correlations. Heat Transfer Engineering **37**(10), 862–874 (2016). DOI 10.1080/01457632.2015.1089745
30. Luke, S., Panait, L.: Lexicographic parsimony pressure. In: Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 2002), pp. 829–836. Morgan Kaufmann Publishers (2002)
31. Mahfoud, S.W.: Crowding and preselection revisited. Parallel Problem Solving from Nature, 2 pp. 27–36 (1992)
32. Michalewicz, Z.: Genetic Algorithms + Data Structures = Evolution Programs (3rd Ed.). Springer-Verlag, London, UK (1996). ISBN 3-540-60676-9
33. Miller, J.F., Mohid, M.: Function optimization using cartesian genetic programming. In: Proceedings of the 15th Annual Conference Companion on Genetic and Evolutionary Computation, GECCO '13 Companion, pp. 147–148. ACM, New York, NY, USA (2013). DOI 10.1145/2464576.2464646
34. Moraglio, A., Krawiec, K., Johnson, C.G.: 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, pp. 21–31. Springer Berlin Heidelberg, Berlin, Heidelberg (2012)
35. Ni, J., Drieberg, R.H., Rockett, P.I.: The use of an analytic quotient operator in genetic programming. IEEE Transactions on Evolutionary Computation **17**(1), 146–152 (2013). DOI 10.1109/TEVC.2012.2195319
36. Nordin, P., Francone, F., Banzhaf, W.: Explicitly defined introns and destructive crossover in genetic programming (1995)
37. Petrowski, A.: A clearing procedure as a niching method for genetic algorithms. In: Proceedings of IEEE International Conference on Evolutionary Computation, pp. 798–803 (1996). DOI 10.1109/ICEC.1996.542703
38. Poli, R., Langdon, W.B., McPhee, N.F.: A field guide to genetic programming. Published via `http://lulu.com` and freely available at `http://www.gp-field-guide.org.uk` (2008). (With contributions by J. R. Koza)
39. Póvoa, R.C.B.L., Koshiyama, A.S., Dias, D.M., Souza, P.L., Horta, B.A.C.: Multi-modal optimization by multi-gene genetic programming. In: 2018 IEEE Congress on Evolutionary Computation (CEC), pp. 1–8 (2018). DOI 10.1109/CEC.2018.8477715
40. Pujol, J.C.F., Poli, R.: Parameter mapping: A genetic programming approach to function optimization. International Journal of Knowledge-Based and Intelligent Engineering Systems **12**(1), 29–45 (2008). DOI 10.3233/KES-2008-12104
41. Pujol, J.C.F., Poli, R.: A new combined crossover operator to evolve the architecture and the weights of neural networks using a dual representation. Tech. rep. (2009). URL http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.37.4921; ftp://ftp.cs.bham.ac.uk/pub/tech-reports/1997/CSRP-97-12.ps.gz
42. Rueda, J.L., Erlich, I.: Testing MVMO on learning-based real-parameter single objective benchmark optimization problems. In: 2015 IEEE Congress on Evolutionary Computation (CEC), pp. 1025–1032 (2015). DOI 10.1109/CEC.2015.7257002

43. Sallam, K.M., Sarker, R.A., Essam, D.L., Elsayed, S.M.: Neurodynamic differential evolution algorithm and solving CEC2015 competition problems. In: 2015 IEEE Congress on Evolutionary Computation (CEC), pp. 1033–1040 (2015). DOI 10.1109/CEC.2015.7257003
44. Searson, D.P.: GPTIPS 2: An open-source software platform for symbolic data mining pp. 551–573 (2015). DOI 10.1007/978-3-319-20883-1_22
45. Silva, M., Koshiyama, A., Vellasco, M., Cataldo, E.: Evolutionary features and parameter optimization of spiking neural networks for unsupervised learning. In: 2014 International Joint Conference on Neural Networks (IJCNN), pp. 2391–2398 (2014). DOI 10.1109/IJCNN.2014.6889566
46. Storn, R., Price, K.: Differential evolution - a simple and efficient adaptive scheme for global optimization over continuous spaces. Tech. rep., International Computer Science Institute, Berkeley (1995)
47. Ursem, R.K.: Multinational evolutionary algorithms. In: Proceedings of the 1999 Congress on Evolutionary Computation-CEC99 (Cat. No. 99TH8406), vol. 3, p. 1640 (1999). DOI 10.1109/CEC.1999.785470
48. Walker, J.A., Miller, J.F.: Solving real-valued optimisation problems using cartesian genetic programming. In: Proceedings of the 9th Annual Conference on Genetic and Evolutionary Computation, GECCO '07, pp. 1724–1730. ACM, New York, NY, USA (2007). DOI 10.1145/1276958.1277295
49. Wilson, D.G., Miller, J.F., Cussat-Blanc, S., Luga, H.: Positional cartesian genetic programming. CoRR **abs/1810.04119** (2018). URL http://arxiv.org/abs/1810.04119
50. Yeboah-Antwi, K., Baudry, B.: Online genetic improvement on the java virtual machine with ecselr. Genetic Programming and Evolvable Machines **18**(1), 83–109 (2017). DOI 10.1007/s10710-016-92784
51. Yoshida, S., Harada, T., Thawonmas, R.: Multimodal genetic programming by using tree structure similarity clustering. In: 2017 IEEE 10th International Workshop on Computational Intelligence and Applications (IWCIA), pp. 85–90 (2017). DOI 10.1109/IWCIA.2017.8203566