Adaptive Bi-objective Genetic Programming for Data-driven System Modeling

Vitoantonio Bevilacqua[1], Nicola Nuzzolese[1],
Ernesto Mininno[2], and Giovanni Iacca[2]

[1] Dipartimento di Ingegneria Elettrica e dell'Informazione, Politecnico di Bari,
via Orabona 4, 70125 Bari, Italy
vitoantonio.bevilacqua@poliba.it
[2] Cyber Dyne S.r.l.
Via Scipione Crisanzio 119, 70123 Bari, Italy

Abstract. We propose in this paper a modification of one of the modern state-of-the-art genetic programming algorithms used for data-driven modeling, namely the Bi-objective Genetic Programming (BioGP). The original method is based on a concurrent minimization of both the training error and complexity of multiple candidate models encoded as Genetic Programming trees. Also, BioGP is empowered by a predator-prey co-evolutionary model where virtual predators are used to suppress solutions (preys) characterized by a poor trade-off error vs complexity. In this work, we incorporate in the original BioGP an adaptive mechanism that automatically tunes the mutation rate, based on a characterization of the current population (in terms of entropy) and on the information that can be extracted from it. We show through numerical experiments on two different datasets from the energy domain that the proposed method, named BioAGP (where "A" stands for "Adaptive"), performs better than the original BioGP, allowing the search to maintain a good diversity level in the population, without affecting the convergence rate.

Keywords: Multi-Objective Evolutionary Algorithms, Adaptive Genetic Programming, Machine Learning, Home Automation, Energy Efficiency.

1 Introduction

Computational intelligence techniques, such as Evolutionary Computation (EC) and Machine Learning (ML), have proven to be powerful yet general-purpose tools in a broad range of optimization applications [6, 7, 26]. One of the most important fields of application of such tools is energy management, with successful examples in complex problems such as energy management in buildings, renewable energy systems, heating, ventilation and air conditioning control methodologies, and forecasting energy consumption [9] [28]. Of special interest in this domain is the forecasting e.g. of the air temperature in a room [31] as a function of both weather parameters (mainly solar radiation and air temperature) and actuator states or manipulated variables (heating, ventilating, cooling), with the subsequent use of these mid/long-range prediction models for a more efficient temperature control, both in terms of regulation and energy consumption.

An example of application of these predictive models is efficient building design, where the computation of the heating load and the cooling load is required to determine the specifications of the heating and cooling equipment needed to maintain comfortable indoor air conditions. However, first-principles modeling of indoor air temperature can be a complicated task: it involves a nonlinear dynamical system whose inputs are the weather parameters and actuators manipulated variables, and the output is the predicted room temperature. An additional complexity is due to that fact that for each room in a building and for each variable of interest a separate model may be needed, to approximate the complex relationship between the system inputs (in this case the weather parameters and actuators manipulated variables) and the indoor air temperature. To overcome these issues, data-driven modeling techniques, such as Genetic Programming (GP) or Neural Networks (NN), are viable alternatives[3].

In this paper, we propose an adaptive Genetic Programming algorithm built upon a state-of-the-art method from the literature, namely the Bi-objective Genetic Programming (BioGP) [11]. Motivated by the empirical observation that the performance of BioGP highly depends on the chose parameter setting - particularly the mutation rate- we introduce in the original algorithm an adaptive mutation scheme that automatically tunes the mutation rate, based on a characterization of the current population (in terms of entropy) and on the information that can be extracted from it. We then show through numerical experiments on two different real world datasets that the proposed method, named BioAGP (where "A" stands for "Adaptive"), performs better than the original BioGP, allowing the search to maintain a good diversity level in the population, without affecting the convergence rate.

The remainder of this paper is structured as follows. The next section briefly presents the related works on Genetic Programming and details the working principles of BioGP. Section 3 describes the adaptive mutation mechanism proposed, and the motivations behind that. Then, in section 4 we present the numerical results obtained by BioAGP on two different datasets from the energy domain, in comparison with the original BioGP and with alternative data-driven modeling techniques. Finally, in section 5 we give the conclusions of this work.

2 Related work

In this section, we first recapitulate the main principles of Genetic Programming (section 2.1) then we describe in detail the original algorithm that is at the base of the proposed BioAGP (sec 2.2).

2.1 Genetic Programming

Genetic Programming is an evolutionary algorithm originally designed to find computer programs that perform a user-defined task [17]. Similar to other ge-

[3] We should observe, however, that the main drawback of data-driven modeling methods is that they depend entirely on experimental data. Therefore, such methods can only be applied *after* the actual building is built and measurements are available.

netic techniques used as optimization tools [5, 21], GP conducts a parallel search on a population of candidate solutions in the search space. In GP, however, a solution represents a set of mathematical functions which can be considered as an approximate model of the system at hand. A GP solution is typically encoded as a *binary tree*, whose nodes can be arbitrary mathematical functions, and leaves can be variables, or constants (as opposed to binary or real-valued genetic algorithms where solutions are encoded as arrays of bits or real values). The elements of the trees are, in general, predetermined, as they are extracted from two user-defined sets initialized before the beginning of the evolutionary process: a *function set*, containing user-defined mathematical functions, and a *terminal set*, containing variables and constants. Notably, one of the main advantages of GP e.g. in comparison to Neural Networks is the possibility of adding custom functions, which makes the algorithm extremely flexible [10].

GP starts off by creating an initial population of trees randomly initialized with elements from the function and terminal sets. Then, the population is evolved over a sequence of generations through mechanisms that mimic genetic recombination (crossover), mutation and selection. When used as a data-driven modeling method, typically a candidate GP tree is fed with a dataset containing n samples $\{(i_1, o_1), (i_2, o_2), \ldots, (i_n, o_n)\}$ (where each sample consists, in general, of m input variables and p output variables[4], i.e. $i_k \in \mathcal{R}^m$ and $o_k \in \mathcal{R}^p \ \forall k \in [1, n]$), then the root mean square error (RMSE) of each tree is computed as:

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{k=1}^{n}(o_k - \hat{o}_k)^2} \tag{1}$$

where $\hat{o}_k = f(i_k)$ is the estimated output calculated by that tree when fed with the k-th sample. The fitness of the tree is then its RMSE, and conventional GP algorithms simply try to find the minimum-RMSE tree [1].

2.2 Bi-objective Genetic Programming

We focus now on the Bi-objective Genetic Programming (BioGP) [11], the algorithm at the base of the present work. BioGP differs from conventional GP in that it does not minimize only the RMSE, rather it performs a bi-objective optimization. In particular, the algorithm tries to find a trade-off between the training error ξ (i.e., the RMSE calculated on the training set) and the model complexity ζ (described below) -both to be minimized- by employing the predator-prey scheme proposed in [25]. The algorithm then returns a set of Pareto-optimal solutions characterized by different trade-offs between model complexity and error, considering that more complex models tend to show lower errors (with

[4] In the rest of the paper, we will consider problems with $p = 1$ output variable. Nonetheless multiple output variables can be approximated by multiple GP trees, one per variable. An extension of the analysis for $p > 1$ is also possible and will be considered in future studies.

the risk over-fitting) while simpler models are characterized by higher errors (under-fitting).

In BioGP, each solution is represented as a weighed sum of sub-trees (each encoding a mathematical expression in Polish notation). More specifically, a linear sum is introduced as parent node, from which r roots emerge (one per sub-tree), where the number of roots r is a parameter of the algorithm. Consequently, the estimated output of a candidate solution fed with a sample i_k can be expressed as follows:

$$\hat{o}_k = f(i_k) = \sum_{j=1}^{r} \omega_j f_j(i_k) + \Theta \tag{2}$$

where $f_j(i_k)$ is the function corresponding to each j-th sub-tree calculated on the sample i_k, ω_j is its associated weight, $j = 1, 2, \ldots, r$, and Θ is a bias value. For each candidate solution, the weights and the bias value are calculated by the linear least square technique, such that the RMSE across all samples is minimized. Then, the solution's error ξ is simply determined by its RMSE. As for the complexity ζ, this is calculated as the weighted sum of the maximum depth of among all the sub-trees in the solution (δ), and the total number of function nodes (ν), since both terms contribute to the parameterization in the model. The objective function denoting the complexity ζ is then defined as:

$$\zeta = \lambda\delta + (1 - \lambda)\nu \tag{3}$$

where λ is a scalar, set equal to 0.5 as suggested in [30]. The parameter λ can be set to a different value to effectively control the growth of trees, in case the complexity goes beyond an acceptable limit. Furthermore, an additional parameter of the algorithm, d, allows to set the maximum depth of the sub-trees.

In a nutshell, the BioGP algorithm consists of two optimization steps. The first phase is a single-objective minimization of the error ξ, which continues until a prefixed error level is reached, or, alternatively, until a predefined number of generations is completed. Once this first step ends, the bi-objective optimization predator-prey genetic algorithm [25] starts. This mimics the interaction between a population of preys (each one corresponding to a candidate solution) and a population of predators, both moving on a two-dimensional lattice. The demography of preys is controlled by predators, which kill the least fit prey in their neighborhood.

It should be noted that the bi-objective is particularly computationally expensive, since it tends to generate solutions over a wide range of model complexity and error. In this sense, the introduction of the single-objective error reduction phase significantly reduces the computing cost of the algorithm, by guiding the solutions towards an acceptable limit of error that the user would be able to specify and find acceptable, without losing the choice and flexibility of the Pareto solutions.

One final note about BioGP regards its genetic operators. During the single-objective phase, BioGP applies a tournament selection (in the original paper

[11] of size 5). For the bi-objective part, selection is performed by means of non-dominated sorting and ranking [8]. As for crossover, BioGP uses a combination of standard and height-fair crossover [23]: in the former, two sub-trees are randomly interchanged between the participating parents, while in the latter case the exchange takes place at a selected depth. Also, BioGP use different types of mutations, namely standard, small and mono parental exchange: in case of a standard mutation, a sub-tree is deleted and then randomly regrown; small mutation implies replacing a terminal set by another (e.g. a numerical value in the terminal set is slightly altered, or a function is replaced with another having the same arity - e.g. "×" by "/"); finally, mono parental exchange involves swapping two sub-trees belonging to the same tree. In BioGP, the probability of activating one of the crossover or mutation operators is fixed.

3 Adaptive mutation mechanism

Just like for any other evolutionary algorithm, a crucial aspect in GP algorithms (including BioGP) is the balance between exploration and exploitation [29], i.e., respectively, the ability to visit entirely new regions of the search space, and to refine the search within the neighborhood of previously visited points. In general this is obtained by a careful tuning of the activation probability of the genetic operators, i.e. mutation and crossover. In the following, we refer to the mutation and crossover probability as p_m and p_c, respectively. In general, both p_m and p_c depend on the specific problem at hand, and in turn the overall effectiveness of a GP algorithm significantly depends on their setting. This might impair the use of GP especially for practitioners and users who may be not fully aware of the importance of this tuning of else are not sufficiently knowledgeable about the influence of such parameters.

To overcome the need for application-specific parameter tuning, a current trend in evolutionary algorithms is the use of adaptive mechanisms that are able to automatically tune the activation probability of the genetic operators [22, 15], such that the algorithm can flexibly adjust its behavior to a broad range of problems. The present work falls in this research area, as we focus specifically on the auto-adaptation of the mutation probability in BioGP.

As we saw in the previous section, BioGP uses three mutation operators (standard, small and mono parental exchange): in the following, we will refer to their activation probabilities respectively as p_{stand}, p_{small}, p_{mono}. In the original BioGP algorithm, $p_{stand} = p_{small} = p_{mono} = 1/3 \times 0.1$. Our intuition is that while these predetermined values might be efficient in some cases, the same could be sub-optimal in others. Therefore, we aim here at endowing BioGP with adaptive capabilities, such that the mutation rates can automatically change during evolution, rather than being fixed as in the original algorithm. Allowing for dynamic changes of the mutation rates, we could obtain a better balance between exploration and exploitation as the algorithm would adapt automatically its behavior to different fitness landscapes and search conditions: in fact, higher mutation rates might be needed when the algorithm is stuck into a local

optimum and the search should move to new regions of the solution space in the attempt to find the global optimum; vice versa, lower mutation rates are instead needed to improve exploitation when the algorithm is converging towards the optimum. With these considerations in mind, we introduce an adaptive mutation scheme as follows.

At each generation, the intrinsic information content of the population is calculated taking into account the worst (f_{max}) and best (f_{min}) individual training error ξ among all the trees in the current population, as well as the average error in the population (f_{mean}). Therefore, the normalized deviation between the worst individual and the population mean is calculated as:

$$diversity = \frac{f_{max} - f_{mean}}{f_{max} - f_{min}} \tag{4}$$

which is defined for $f_{max} \neq f_{min}$. If this value is low, the population is homogeneous (i.e. the worst individual is close to the population mean, indicating that all individuals have similar error values), otherwise the population contains individuals with a higher diversity in terms of errors. In the presence of over-fitting, all GP trees tend to show a very small training error: however, keeping some higher-error solution in the population (thus allowing for a higher level of diversity) might prevent an excessive convergence towards similar over-fitting GP trees characterized by small errors.

In addition to the diversity index, at each generation the Shannon entropy of the population is measured as:

$$-\sum_{k} p_k \log(p_k) \tag{5}$$

where the sum is calculated over all the groups in the population having the same value of training error ξ, and p_k is the fraction of the population having the k-th error value. Higher entropy values correspond to a more heterogeneous population, and vice versa.

Once the above metrics in eq. (4) and (5) are calculated, their values are compared with the values from the previous generation. If both the new values are lower than the previous ones (which happens when the entropy of the population is decreasing and the worst individual is getting closer to the population mean), the mutation rates are re-sampled as follows:

$$\begin{cases} p_{stand} = 0.1 + (0.33 - 0.1) \times rand() \\ p_{small} = 0.1 + (0.33 - 0.1) \times rand() \\ p_{mono} = 0.1 + (0.33 - 0.1) \times rand() \end{cases} \tag{6}$$

otherwise they are kept at their current values (starting from the original values used in BioGP). Here $rand()$ indicates a uniform random number sampled in $[0, 1]$. The rationale behind this adaptive scheme is that an increase of the mutation rates when the diversity in the population is decreasing might be beneficial to counterbalance premature convergence. Otherwise, the current mutation rates are maintained when the population has already a sufficient level of diversity.

4 Numerical results

To assess the performance of the proposed BioAGP (and determine the performance benefit due to the adaptive scheme), we performed numerical experiments on two different datasets from the energy domain, and compared the performance of BioAGP with that of the original BioGP and with alternative machine learning methods. In the following, first we describe our experimental setup (i.e. the datasets), then we analyze the numerical results obtained on the two datasets.

4.1 Datasets

The datasets considered in our experiments are both taken from the UCI repository [18]:

- The **Energy Efficiency** dataset. It is composed of 768 samples and 8 features, namely: compactness, surface area, wall area, roof area, overall height, orientation, glazing area, glazing area distribution. The goal is to predict one real valued response: heating load or cooling load.
- The **SML2010** dataset. It contains 4137 samples and 18 features, namely: date, time, weather forecast temperature, indoor temperature (dinning-room); relative to dinning room and room: carbon dioxide, relative humidity, lighting; sun dusk, wind, sun light in west/east/south facade, sun irradiance, outdoor temperature, outdoor relative humidity. The goal is to predict the indoor temperature.

In the experiments, we split each dataset into three sets (with sizes $60\%, 20\%, 20\%$) respectively for training, validation and test.

4.2 Experiments

Several tests were performed on the two real datasets with the main objective to obtain mathematical models with minimum test error and complexity.

- **BioAGP**: the algorithm proposed here, with parameter setting as suggested in [11], apart for the mutation rates that are replaced by adaptive mutation.
- **BioGP**: the original algorithm presented in [11], with the parameter setting suggested in that paper.
- **NEAT**: the NeuroEvolution of Augmenting Topologies (NEAT) method proposed in [27], with population size set to 500. All other parameters were set as in the original paper.
- **Neural Network (NN)**: a NN trained by Resilient Propagation [24]. The NN configuration was chosen applying the methodology described in [2] by means of the optimization software Kimeme [16], resulting in a network with three hidden layers, respectively with 91 nodes (Elliot symmetric activation function), 84 (with ramp activation function) and 68 nodes (with Gaussian activation function). The single output node used a hyperbolic tangent activation function.

– **Multiple Regression (MR)**: the ordinary ordinary least squares method that estimates the parameters of a multiple linear regression model.

All the algorithms above were implemented in Java code, with multi-thread parallelization at the level of each model evaluation. The proposed BioAGP was implemented by porting the original Matlab code available from [11] and adding the new adaptive mutation scheme. In both BioGP and BioAGP, we used as a function set $\{+, -, /, \times, \hat{}, \sqrt{x}, \ln(x)\}$ where "$\hat{}$" indicates the power function x^y. Since the computational cost of the GP algorithms is considerable higher than the other techniques (due to the parsing of a very large number of trees generated during the evolutionary process), we decided to run the two GP algorithms for a small number of generations (20) and a large number of predators and preys (respectively 100 and 500), to test their convergence under hard computational constraints and have a fair comparison with the other methods. As for the NEAT and NN, we used the open-source Java library Encog [12], coupled with Kimeme [16] as explained in [2]. Finally, the Multiple Regression algorithm was taken from the Apache Commons Java math library[5]. Each algorithm execution was repeated five times on both datasets, to calculate statistics on training, validation and test error.

First, we analyze the performance of BioAGP for different values of maximum depth and number of roots, see Tables 1-3. In the tables, we show the mean and std. dev. (over five repetitions) of training, validation and test error[6]. Also, we show in boldface the best tree configurations, i.e. the trees whose test errors are lower than those of the other configurations. We consider values smaller than $\varepsilon = 10^{-16}$ equal to zero.

We can see that the optimal configuration of tree depth and number of roots depend on the specific dataset and system to model: indicating with (d, r) a configuration (max depth, number of roots), we observe that the configurations (11,7) and (8,5) allow to obtain the minimum test error on the Energy Efficiency dataset, respectively for heating load and cooling load response; on the SML2010 dataset, the configuration (12,7) performs best. Despite the different optimal configurations, we can also observe qualitatively that the variance of the performance across all the tested configurations is quite small.

As a second part of the analysis, we compare the performance of BioAGP with that of other four methods, see Tables 4-6. In the tables, the values corresponding to BioAGP were taken from the best configurations shown in the previous tables. The comparative analysis shows that BioAGP performs consistently better than the original BioGP on all the tested datasets and variables of interest. However, when compared against other methods (NN, NEAT and MR), both GP methods produce higher validation and test error, especially on the Energy Efficient dataset. On the other hand, the GP methods are able to

[5] `http://commons.apache.org`

[6] We should remark that among all the Pareto-optimal solutions returned by BioAGP, we report here the ones characterized by the lowest training error, following the approach suggested in [11]. However other choices e.g. based on information criteria can also be made.

GP Tree Configuration (depth, roots)	Training Error (Mean ± Std. Dev)	Validation Error (Mean ± Std. Dev)	Test Error (Mean ± Std. Dev)
(6, 4)	0.0719 ± 0.0945	2.7733 ± 0.0648	2.9912 ± 0.1357
(8, 5)	0.0701 ± 0.1746	2.6555 ± 0.2184	3.0557 ± 0.1567
(9, 6)	0.0747 ± 0.1456	2.8500 ± 0.3360	3.1421 ± 0.2664
(10, 6)	0.0682 ± 0.1136	2.6166 ± 0.1221	2.8645 ± 0.1495
(11, 7)	**0.0647 ± 0.1169**	**2.4665 ± 0.1470**	**2.7391 ± 0.2373**
(12, 7)	0.0692 ± 0.2721	2.6330 ± 0.2546	2.9217 ± 0.2147

Table 1: Energy Efficiency dataset, heating load response

GP Tree Configuration (depth, roots)	Training Error (Mean ± Std. Dev)	Validation Error (Mean ± Std. Dev)	Test Error (Mean ± Std. Dev)
(6, 4)	0.0841 ± 0.0225	2.9994 ± 0.1808	3.0786 ± 0.1217
(8, 5)	**0.0839 ± 0.0146**	**3.0379 ± 0.1378**	**2.7965 ± 0.0449**
(9, 6)	0.0795 ± 0.0536	2.8104 ± 0.1496	2.8773 ± 0.1315
(10, 6)	0.0808 ± 0.0136	2.8795 ± 0.1349	2.9788 ± 0.0999
(11, 7)	0.0802 ± 0.0565	2.8554 ± 0.0889	2.8998 ± 0.0857
(12, 7)	0.0788 ± 0.0357	2.7607 ± 0.1439	2.9463 ± 0.0748

Table 2: Energy Efficiency dataset, cooling load response

GP Tree Configuration (depth, roots)	Training Error (Mean ± Std. Dev)	Validation Error (Mean ± Std. Dev)	Test Error (Mean ± Std. Dev)
(6, 4)	0.0170 ± 0.0435	0.3609 ± 0.3438	0.2866 ± 0.1217
(8, 5)	0.0164 ± 0.0564	0.3519 ± 0.3519	0.2817 ± 0.0034
(9, 6)	0.0163 ± 0.0239	0.3518 ± 0.3355	0.2793 ± 0.0041
(10, 6)	0.0165 ± 0.0594	0.3543 ± 0.3378	0.2839 ± 0.0026
(11, 7)	0.0165 ± 0.0882	0.3512 ± 0.3347	0.2842 ± 0.0125
(12, 7)	**0.0164 ± 0.0357**	**0.3433 ± 0.3271**	**0.2773 ± 0.0030**

Table 3: SML2010 dataset, temperature response

fit the data reasonably well even in a short number of generations, producing training errors which are almost on par with the other methods, if not better (compare the training error of BioAGP with that of the other methods in Table 4). Furthermore, the GP algorithms have the advantage of providing an actual mathematical model (instead of a black-box system such as NN and NEAT) that could be used for further analysis.

Another interesting aspect is that both GP algorithms seem to perform better for larger datasets (see the validation and test error on the SML2010 dataset in Table 6, which are is much smaller than the errors on the Energy Efficiency dataset shown in Tables 4-5), producing errors that are almost on par with the other techniques, if not better in some cases (compare e.g. the test error of BioAGP against that of NN in Table 6). This might indicate that both BioGP and BioAGP are able to leverage larger amounts of training data to build mathematical models with higher generalization capabilities, and explain

Algorithm	Training Error (Mean ± Std. Dev)	Validation Error (Mean ± Std. Dev)	Test Error (Mean ± Std. Dev)
BioAGP	0.0647 ± 0.1169	2.4665 ± 0.1470	2.7391 ± 0.2373
BioGP	0.0739 ± 0.0575	2.8612 ± 0.0644	3.1364 ± 0.0835
NEAT	0.0762 ± 0.0000	0.0747 ± 0.0053	0.0844 ± 0.0073
NN	0.0763 ± 0.0000	0.1667 ± 0.0000	0.1568 ± 0.0000
MR	0.0770 ± 0.0000	0.0801 ± 0.0000	0.0948 ± 0.0000

Table 4: Performance on Energy Efficiency dataset (heating load response)

Algorithm	Training Error (Mean ± Std. Dev)	Validation Error (Mean ± Std. Dev)	Test Error (Mean ± Std. Dev)
BioAGP	0.0839 ± 0.0146	3.0379 ± 0.1378	2.7965 ± 0.0449
BioGP	0.0867 ± 0.3467	3.1349 ± 0.0320	3.1076 ± 0.0399
NEAT	0.0796 ± 0.0000	0.0821 ± 0.0011	0.0877 ± 0.0183
NN	0.0697 ± 0.0000	0.2208 ± 0.0000	0.2248 ± 0.0000
MR	0.0886 ± 0.0000	0.0931 ± 0.0000	0.0964 ± 0.0000

Table 5: Performance on Energy Efficiency dataset (cooling load response)

Algorithm	Training Error (Mean ± Std. Dev)	Validation Error (Mean ± Std. Dev)	Test Error (Mean ± Std. Dev)
BioAGP	0.0164 ± 0.0357	0.3433 ± 0.3271	0.2773 ± 0.003
BioGP	0.0172 ± 0.0009	0.3613 ± 0.0164	0.2939 ± 0.0128
NEAT	0.0993 ± 0.0932	0.1038 ± 0.0023	0.1042 ± 0.0069
NN	0.0694 ± 0.0000	0.2952 ± 0.0000	0.3267 ± 0.0000
MR	0.0148 ± 0.0000	0.0181 ± 0.0000	0.1448 ± 0.0000

Table 6: Performance on SML2010 dataset

why on the Energy Efficiency dataset (which has only 768 samples, while the SML2010 dataset has 4137) the performance is lower.

5 Conclusion

In this paper we proposed an adaptive Genetic Programming method, BioAGP. The method is based on the state-of-the-art Bi-objective Genetic Programming algorithm (BioGP), and improves upon it by introducing an adaptive mutation mechanism that adjusts the mutation rates according to the current population diversity level. The proposed algorithm is specifically designed for data-driven modeling, as one of its main advantages is the ability to construct models with different levels of complexity and modeling error.

We tested BioAGP on two datasets related to indoor temperature prediction and energy efficiency in domestic environments. In a first part of the experiments, we showed that the algorithm is fairly robust for various configurations of maximum tree depth and number of roots, nevertheless it was possible to

identify an optimal configuration for each dataset and variable of interest. In the second part of the experimentation, we compared the performance of the best configurations of BioAGP against that one of alternative machine learning algorithms (the original BioGP, NN, NEAT and multiple linear regression). The numerical experiments showed that BioAGP consistently performs better than the original BioGP algorithm. The comparison against the other algorithms highlighted that the performance of BioAGP improves when a larger dataset is available, with comparable errors w.r.t. the other methods.

Future works will focus on the adaptation of the additional parameters of the algorithm (crossover probabilities, maximum depth and number of roots). Furthermore, we will test the method on modeling problems from other domains, such as bioinformatics [19, 20], sensor systems [3, 13], or robotics [4, 14].

References

1. Behbahani, S., De Silva, C.W.: Mechatronic design evolution using bond graphs and hybrid genetic algorithm with genetic programming. Mechatronics, IEEE/ASME Transactions on 18(1), 190–199 (2013)
2. Bevilacqua, V., Cassano, F., Mininno, E., Iacca, G.: Optimizing feed-forward neural network topology by multi-objective evolutionary algorithms: a comparative study on biomedical datasets. In: Artificial Life and Evolutionary Computation (WIVACE), Workshop on, pp. 1–12. Communications in Computer and Information Science, Springer (2016)
3. Caraffini, F., Neri, F., Iacca, G., Mol, A.: Parallel memetic structures. Information Sciences 227, 60 – 82 (2013)
4. Caraffini, F., Neri, F., Passow, B.N., Iacca, G.: Re-sampled inheritance search: high performance despite the simplicity. Soft Computing 17(12), 2235–2256 (2013)
5. Coello, C.A.C.: Multi-Objective Evolutionary Algorithms in Real-World Applications: Some Recent Results and Current Challenges. In: Advances in Evolutionary and Deterministic Methods for Design, Optimization and Control in Engineering and Sciences, pp. 3–18. Springer (2015)
6. Dasgupta, D., Michalewicz, Z.: Evolutionary algorithms in engineering applications. Springer Science & Business Media (2013)
7. Deb, K.: Multi-Objective Optimization Using Evolutionary Algorithms, vol. 16. John Wiley & Sons (2001)
8. Deb, K., Pratap, A., Agarwal, S., Meyarivan, T.: A fast and elitist multiobjective genetic algorithm: Nsga-ii. Evolutionary Computation, IEEE Transactions on 6(2), 182–197 (2002)
9. Ferreira, P., Ruano, A., Silva, S., Conceio, E.: Neural networks based predictive control for thermal comfort and energy savings in public buildings. Energy and Buildings 55, 238 – 251 (2012)
10. Garg, A., Tai, K.: Comparison of regression analysis, artificial neural network and genetic programming in handling the multicollinearity problem. In: Modelling, Identification & Control (ICMIC), International Conference on. pp. 353–358. IEEE (2012)
11. Giri, B.K., Hakanen, J., Miettinen, K., Chakraborti, N.: Genetic programming through bi-objective genetic algorithms with a study of a simulated moving bed process involving multiple objectives. Applied Soft Computing 13(5), 2613 – 2623 (2013)

12. Heaton, J.: Programming Neural Networks with Encog 2 in Java (2010)
13. Iacca, G.: Distributed optimization in wireless sensor networks: an island-model framework. Soft Computing 17(12), 2257–2277 (2013)
14. Iacca, G., Caraffini, F., Neri, F.: Memory-saving memetic computing for path-following mobile robots. Applied Soft Computing 13(4), 2003–2016 (2013)
15. Iacca, G., Caraffini, F., Neri, F.: Multi-strategy coevolving aging particle optimization. International Journal of Neural Systems 24(01), 1450008 (2014)
16. Iacca, G., Mininno, E.: Introducing Kimeme, a novel platform for multi-disciplinary multi-objective optimization. In: Artificial Life and Evolutionary Computation (WIVACE), Workshop on, pp. 1–13. Communications in Computer and Information Science, Springer (2016)
17. Koza, J.R.: Genetic programming: on the programming of computers by means of natural selection. MIT press (1992)
18. Lichman, M.: UCI Machine Learning Repository (2013), `http://archive.ics.uci.edu/ml`
19. Menolascina, F., Tommasi, S., Paradiso, A., Cortellino, M., Bevilacqua, V., Mastronardi, G.: Novel data mining techniques in acgh based breast cancer subtypes profiling: the biological perspective. In: Computational Intelligence and Bioinformatics and Computational Biology (CIBCB). IEEE Symposium on. pp. 9–16 (April 2007)
20. Menolascina, F., Bellomo, D., Maiwald, T., Bevilacqua, V., Ciminelli, C., Paradiso, A., Tommasi, S.: Developing optimal input design strategies in cancer systems biology with applications to microfluidic device engineering. BMC bioinformatics 10(12), 1 (2009)
21. Onwubolu, G.C., Babu, B.: New optimization techniques in engineering, vol. 141. Springer (2013)
22. Parmee, I.C.: Evolutionary and adaptive computing in engineering design. Springer Science & Business Media (2012)
23. Rennard, J.P.: Handbook of research on nature-inspired computing for economics and management. IGI Global (2006)
24. Riedmiller, M., Braun, H.: RPROP-a Fast Adaptive Learning Algorithm. In: Proc. of ISCIS VII, Universitat (1992)
25. Costa e Silva, M.A., Coelho, L.d.S., Lebensztajn, L.: Multiobjective biogeography-based optimization based on predator-prey approach. Magnetics, IEEE Transactions on 48(2), 951–954 (2012)
26. Stadler, W.: Multicriteria Optimization in Engineering and in the Sciences, vol. 37. Springer Science & Business Media (2013)
27. Stanley, K.O., Miikkulainen, R.: Evolving neural networks through augmenting topologies. Evolutionary Computation 10(2), 99–127 (2002)
28. Tsanas, A., Xifara, A.: Accurate quantitative estimation of energy performance of residential buildings using statistical machine learning tools. Energy and Buildings 49, 560 – 567 (2012)
29. Črepinšek, M., Liu, S.H., Mernik, M.: Exploration and exploitation in evolutionary algorithms: A survey. ACM Comput. Surv. 45(3), 35:1–35:33 (Jul 2013)
30. Vrieze, S.I.: Model selection and psychological theory: a discussion of the differences between the Akaike Information Criterion (AIC) and the Bayesian information criterion (BIC). Psychological methods 17(2), 228 (2012)
31. Zamora-Martnez, F., Romeu, P., Botella-Rocamora, P., Pardo, J.: On-line learning of indoor temperature forecasting models towards energy efficiency. Energy and Buildings 83, 162 – 172 (2014)