Computer Science and Information Systems 15(3):635–654 https://doi.org/10.2298/CSIS180110026A

A Fuzzy Rule-Based System to Predict Energy Consumption of Genetic Programming Algorithms

Josefa Díaz Álvarez[1], Franciso Chávez de la O[1], Pedro A. Castillo[2], Juan Angel García[1], Francisco J. Rodriguez[3], and Francisco Fernández de Vega[1]

[1] Mérida Campus. University of Extremadura, Mérida, Spain
{mjdiaz, fchavez, juangm, fcofdez}@unex.es
[2] Computer Architecture and Technology Department. University of Granada, Spain
pacv@ugr.es
[3] Computer Science Department. University of Burgos, Spain
fjrdiaz@ubu.es

Abstract. In recent years, the energy-awareness has become one of the most interesting areas in our environmentally conscious society. Algorithm designers have been part of this, particularly when dealing with networked devices and, mainly, when handheld ones are involved. Although studies in this area has increased, not many of them have focused on Evolutionary Algorithms. To the best of our knowledge, few attempts have been performed before for modeling their energy consumption considering different execution devices. In this work, we propose a fuzzy rule-based system to predict energy comsumption of a kind of Evolutionary Algorithm, Genetic Prohramming, given the device in wich it will be executed, its main parameters, and a measurement of the difficulty of the problem addressed. Experimental results performed show that the proposed model can predict energy consumption with very low error values.

Keywords: Green computing, energy-aware computing, performance measurements, evolutionary algorithms.

1. Introduction

Nowadays, power consumption has become a major concern due to electricity cost and availability constraints. Energy efficiency issues has been extensively addressed in the last decades by the scientific community. New products, procedures, and systems have been designed for ensuring the environmental sustainability, and computing environments are not an exception. The management of the energy consumption is a crucial problem in all kind of systems: mobile devices battery-powered, desktop computers and large data centers. A data center includes both traditional servers and new hardware platforms such as blade systems, which consume a huge amount of energy for both working and dissipating the heat generated. The energy dissipation causes thermal problems. A portion of the energy consumed is converted into heat, which affects the system reliability. The problem is more serious for battery-powered devices. Thus, reducing the power consumption is not only a matter of environmental awareness but also reliability.

In recent years, new algorithmic techniques to save energy have emerged. The term *proportional computing* covers a set of techniques mainly targeted to reduce the energy consumption in large data centers. A detailed and interesting revision is presented in [1].

These techniques can be labelled as Green Computing [20] techniques. The latter reduce the environmental impact and promote the energy efficiency, sustainability, and obtain positive results under most circumstances. However, the energy consumed by the algorithms executed in those systems has been always forgotten.

In particular, Evolutionary Algorithms (EA) are broadly applied for many complex optimization problems and diverse areas on previously mentioned hardware platforms. It is important to note that big systems are not always necessary to address an optimization problem. Desktop systems and new and powerful ephemeral platforms [10] help to experiment and solve problems in different scientific areas in academic and research environments. They can help to reduce the power consumption when the problem is not too complex from a computational point of view. These platforms may need lower power consumption facing a higher execution time.

However, how much energy is needed to get some experiments done for a given optimization problem? It must be noticed that the energy consumed by the algorithm not only depends on its parameters such as number of generations or population size, but also the hardware platform and the problem to be addressed. A good prediction about the energy consumption according to the problem at hand and the parameters of the algorithm would be very useful to make an appropriate decision. This decision can be oriented towards the algorithm settings, the solutions quality (which is not addressed in this work) or the hardware platform to be used. One of the goals of the *Green Computing* approach is modeling the behavior of a system, for example, for being able to predict and respond to future events.

Less powerful hardware platforms can run algorithms with higher number of generations or population size with a lower energy consumption than others more powerful [34]. Therefore, addressing the generation of a predictive model of energy consumption might be interesting to know a priori the most suitable hardware platform. We deal with the problem of predicting the energy consumption of a popular kind of EA, Genetic Programing (GP) [21], which tries to evolve computer programs when looking for optimal solutions for a given problem.

In particular, we focus on implementing a predictive model for energy consumption of GP dealing with two typical regression problems on three different hardware platforms. We aim to predict it taking into account the main GP parameters, the difficulty of the problem addressed, and the hardware platform. For this purpose, we apply a Takagi-Sugeno-Kang Fuzy Rule-Based System [29] (TSK-FRBS), using *k-fold cross validation* as training strategy in order to improve the generalization capability of the predictive model.

The rest of the paper is organized as follows. Section 2 reviews the related work about energy optimization techniques. Section 3 describes the optimization framework, detailing how data are obtained and processed, and how the model has been built. Then, Section 4 describes the experimental framework and analyzes results obtained. Finally, Section 5 draws conclusions and future work.

2. Related Work

The behavior of EAs has been widely analyzed for years from several points of view. Particularly, the behavior of GP and its parameters have been extensive analyzed [22,9,17].

Some works have developed models to predict the performance of a GP algorithm until the optimal solution is found. Graff and Poli [15] introduce two models of performance based on difficulty indicators for EPAs (Evolutionary Program-induction Algorithm). They were applied to symbolic regression and Boolean induction problems. However, the prediction of the performance do not include any study neither about the energy consumed nor the hardware platform that can be used. Trujillo et al. [30,31] developed an approach for predicting the performance of GP applied to data classification. Recently, Martinez et al. [25] presented a predictive model of expected performance. This predictor adds new descriptive measures and evaluates the performance of the GP classifier on the test set of fitness cases.

One of the main goals for last decades has been both to increase the performance and built more energy efficient hardware systems. However, a performance increasing implies a significant growth in energy consumption. Thus, both must evolve at the same speed to improve the energy footprint.

The optimization of the power consumption can be addressed from a hardware or software perspective. In this regard, some successfull techniques have been designed to improve the energy efficiency. The Dynamic Power Management (DPM) [7] is a methodology for dynamically reconfiguring systems by turning off/on some components without performance loss. Power-down mechanism [5] applies different DPM techniques and off/on line algorithms that allow systems to move from a high power state to a lower one (standby or hibernate mode), when there is no activity for a period of time according to an algorithm. Dynamic Voltage Frequency Scaling (DVFS) [8,18] is a stable and powerful technique in power management to save energy. It provides the ability to adjust voltage and work frequency based on the workload requirements. Speed Scaling has long been applied at a chip level for save energy and nowadays is applied at all systems levels [6,28]. Moreover, the search for the optimal algorithms is a promising research field nowadays [2]. These techiques are based on processor speed scaling and power-down mechanisms and assume that they are supported by the CPU technology. Therefore, EAs benefit from this techniques.

On the other hand, new hardware technologies and core-based processor technologies, such as ARM [4], allow changing others hardware settings like the structural parameters of the cache memory according to the running application to be more efficient in terms of performance and energy.

The energy management is a key issue not only in computer based systems. EAs have been applied in different areas to improve the energy efficiency. Literature mentioned below uses tangentially EAs to optimize the energy consumption. Vasicek et al. [32] proposed a new systematic method for designing energy efficient electronic component based on circuit approximation and Cartesian Genetic Programming (CGP). DVFS combined with scheduling techniques and multicore systems at the task level can improve the memory energy consumption [26]. Diaz et al. [3] presented a multi-objective algorithm (NSGA-II) approach to find cache configurations for reducing power consumption and execution time.

Hameed et al. [16] presented a review of the techniques applied in cloud platforms for reducing the power consumption and the energy dissipation, which are important concerns in this kind of platforms. EAs are present in many of the techniques proposed in this area. For instance, Gao et al. [14] applied a multi-objective ant colony algorithm to map

efficiently virtual machines to physical machines and improve the energy efficiency and reduce the wasted resources.

Regarding mobile devices, for example, Linares et al. [23] proposed a Graphical User Interface of Android applications, which uses a colour palette aiming to optimize the energy consumption. This colour palette is generated using a multi-objective optimization technique.

From other different perspectives, Lu et al. [12] reviewed several research works, where multi-optimization algorithms are applied for solving problems in the area of renewable and sustainable energy. Logenthiran et al. [24] presented a management strategy from the demand side to be applied in smart grid. They developed a heuristic optimization based on evolutionary algorithms to help clients to take decisions about their energy consumption.

However, works where EAs are applied, including all previously mentioned, generally consider EAs as a tool to improve the energy efficiency when facing a given problem, and they do not take into account their own energy consumption whilst the optimization process is carried out.

In this context, a preliminary work [33] measured the energy consumed by an EA, particularly GP. The behavior of the multiplexer problem was analized on several different platforms such as blade systems, tablet, laptop and Raspberry-pi. The main goal was to determine the relationship between the main parameters of the algorithm and the power consumption. As a conclusion, a device like Raspberry-Pi can be more energy efficient when not a very computationally complex problems are addressed.

Wang et al. [35] presented an instruction-level energy model for a single core and RISC processor architecture, however their approach is different the one we present.

An energy estimation model, especially aimed at GP algorithms, was proposed in a subsequent work [11]. This preliminary model applied a simple Fuzzy Rule-Based System to predict the energy consumption by using an inference engine. Experimental tests with the multiplexer-6 problem were carried out on Raspberry-Pi, laptop and tablet platforms. As a result, the energy consumption predictive model has a low error in the predictive phase. However, this preliminary work obtained one predictive model for each hardware platform and only the multiplexer problem.

3. Methodology

As mentioned before, in this work we propose an approach to implement an energy predictive model for EAs, particularly GP. This approach is based on the study of the main three GP parameters: the number of generations, population size and maximum depth. We add two new parameters: the hardware platforms to be used and the difficulty of the optimization problem to be solved, obtained by a profiling.

The method presented in this work to develop the energy predictive model is divided into three different stages. Fig.1 presents an overview of the whole process. The first stage we have to calculate the difficulty of the problems chosen, which must be computed only once for each problem at hand. The second stage corresponds to the execution of the algorithm according to the problem and defined parameters. In this state, the energy consumption value is stored in a profiling report for each application. The third stage is called the FRBS module, which the fuzzy system is designed and optimized taking as

input the profiling report previously obtained, where the values of the parameters that describe the problem executed are stored, together the energy consumption value.

First and second stages are intended to provide key information to fulfill the profiling report needed for the third one. The profile report designed contains a line for every different combination of the main GP parameters addressed, the measurement of the problem difficulty obtained in the first stage, and the energy consumption according to the parameters. Next, we explain each stage in detail. The following subsection details how we compute the difficulty of the problems addressed.

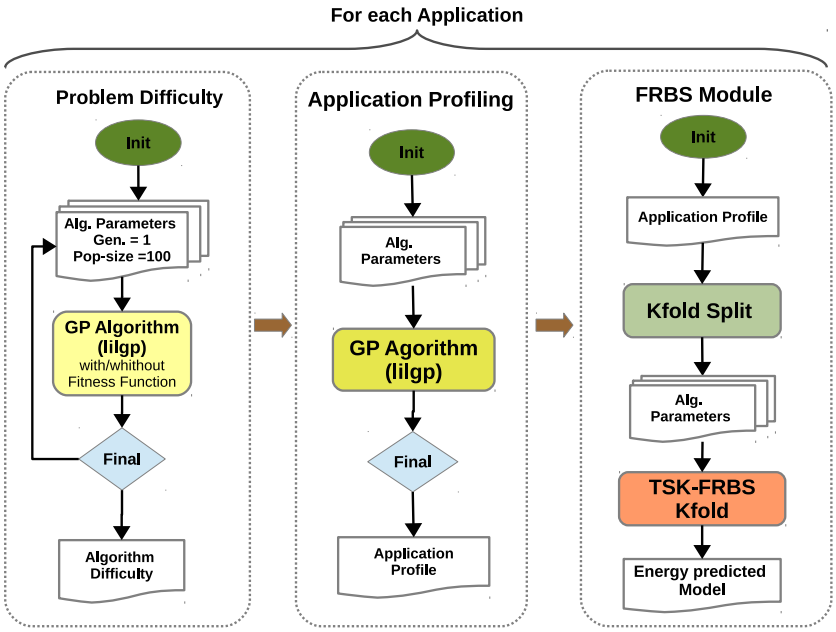

Fig. 1. Stages involved in the process to obtain the energy prediction model.

3.1. Optimization Problem Difficulty

To identify the difficulty of a problem, the algorithm must be executed. Different metrics can be applied to evaluate the problem difficulty until the algorithm finds the optimal solution. However, the method applied to find the optimal solution is a key factor and it is represented by the fitness function [17]. Thus, the difficulty of an optimization problem to be solved by means of EAs is evident in the fitness function, which also identifies the optimization problem.

We compute the difficulty as the effort needed to solve an optimization problem as shown in (1), where C_x is the effort the algorithm have to do according to the problem at hand; T_{af} is the time computed when the algorithm evaluates each solution with the fitness function, and T_{sf} represents the time of the algorithm without implementing the

Table 1. Main GP Parameters. A total number of 1716 combinations were tested.

Generations	10, 20, 40, 60, 80, 100, 150, 200, 300, 400, 500
Population sizes	10, 20, 30, 40, 50, 60, 70, 80, 90, 100, 120, 140, 160, 180, 200, 250, 300, 350, 400, 450, 500, 600, 700, 800, 900, 1000
Max depth	3, 5, 6, 7, 8, 9
Crossover probability	0.9
Mutation probability	0.1

fitness function. In order to compute this value accurately, we launch twenty times each experiment for one generation and 100 individuals and we average the values obtained. This set of tests are done for each problem and value of maximum depth to be addressed. As a result, we obtain the time influence of the fitness function in terms of percentage, with respect to the total time of the algorithm.

$$C_x = \frac{(T_{af} - T_{sf})}{T_{af}} \times 100 \tag{1}$$

The time measurements are calculated taking the difference between the time just immediately before launching the algorithm and just after finishing. This measurement do not take into account the time spent filling the log report neither computing these measures. Time units used in seconds and the percentage computed will complement the information obtained by the second stage, which is explained below. Moreover, whilst this first stage is carried out we measure the average power delivered for each device, which is used to compute the percentage over the highest power and allow us to adjust the ECPM proposed.

3.2. Application Profiling

This stage is in charge of executing the GP algorithm for each problem according to the parameters chosen, and get a profiling report to which computational effort obtained in the previous stage will be added.

To design this energy consumption estimation model, two well-known GP benchmark problems have been selected: multiplexer (6 bits) and regression. The multiplexer problem simulates the behavior of this electronic component. The 6 bits implementation has two address input and four data entries. The regression problem uses GP to discover the function $Y = \mathrm{X}^4 + \mathrm{X}^3 + \mathrm{X}^2 + \mathrm{X}$. Both are included in lilgp [4], a well-known GP implementation in *C* language. The language related decisions have a high computational influence on the algorithm [27] and, thus on the energy consumption, which is addressed in this work. Functions and terminal sets are the standard ones described by Koza [21].

The main EAs parameters have a considerably impact on the algorithm performance and, hence on the energy consumption. Table 1 shows the main parameters and their set of values considered to carried out the experimental tests and obtain the profiling report. The stop condition is set when the optimal solution is found. The more information we provide to the fuzzy system, the better prediction we obtain.

[4] http://garage.cse.msu.edu/software/lil-gp

Each individual experiment has been launch 30 times because of the stochastic nature of EAs and we compute the performance and energy consumption as the average of the 30 runs. Each run is parameterized with a different seed and the same thirty seeds are used in all experiments and devices. We use seconds to compute the performance as the difference between time before launching the algorithm with a particular combination of parameter values and just after finishing. We do not compute the time needed to store log information neither computing.

The energy consumption is computed assuming devices are working at the highest power according to $Energy = Power * Time$. This decision was taken given the interest of this work in testing this methodology to provide a model to predict the energy consumption. Future development will need accuracy measures about the average power delivered by each device and problem addressed, whilst the algorithm is carried out. As previously mentioned, for each different combination of the main parameters of GP algorithms described in Table 1, the algorithm is run thirty times and computes the average time and energy consumption in each device.

Results obtained for each GP problem and devices are stored in a different profiling report for being further processed by the fuzzy system in order to build the energy estimation model, where units used for energy, power and time are Joules, Watts and Seconds, respectively. As soon as the profiling stage finishes, the next stage executes the FRBS module, which is addressed in the following subsection.

3.3. FRBS Module

As previously stated, the prediction model is performed by the FRBS module, as seen in Fig. 1. The Energy Consumption Predictive Model (ECPM) we propose needs, first of all, to establish the relationship among the set of parameters defined, whose values are collected in the profiling report previously mentioned, by executing the GP problems chosen. These data or input variables are included as key information during the learning phase to design the ECPM, what will allow subsequently evaluate the energy consumption. The variables defined for the ECPM described are:

- Input Variables
  - device
  - population size
  - number of generation
  - maximum depth
  - problem difficulty
- Output Variable
  - energy consumption, defined as the value to be predicted.

In this work, we propose a predictive system based on METSK-HD [13], an evolution of TSK-FRBS [29], which improves the accuracy and convergence when high-dimensional and large-scale regression datasets are managed. The structure consists of a Knowledge Base (KB) and Rules Database (RB). The KB stores the knowledge extracted of the problem at hand, and establishes the relationship between the input and output variables by means the well-known Membership Functions(MF). The knowledge is finally stored as fuzzy rules with an *IF-THEN* format. Fuzzy rules have a structure with

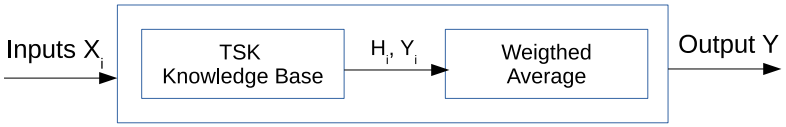

Fig. 2. TSK-FRBS model.

an antecedent and a consequent. The antecedent is composed of linguistic variables, and the consequent is a polynomial function of the input variables. Thus, the rules format of a TSK-FRBS has the following structure:

$$\begin{aligned} &\text{IF } X_1 \text{ is } A_1 \ \ldots \text{ and } X_n \text{ is } A_n \\ &\qquad \text{then Y} = p_1 * X_1 + \ \ldots \ + p_n * X_n + p_0 \end{aligned}$$

where the system inputs variables are denoted as X_i, Y is the system output variable, p_i represents real-values coefficients and A_i are fuzzy sets.

Fig. 2 shows the flow of the TSK-FRBS model. Considering the KB contains m TSK rules the output of TSK system is computed as the weighted average of each individual rule output Y_i as shown (2):

$$\frac{\sum_m^{i=1} h_i \cdot Y_i}{\sum_m^{i=1} h_i} \tag{2}$$

where $i = 1...m$, $h_i = T(A_1(x_1)...A_n(x_n))$ represents the matching degree between the antecedent part of the ith rule and the current system inputs $x = (x_1...x_n)$, and with T being a t-norm.

TSK FRBSs have been applied successfully to a large quantity of problems. The main advantage of these kinds of systems is the fact that they present a compact system equation for estimating the parameters p_i using classical methods, and obtaining an accurate system, which can be very useful for accurate fuzzy modeling.

Problems with large or high-dimensional data sets make non-feasible an ad-hoc implementation and need an automatic learning process for the KB and RB. The process is divided into different stages due to the high complexity of the search space involved, although KB and RB should be learnt and optimized together. Different techniques have been applied for this task, but the Genetic Fuzzy Systems (GFS) [19] has the best results. EAs are able to learn the antecedents and consequents of the rules system together, and to optimize the MF of the KB. Figure 3 shows the summary of the optimization process of TSK FRBS.

This process is divided into two stages, called *Learning* and *Tuning*. In the first stage, *Learning*, the initial Data Base (DB) based on a fuzzy grid in order to obtain zero-order TSK candidate rules, is learned. The second stage, *Tuning*, applies an advanced post-processing for fine scatter-based evolutionary tuning of MFs combined with a rule selection. Figure 4 shows the process described previously.

Additionally, we use K-fold cross validation (CV) with $K = 5$ as training strategy, which is widely applied for model evaluation in classification. K-fold CV splits the train-

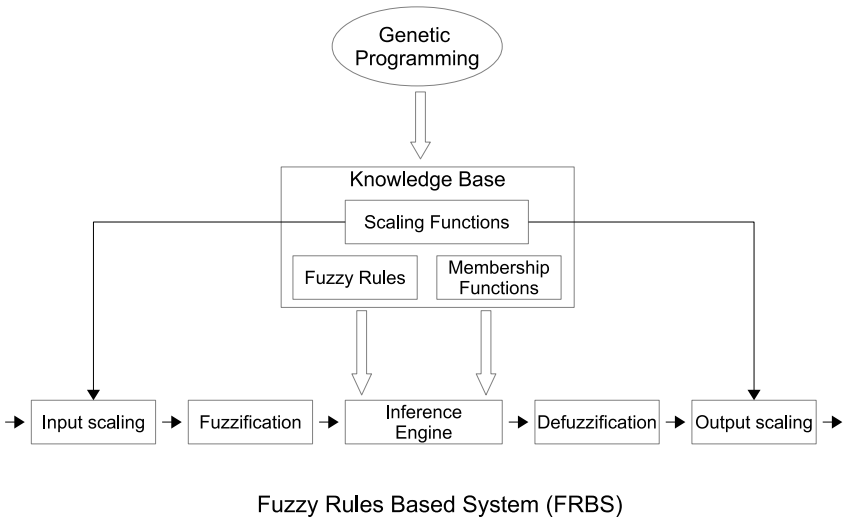

Fig. 3. Processes involved in the FRBS module.

ing data into K different sets, $S = (S_1.....S_k)$. For a particular evaluation i, S_i represents the test set and the combination of the remaining sets are used for model fitting.

4. Experimental Results

4.1. Experimental Framework

In order to test our proposal, we have selected the two benchmarks previously mentioned from the lilgp suite: multiplexer-6 and regression, which are representative benchmark problems. Three computational platforms have been tested: Raspberry-Pi, laptop and tablet. Table 2 provides the technical details of the hardware architectures and operating systems. Facing an optimization problem, the power consumption depends not only on the algorithm but also the architectural features of the device in which it is executed. Future decisions on the hardware to run an optimization algorithm may take into account the processor speed, instruction set architecture and operating system, which we do not take into account in this work.

We are aware of the possibility of disabling services unused such as wifi, bluetooth, hdmi connection, leds on the board, etc. to reduce the power consumption. In this paper, we consider all components are enabled and devices are plugged whilst the GP algorithm is running.

As stated before, the difficulty of the problems addressed has been computed as the time taken by the algorithm when the fitness function is computed or not. Next, the algorithms have been launched on each hardware platform and the set of parameters chosen. As a result, the profiling report needed to the next step has been completed. The profiling report obtained contains five input variables and the output one, defined in Section 3.3.

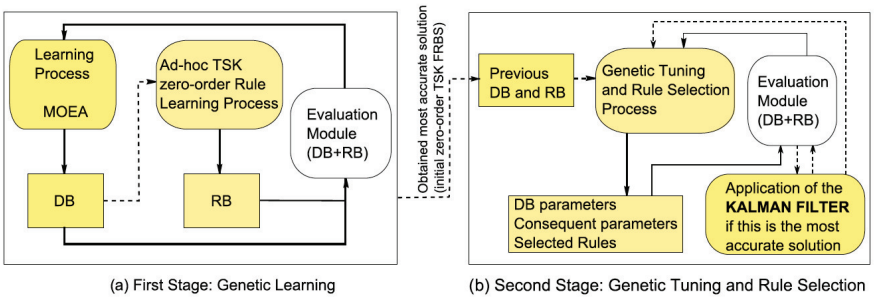

Fig. 4. TSK FRBS optimization process.

Table 2. Computational platforms (devices) used in this work.

Device	raspberry pi	tablet	laptop
Processor	Cortex-A7 900 MHz	Samsung Galaxy Tab 3 SM-T311. Exynos 4212 1.5 GHz	Intel(R) Core(TM) i5-2450M 2.5GHz
Cores	4	2	4
RAM	1GB	1.5GB	8GB
OS	Raspbian GNU/Linux 7	Android 4.4.2 (kernel 3.0.31)4	Ubuntu 12.04.5 LTS
Watt-hour	12.5	14.8	75

Finally, the ECPM is built after the execution of TSK-FRBS with 5-fold cross validation as the training strategy. As mentioned before, TSK-FRBS is launched 30 times for each experiment due to the stochastic nature of this kind of algorithms.

For the sake of clarity, we would like to remark that in this work an ECPM model for each kind of problem have been built. The TSK-FRBS used has two different processes. The first process is called *Learning*, in this process the systems used a EA to design the whole KB (DB and MFs). In this process the system begins with an initial FRBS designed randomly and, be means of an EA, the systems describe the whole system, where the DB and MFs are described. The second process tries to improve the FRBS designed in the above stage. In this stage, again a EA is applied to reduce the number of MFs obtained in the *learning* process. But not only do they try to reduce the rules, but also try to improve their efficiency again. In both processes, the method applied will try to eliminate variables that apparently are not necessary.

The energy consumption is highly dependent on the hardware platform, and the population size, number of generations and maximum depth the main parameters of GP algorithms. Moreover, the problem difficulty is together with the previously mentioned parameters the key to estimate the time needed to reach a solution, and consequently the energy consumption. Therefore, an ECPM particularly targeted at evolutionary algorithms cannot ignore them.

4.2. Experimental Analysis

This section analyzes the experimental results obtained. Fig.5 shows the evolution of the Mean Squared Error (*MSE*) and Root Mean Squared Error (*RMSE*) metrics for the multiplexer problems in training and test. Results obtained with our methodology reach a high hit rate, values predicted are close to the original values. More detailed information is given in Appendix A, Table 3, which compiles results obtained for each individual experiment of FRBS, particularly: the number of rules, *MSE* and *RMSE* for training and test. We can see values for *MSE* and *RMSE* are close to 0, which would be the perfect tuning between the observed value and the estimated value.

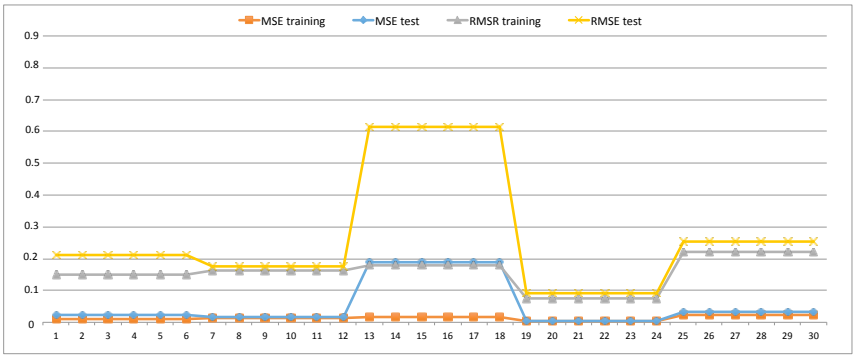

Fig. 5. *MSE* and *RMSE* values for the multiplexer problem in training and test through 30 executions.

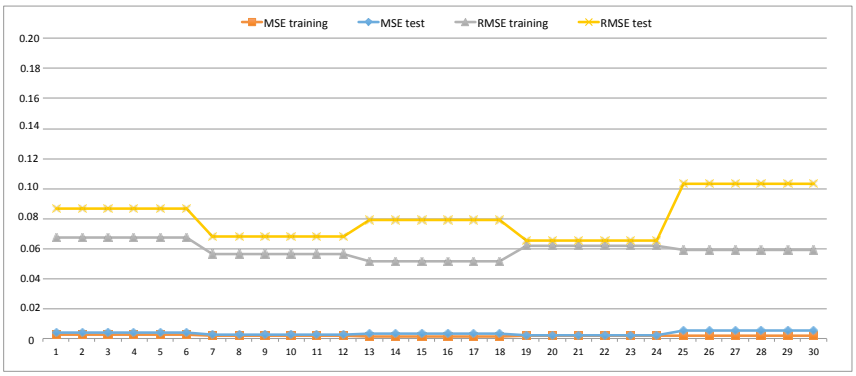

Fig. 6. MSE and RMSE values for the regression problem in training and test through 30 executions.

Moreover, the number of final rules in the tuning phase has been reduced more than a 20.9% with respect to the preliminary work [11], previously mentioned. The latter gen-

erated a model for the multiplexer problem and each hardware platform, thus we refer to the average number of rules 81. A lower number of rules allows to improve the system interpretability and makes easier a further development to provide predictions to users.

Fig.6 plots the evolution of the *MSE* and *RMSE* for the regression problem. More extended information is represented in Appendix A, Table 4. Values obtained for the *MSE* and *RMSE* metrics are more relevant in the case of the regression problem. These metrics are lower than in the multiplexer problem and they are closer to 0, which shows the tuning obtained is near the perfect tuning. On the other hand, the number of rules for the ECPM built is low enough to make easier the development of an interactive applications for final users. Users could introduce their custom parameters and the system would give as answer the energy consumption of the set of tests, using a given hardware platform. Users could take the decision according to the information that best fits the desired energy consumption.

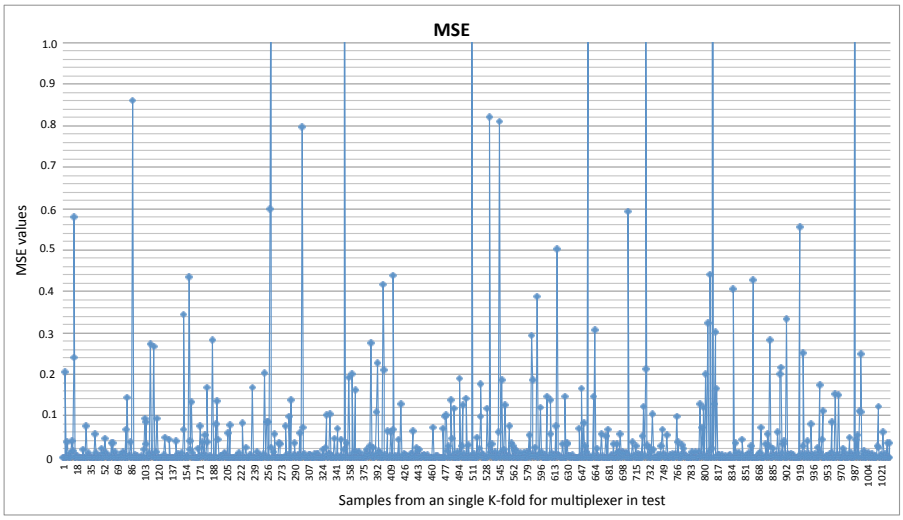

Fig. 7. MSE for the multiplexer problem for an individual kfold

Moreover, it might be interesting to analyze the prediction accuracy according to GP parameters. Fig. 7 and 8 show *MSE* for each profiling data in one of the k-folds for the multiplexer and regression problems, respectively. Note that we have chosen different k-folds for both problems in order to avoid any possible bias. If we analyze firstly *MSE* values in Fig. 7 for each of the 61 profiling data in the selected k-fold, we observe that the vast majority of *MSE* are under 0.2. However, some of the *MSE* values differs ostensibly from the latter, reaching values close to 1. We can therefore conclude that the accuracy of the prediction obtained depend on the values of the profiling data. We have checked out those profiling data for which *MSE* are higher, and we have discovered that these predictions correspond to profiling data with values for the GP parameters at the upper range (See Table 1). This lower accuracy might me due to the lack of enough samples at the upper range of GP parameters. While the interval between different values of GP

parameters is small for the lower range, these intervals increase as we approach the upper range. Therefore, we think that we could improve *MSE* by including more values for GP parameters within their upper ranges. However, it is important to point out that despite this, the accuracy obtained by our model is outstanding, as *MSE* values show.

We also perform the same analysis for the regression problem, but using a different k-fold. Fig. 8 shows the *MSE* values for each of the 65 profiling data of the chosen k-fold. In this case, we can clearly see that are much less than in the previous case the values that move away from the most frequent value, which as can be appreciated is very close to 0. Again, we can verify that these instances with larger *MSE* values correspond to instances with values of the GP parameters in their highest ranges. The conclusions obtained are therefore very similar to those obtained for the of the multiplexer problem.

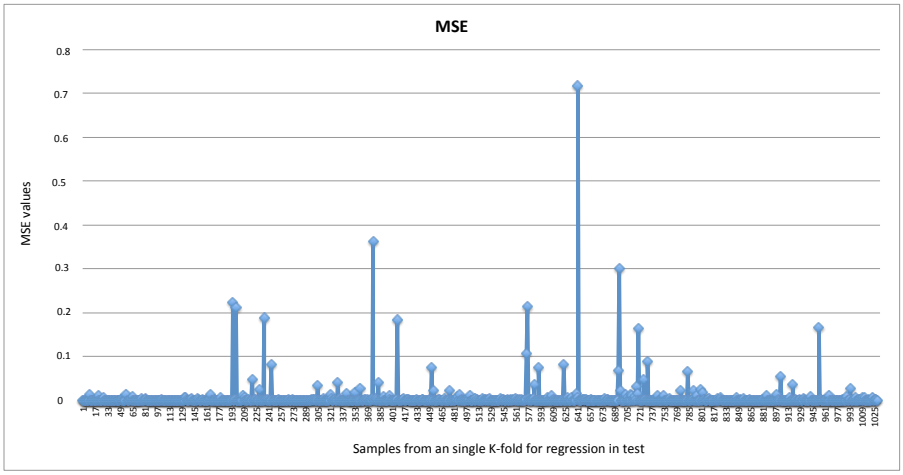

Fig. 8. MSE for the regression problem for an individual kfold

5. Conclusions and Future Work

We have presented an energy estimation model, which is especially aimed at EAs, in particular, GP algorithms. To implement this model we have applied a TSK-FRBS that is able to predict the energy consumption by using an inference engine. A 5-fold cross validation process has been applied as the training strategy. We have addressed two typical problem of GP, the multiplexer-6 and regression problems developed with `lilgp`, a well known C implementation for the GP algorithms. Experimental tests have been carried out on three different hardware platforms: raspberry pi, laptop and tablet , where a different operating system is running: *Raspbian*, *Ubuntu* and *Android*.

In this paper, we have obtained one model for each problem addressed. Results obtained show the Energy Consumption Predictive Model reaches a high hit rate, as illustrated by the low values of *MSE* and *RMSE* in the prediction phase on tests performed.

As it is expected, the main parameters of the GP algorithm and the difficulty of the problem must have a considerable influence on the energy consumption. The ECPM is able to make good predictions, however there are some differences due to the lack of samples of the high range of the GP parameters. This can be solved running the algorithm with these parameters and adding values to the profiling report to improve both the learning and tuning phase.

The main conclusion we may present from the results is that this ECPM built with TSK-FRBS is able to correctly estimate the energy consumption of GP for each problem addressed. The usefulness is extensive when some problems need to be solve, an upper limit of energy is established, and different hardware platforms are available to be used. The model would allow to compare several settings before launching the run and propose the best option at the least energy consumption. Moreover, this model can be improved associating the energy consumption with the solutions quality in order to reach the best compromise between the energy efficiency and the solutions quality found.

It is important to note that, unlike the preliminary work, this model is not dependent on the hardware platform. The hardware platform is added as an additional input parameter of the designed model. Moreover, the difficulty of the problem has been also included as an input parameter to the TSK-FRBS prediction model. Those parameters did not belong to the preliminary prediction model presented in previous works.

Although the Energy Consumption Predictive Model presented is based on two GP problem, we think it can be applicable to other EAs. We hope to extend in the future this model with a wide set of experiments, considering more different problems and EAs. Moreover, we do not have a system that automatically analyzes the set of rules when a final user requests information. An interesting future work would be the development of an application, which give automatically the answer according to the given parameters values.

Acknowledgements. We acknowledge support from Spanish Ministry of Economy and Competitiveness under projects TIN2014-56494-C4-{1,2,3}-P and TIN2017-85727-C4-{2,4}-P, Regional Government of Extremadura, Department of Commerce and Economy, conceded by the European Regional Development Fund, a way to build Europe, under the project IB16035, and Junta de Extremadura FEDER, projects GR15068 and GR15130. We would also like to express our very great appreciation to Dr. Miguel Frade and Dr. Joao Sousa for their valuable and constructive suggestions

A. Detailed Results

Table 3. Individual experiments of FRBS 5-fold for the multiplexer problem.

	Learning					Tuning				
	Rules	*MSE*		*RMSE*		Rules	*MSE*		*RMSE*	
		Training	Test	Training	Test		Training	Test	Training	Test
1	96	0.0774	0.1036	0.3934	0.4552	63	0.0111	0.0224	0.1487	0.2118
2	96	0.0774	0.1036	0.3934	0.4552	63	0.0111	0.0224	0.1487	0.2118
3	96	0.0774	0.1036	0.3934	0.4552	63	0.0111	0.0224	0.1487	0.2118
4	96	0.0774	0.1036	0.3934	0.4552	63	0.0111	0.0224	0.1487	0.2118
5	96	0.0774	0.1036	0.3934	0.4552	63	0.0111	0.0224	0.1487	0.2118
6	96	0.0774	0.1036	0.3934	0.4552	63	0.0111	0.0224	0.1487	0.2118
7	96	0.0711	0.0550	0.3772	0.3317	65	0.0134	0.0157	0.1637	0.1774
8	96	0.0711	0.0550	0.3772	0.3317	65	0.0134	0.0157	0.1637	0.1774
9	96	0.0711	0.0550	0.3772	0.3317	65	0.0134	0.0157	0.1637	0.1774
10	96	0.0711	0.0550	0.3772	0.3317	65	0.0134	0.0157	0.1637	0.1774
11	96	0.0711	0.0550	0.3772	0.3317	65	0.0134	0.0157	0.1637	0.1774
12	96	0.0711	0.0550	0.3772	0.3317	65	0.0134	0.0157	0.1637	0.1774
13	96	0.0654	0.0646	0.3616	0.3596	62	0.0158	0.1895	0.1776	0.6156
14	96	0.0654	0.0646	0.3616	0.3596	62	0.0158	0.1895	0.1776	0.6156
15	96	0.0654	0.0646	0.3616	0.3596	62	0.0158	0.1895	0.1776	0.6156
16	96	0.0654	0.0646	0.3616	0.3596	62	0.0158	0.1895	0.1776	0.6156
17	96	0.0654	0.0646	0.3616	0.3596	62	0.0158	0.1895	0.1776	0.6156
18	96	0.0654	0.0646	0.3616	0.3596	62	0.0158	0.1895	0.1776	0.6156
19	90	0.1094	0.1114	0.4678	0.4719	65	0.0027	0.0042	0.0741	0.0915
20	90	0.1094	0.1114	0.4678	0.4719	65	0.0027	0.0042	0.0741	0.0915
21	90	0.1094	0.1114	0.4678	0.4719	65	0.0027	0.0042	0.0741	0.0915
22	90	0.1094	0.1114	0.4678	0.4719	65	0.0027	0.0042	0.0741	0.0915
23	90	0.1094	0.1114	0.4678	0.4719	65	0.0027	0.0042	0.0741	0.0915
24	90	0.1094	0.1114	0.4678	0.4719	65	0.0027	0.0042	0.0741	0.0915
25	96	0.0861	0.1031	0.4150	0.4541	66	0.0245	0.0323	0.2214	0.2541
26	96	0.0861	0.1031	0.4150	0.4541	66	0.0245	0.0323	0.2214	0.2541
27	96	0.0861	0.1031	0.4150	0.4541	66	0.0245	0.0323	0.2214	0.2541
28	96	0.0861	0.1031	0.4150	0.4541	66	0.0245	0.0323	0.2214	0.2541
29	96	0.0861	0.1031	0.4150	0.4541	66	0.0245	0.0323	0.2214	0.2541
30	96	0.0861	0.1031	0.4150	0.4541	66	0.0245	0.0323	0.2214	0.2541
Mean	94.8	0.0819	0.0875	0.4030	0.4145	64.2	0.0135	0.0528	0.1571	0.2701
DT	2.4410	0.0157	0.0234	0.0376	0.0583	1.4948	0.0072	0.0701	0.0489	0.1840

Table 4. Individual experiments of FRBS 5-fold for the regression problem.

	Learning					Tuning				
	Rules	*MSE*		*RMSE*		Rules	*MSE*		*RMSE*	
		Training	Test	Training	Test		Training	Test	Training	Test
1	54	0.0171	0.0223	0.1851	0.2112	28	0.0023	0.0038	0.0673	0.0868
2	54	0.0171	0.0223	0.1851	0.2112	28	0.0023	0.0038	0.0673	0.0868
3	54	0.0171	0.0223	0.1851	0.2112	28	0.0023	0.0038	0.0673	0.0868
4	54	0.0171	0.0223	0.1851	0.2112	28	0.0023	0.0038	0.0673	0.0868
5	54	0.0171	0.0223	0.1851	0.2112	28	0.0023	0.0038	0.0673	0.0868
6	54	0.0171	0.0223	0.1851	0.2112	28	0.0023	0.0038	0.0673	0.0868
7	72	0.0119	0.0160	0.1542	0.1791	48	0.0016	0.0023	0.0560	0.0683
8	72	0.0119	0.0160	0.1542	0.1791	48	0.0016	0.0023	0.0560	0.0683
9	72	0.0119	0.0160	0.1542	0.1791	48	0.0016	0.0023	0.0560	0.0683
10	72	0.0119	0.0160	0.1542	0.1791	48	0.0016	0.0023	0.0560	0.0683
11	72	0.0119	0.0160	0.1542	0.1791	48	0.0016	0.0023	0.0560	0.0683
12	72	0.0119	0.0160	0.1542	0.1791	48	0.0016	0.0023	0.0560	0.0683
13	96	0.0161	0.0122	0.1794	0.1564	57	0.0013	0.0031	0.0513	0.0787
14	96	0.0161	0.0122	0.1794	0.1564	57	0.0013	0.0031	0.0513	0.0787
15	96	0.0161	0.0122	0.1794	0.1564	57	0.0013	0.0031	0.0513	0.0787
16	96	0.0161	0.0122	0.1794	0.1564	57	0.0013	0.0031	0.0513	0.0787
17	96	0.0161	0.0122	0.1794	0.1564	57	0.0013	0.0031	0.0513	0.0787
18	96	0.0161	0.0122	0.1794	0.1564	57	0.0013	0.0031	0.0513	0.0787
19	72	0.0168	0.0191	0.1835	0.1955	43	0.0019	0.0021	0.0617	0.0653
20	72	0.0168	0.0191	0.1835	0.1955	43	0.0019	0.0021	0.0617	0.0653
21	72	0.0168	0.0191	0.1835	0.1955	43	0.0019	0.0021	0.0617	0.0653
22	72	0.0168	0.0191	0.1835	0.1955	43	0.0019	0.0021	0.0617	0.0653
23	72	0.0168	0.0191	0.1835	0.1955	43	0.0019	0.0021	0.0617	0.0653
24	72	0.0168	0.0191	0.1835	0.1955	43	0.0019	0.0021	0.0617	0.0653
25	90	0.0179	0.0206	0.1890	0.2032	61	0.0017	0.0053	0.0587	0.1034
26	90	0.0179	0.0206	0.1890	0.2032	61	0.0017	0.0053	0.0587	0.1034
27	90	0.0179	0.0206	0.1890	0.2032	61	0.0017	0.0053	0.0587	0.1034
28	90	0.0179	0.0206	0.1890	0.2032	61	0.0017	0.0053	0.0587	0.1034
29	90	0.0179	0.0206	0.1890	0.2032	61	0.0017	0.0053	0.0587	0.1034
30	90	0.0179	0.0206	0.1890	0.2032	61	0.0017	0.0053	0.0587	0.1034
Mean	76.8	0.0160	0.0181	0.1783	0.1891	47.4	0.0018	0.0033	0.0590	0.0805
SD	15,1462	0.0021	0.0036	0.0126	0.0198	11.8018	0.0003	0.0012	0.0055	0.0140

References

1. Albers, S.: Energy-efficient algorithms. Communications of the ACM 53(5), 86–96 (2010)
2. Albers, S.: Algorithms for dynamic speed scaling. In: Symposium on Theoretical Aspects of Computer Science (STACS2011). vol. 9, pp. 1–11 (2011)
3. Álvarez, J.D., Risco-Martín, J.L., Colmenar, J.M.: Multi-objective optimization of energy consumption and execution time in a single level cache memory for embedded systems. Journal of Systems and Software 111, 200 – 212 (2016)
4. ARM946E-S TM: Technical reference manual. http://www.arm.com/products/processors/-classic/arm9/ arm946.php (2014)
5. Augustine, J., Irani, S., Swamy, C.: Optimal power-down strategies. SIAM Journal on Computing 37(5), 1499–1516 (2008)
6. Bansal, N., Kimbrel, T., Pruhs, K.: Speed scaling to manage energy and temperature. Journal of the ACM (JACM) 54(1), 3 (2007)
7. Benini, L., Bogliolo, A., De Micheli, G.: A survey of design techniques for system-level dynamic power management. IEEE Trans. Very Large Scale Integr. Syst. 8(3), 299–316 (jun 2000), http://dx.doi.org/10.1109/92.845896
8. Brooks, D.M., Bose, P., Schuster, S.E., Jacobson, H., Kudva, P.N., Buyuktosunoglu, A., Wellman, J., Zyuban, V., Gupta, M., Cook, P.W.: Power-aware microarchitecture: Design and modeling challenges for next-generation microprocessors. IEEE Micro 20(6), 26–44 (2000)
9. Burke, E.K., Gustafson, S., Kendall, G.: Diversity in genetic programming: an analysis of measures and correlation with fitness. IEEE Transactions on Evolutionary Computation 8(1), 47–62 (Feb 2004)
10. Cotta, C., Fernández-Leiva, A., de Vega, F.F., Chávez, F., Merelo, J., Castillo, P., Bello, G., Camacho, D.: Ephemeral computing and bioinspired optimization – challenges and opportunities. In: 7th International Joint Conference on Evolutionary Computation Theory and Applications. pp. 319–324. Scitepress, Lisboa, Portugal (2015)
11. Daz, J., Chvez, F., Garca, J., Castillo, P., de Vega, F.F.: Estimating energy consumption in evolutionary algorithms by means of frbs. towards energy-aware bioinspired algorithms. In: Proceedings of the 18th EPIA Conference on Artificial Intelligence. pp. 1–12. Oporto, Portugal (2017)
12. Fadaee, M., Radzi, M.: Multi-objective optimization of a stand-alone hybrid renewable energy system by using evolutionary algorithms: A review. Renewable and Sustainable Energy Reviews 16(5), 3364 – 3369 (2012)
13. Gacto, M.J., Alcalá, R., Herrera, F.: A multi-objective evolutionary algorithm for an effective tuning of fuzzy logic controllers in heating, ventilating and air conditioning systems. Applied Intelligence 36(2), 330–347 (2012)
14. Gao, Y., Guan, H., Qi, Z., Hou, Y., Liu, L.: A multi-objective ant colony system algorithm for virtual machine placement in cloud computing. Journal of Computer and System Sciences 79(8), 1230 – 1242 (2013)
15. Graff, M., Poli, R.: Performance models for evolutionary program induction algorithms based on problem difficulty indicators. In: European Conference on Genetic Programming. pp. 118–129. Springer (2011)
16. Hameed, A., Khoshkbarforoushha, A., Ranjan, R., Jayaraman, P.P., Kolodziej, J., Balaji, P., Zeadally, S., Malluhi, Q.M., Tziritas, N., Vishnu, A., Khan, S.U., Zomaya, A.: A survey and taxonomy on energy efficient resource allocation techniques for cloud computing systems. Computing 98(7), 751–774 (Jul 2016)
17. He, J., Chen, T., Yao, X.: On the easiest and hardest fitness functions. IEEE Transactions on Evolutionary Computation 19(2), 295–305 (2015)
18. Herbert, S., Marculescu, D.: Analysis of dynamic voltage/frequency scaling in chip-multiprocessors. In: Low Power Electronics and Design (ISLPED), 2007 ACM/IEEE International Symposium on. pp. 38–43. IEEE (2007)

19. Herrera, F.: Genetic fuzzy systems: taxonomy, current research trends and prospects. Evolutionary Intelligence 1(1), 27–46 (2008)
20. HOOPER, A.: green-computing (2008)
21. Koza, J.R.: Genetic Programming: On the Programming of Computers by Means of Natural Selection. MIT Press, Cambridge, MA, USA (1992)
22. Langdon, W.B., Poli, R.: Foundations of Genetic Programming. Springer-Verlag New York, Inc., New York, NY, USA (2002)
23. Linares-Vsquez, M., Bavota, G., Crdenas, C.E.B., Oliveto, R., Di Penta, M., Poshyvanyk, D.: Optimizing energy consumption of guis in android apps: A multi-objective approach. In: Proceedings of the 2015 10th Joint Meeting on Foundations of Software Engineering. pp. 143–154. ESEC/FSE 2015, ACM, New York, NY, USA (2015)
24. Logenthiran, T., Srinivasan, D., Shun, T.Z.: Demand side management in smart grid using heuristic optimization. IEEE Transactions on Smart Grid 3(3), 1244–1252 (Sept 2012)
25. Martínez, Y., Trujillo, L., Legrand, P., Galván-López, E.: Prediction of expected performance for a genetic programming classifier. Genetic Programming and Evolvable Machines 17(4), 409–449 (2016)
26. Merkel, A., Bellosa, F.: Memory-aware scheduling for energy efficiency on multicore processors. In: Proceedings of the 2008 Conference on Power Aware Computing and Systems. pp. 1–1. HotPower'08, USENIX Association, Berkeley, CA, USA (2008)
27. Nesmachnow, S., Luna, F., Alba, E.: An empirical time analysis of evolutionary algorithms as c programs. Software: Practice and Experience 45(1), 111–142 (2015), http://dx.doi.org/10.1002/spe.2217
28. Sharma, R.K., Bash, C.E., Patel, C.D., Friedrich, R.J., Chase, J.S.: Balance of power: dynamic thermal management for internet data centers. IEEE Internet Computing 9(1), 42–49 (Jan 2005)
29. Takagi, T., Sugeno, M.: Fuzzy identification of systems and its applications to modeling and control. IEEE transactions on systems, man, and cybernetics (1), 116–132 (1985)
30. Trujillo, L., Martínez, Y., Galván-López, E., Legrand, P.: Predicting problem difficulty for genetic programming applied to data classification. In: Proceedings of the 13th annual conference on Genetic and evolutionary computation. pp. 1355–1362. ACM (2011)
31. Trujillo, L., Martínez, Y., Galván López, E., Legrand, P.: A comparative study of an evolvability indicator and a predictor of expected performance for genetic programming. In: Proceedings of the 14th annual conference companion on Genetic and evolutionary computation. pp. 1489–1490. ACM (2012)
32. Vasicek, Z., Sekanina, L.: Evolutionary approximation of complex digital circuits. In: Proceedings of the Companion Publication of the 2015 Annual Conference on Genetic and Evolutionary Computation. pp. 1505–1506. GECCO Companion '15, ACM, New York, NY, USA (2015)
33. de Vega, F.F., Chávez, F., Díaz, J., García, J.A., Castillo, P.A., Merelo, J.J., Cotta, C.: A Cross-Platform Assessment of Energy Consumption in Evolutionary Algorithms, pp. 548–557. Springer International Publishing, Cham (2016)
34. Vega, F., Chávez, F., Díaz, J., A. García, J., Castillo, P., J. Merelo, J., Cotta, C.: A cross-platform assessment of energy consumption in evolutionary algorithms 9921, 548–557 (09 2016)
35. Wang, W., Zwolinski, M.: An improved instruction-level energy model for risc microprocessors. In: Ph. D. Research in Microelectronics and Electronics (PRIME), 2013 9th Conference on. pp. 349–352. IEEE (2013)

Josefa Díaz Álvarez obtained a M.S. degree in Computer Engineering in 2007 from the University of Extremadura (UNEX), and received a Ph.D. degree in 2017 from the Complutense University of Madrid (UCM). She is currently Assistant Professor of Computer Science, at the Computer and Technology Department at the Merida campus of the

UNEX. Her current research interests focus on Evolutionary Algorithms. A special interest in design methodologies for the optimization of hardware resources, dynamic memory management and memory hierarchy optimizations for embedded systems. She is also interested on the application of artificial intelligence for the study of neurodegenerative diseases.

Franciso Chávez received his Ph.D. in Computer Science from the University of Extremadura in 2012. Since 2001, he has belonged to the Department of Engineering of Computer Science and Telematics Systems of the University of Extremadura. He is currently an Assistant Professor and belongs to the Artificial Evolution Research Group. It has over 50 national and international publications. He has worked as a researcher in several research projects granted by the Spanish Government, he has been a Principal Investigator in two of them EPHEMCH and DEEPBIO. He has also worked as a researcher in several research projects awarded by the Autonomous Community of Extremadura. He recently leads a project granted by the Autonomous Community of Extremadura, where he performs the tasks of principal investigator. Finally, he has worked on several projects with companies in the sector in the field of technology transfer with companies. He has three patents and his lines of research focus on systems based on fuzzy rules and diffuse genetic systems, image processing through deep learning and massive data processing.

Pedro A. Castillo received his Ph.D. degree in 2000 from the University of Granada. He has been visiting researcher at the Napier University (July, 1998) and at the Santa Fe Institute (September, 2000). He was a Teaching Assistant at Computer Science Department of the University of Jan, Spain. He is currently Associate Professor at the Department of Computer Architecture and Technology of the University of Granada (Spain). His main research interests are in the fields of bio-inspired systems, hybrid system and combination of evolutionary algorithms and neural networks.

Juan Angel García received his Ph.D. in Electrical, electronic and Control Engineering from the National Distance Education University (UNED). He is currently assistant professor in the University of Extremadura. His research interests are in the field of numerical analysis, construction of algorithms, and robust control.

Francisco J. Rodriguez received his Ph.D in Computer Science in 2012 from the University of Granada. He is currently assistant professor in the Department of Computer Science in the University of Burgos. He has over 40 publications in prestigious conferences and journals, 17 of which in JCR journals. His main research interests are focused on evolutionary computation, hybrid metaheuristics, and machine learning.

Francisco Fernández de Vega is associate professor of Computer Architecture at the University of Extremadura (UEX), Spain. Received his PhD in 2001, best PhD award in Engineering, UEX. He has been UEX CIO from 2005-2007. He has published more than 250 papers in international conferences and JCR journals. He has received several research awards, including ACM GECCO Evolutionary Art Design and Creativity Competition award in 2013, Amsterdam. His interdisciplinary research on Computational Intelligence and Creativity has allowed to display his evolutionary art works on art galleries

around the world, including Vancouver, Paris and New York. His work was selected finalist in the ”Show Your World” international art competition, New York 2017. Hi is co-chair of the Creative Intelligence Task Force, IEEE Computational Intelligence Society.

Received: January 10, 2018; Accepted: September 1, 2018.