Future Generation Computer Systems 167 (2025) 107753

Contents lists available at ScienceDirect

Future Generation Computer Systems

journal homepage: www.elsevier.com/locate/fgcs

Energy–time modelling of distributed multi-population genetic algorithms with dynamic workload in HPC clusters

Juan José Escobar [a,*], Pablo Sánchez-Cuevas [b], Beatriz Prieto [c], Rukiye Savran Kızıltepe [d,e], Fernando Díaz-del-Río [b], Dragi Kimovski [f]

[a] Department of Software Engineering, CITIC, University of Granada, Spain
[b] Department of Computer Architecture and Technology, University of Seville, Spain
[c] Department of Computer Engineering, Automation, and Robotics, CITIC, University of Granada, Spain
[d] Department of Software Engineering, Karadeniz Technical University, Turkey
[e] Department of Software Engineering, Ankara University, Turkey
[f] Institute of Information Technology, University of Klagenfurt, Austria

ARTICLE INFO

Keywords:
Energy–time modelling
Heterogeneous clusters
Distributed computing
Parameter optimisation
Task scheduling
Genetic algorithms

ABSTRACT

Time and energy efficiency is a highly relevant objective in high-performance computing systems, with high costs for executing the tasks. Among these tasks, evolutionary algorithms are of consideration due to their inherent parallel scalability and usually costly fitness evaluation functions. In this respect, several scheduling strategies for workload balancing in heterogeneous systems have been proposed in the literature, with runtime and energy consumption reduction as their goals. Our hypothesis is that a dynamic workload distribution can be fitted with greater precision using metaheuristics, such as genetic algorithms, instead of linear regression. Therefore, this paper proposes a new mathematical model to predict the energy–time behaviour of applications based on multi-population genetic algorithms, which dynamically distributes the evaluation of individuals among the CPU–GPU devices of heterogeneous clusters. An accurate predictor would save time and energy by selecting the best resource set before running such applications. The estimation of the workload distributed to each device has been carried out by simulation, while the model parameters have been fitted in a two-phase run using another genetic algorithm and the experimental energy–time values of the target application as input. When the new model is analysed and compared with another based on linear regression, the one proposed in this work significantly improves the baseline approach, showing normalised prediction errors of 0.081 for runtime and 0.091 for energy consumption, compared to 0.213 and 0.256 shown in the baseline approach.

1. Introduction

To deliver higher performance, modern processors can no longer rely on increasingly complex designs. In fact, the industry has focused on creating more efficient alternatives to solve this problem. In this sense, Instruction Level Parallelism (ILP) is no longer the only resource available to the designer and a new trend has appeared towards architectures that allow Data Level Parallelism (DLP) and Thread Level Parallelism (TLP) [1]. Nevertheless, these techniques are insufficient, since the vast amount of data requiring processing nowadays surpasses the computing capabilities of any parallel architecture. This is why distributed computing has gained importance for years through the use of clusters, where multiple computing nodes are grouped to cooperate in solving a specific problem.

Another key factor that has motivated the creation of these new computer architectures is energy consumption, since it has become a considerable problem in the case of single-core processors [2]. Currently, in the context of Big Data and High-Performance Computing (HPC), energy must be minimised to lower service costs and cushion the environmental impact of massive computing [3–5]. In this sense, many proposals have been published to address the problem. One of them is to use heterogeneous clusters, since they offer multiple levels of parallelism through a set of interconnected nodes which contain multiple Central Processing Units (CPUs) and accelerators, such as Graphics Processing Units (GPUs). It has been demonstrated that these platforms can increase the performance of different applications while reducing the energy cost [6–8]. One of the most distinctive features of distributed computing on heterogeneous clusters is workload distribution, as it has

* Corresponding author.
E-mail addresses: jjescobar@ugr.es (J.J. Escobar), psanchez4@us.es (P. Sánchez-Cuevas), beap@ugr.es (B. Prieto), rskiziltepe@ankara.edu.tr (R.S. Kızıltepe), fdiaz@us.es (F. Díaz-del-Río), Dragi.Kimovski@aau.at (D. Kimovski).

https://doi.org/10.1016/j.future.2025.107753
Received 10 July 2024; Received in revised form 29 January 2025; Accepted 1 February 2025
Available online 10 February 2025

the potential to lead to unbalanced workload [9]. Therefore, deciding which devices the tasks should be assigned at runtime in systems that include resources with different performance and energy efficiencies is crucial.

A topic of research in this direction is energy–time modelling. The heterogeneous clusters with dynamic workload distribution require dealing with non-linear models to predict their energy–time behaviour accurately. This subject consists of the definition and evaluation of mathematical models that estimate the behaviour in performance and energy consumption of a given computing platform. Then, the parameters involved in the resulting model are fitted by regression or optimisation methods by using experimental data as input. This allows us to take advantage of the newly fitted model to study how each parameter affects performance and energy consumption and, therefore, to tackle costs because it can help designers optimise the behaviour of a given application for a specific platform. For this reason, this paper focuses on building a model to deal with this issue. Specifically, the proposed model copes with the energy–time modelling of applications based on multi-population genetic algorithms that run on heterogeneous clusters with multiple nodes. Genetic Algorithms (GAs) play a fundamental role in dealing with complex problems, such as those that frequently appear in Artificial Intelligence (AI) and Data Science, where, in general, the data is stochastic and high-dimensional. The traditional optimisation tools fail for such problems due to their complexity [10]. GAs, by their nature, require iteratively performing multiple operations with large amounts of data, which implies that they are easily parallelizable in parallel and distributed architectures. Given the high computational cost of GAs, it is necessary to adequately plan the workload distribution between the different computing nodes, especially when they are heterogeneous. In this way, a trade-off can be found between the energy–time performance of the application and the quality of the solutions to the problem to be solved.

The new model presented here aims to improve a previous model for the same type of applications, using another genetic algorithm to optimise its parameters instead of linear regression. Although there are numerous optimisation methods in the literature, a specific GA has been chosen since it offers several significant advantages for the fitting of mathematical models. GAs are highly effective at handling non-linear and high-dimensional problems, such as those encountered in dynamic workload distributions in heterogeneous clusters. This allows them to fit models where gradient-based methods like Gradient Descent or Newton–Raphson may struggle due to non-convex or highly complex cost surfaces. One of the main strengths of GAs is their independence from gradients, making them useful in scenarios where the cost function is non-differentiable or computationally expensive to evaluate. Furthermore, GAs are well-known for their ability to perform global exploration of the search space, enabling them to avoid local optima. This is particularly valuable in problems with multiple local minima, where methods like Bayesian optimisation or Newton–Raphson may become trapped in suboptimal solutions. Another key advantage of GAs is their dynamic adaptability, allowing them to adjust to changes in the problem, such as variations in workload distributions in heterogeneous clusters. In contrast, more static approaches, like Grid Search or convex programming, lack this capability. Finally, GAs provide remarkable flexibility in model structure, which is crucial when handling parameters with complex constraints and non-linear dependencies. This type of flexibility is challenging to achieve with other approaches, such as certain machine learning-specific methods like Hyperband or Tree-structured Parzen Estimator (TPE), which do not directly address these issues.

In summary, the main contributions of this work are as follows:

- Propose a new mathematical energy–time model for distributed multi-population GAs, executed in heterogeneous clusters, which is compared with another pre-existing model taken as a baseline. Despite the existence of some previous works on HPC modelling, those that simultaneously take into account performance and energy consumption are not common, and even less so for parallel and distributed GAs. Moreover, although the model developed here is designed for multi-population applications with specific features, most bioinspired algorithms share a great similarity in their structure so that the model could be easily adapted to them.
- Tackle the vital role that dynamic workload represents by modelling the dynamic distribution of subpopulations among CPU–GPU computing devices.
- Apply a GA to fit the set of parameters comprised in the proposed mathematical model with the objective of handling its non-linearity. The objective is to demonstrate that the use of metaheuristics, in this case, a GA, is more suitable than linear regressions to fit the model parameters in applications with dynamic workload distributions.
- Successfully validate the proposed model, obtaining more accurate results than the baseline one. Moreover, we also analyse the weight on time and energy consumption of the components of the evaluated cluster, while identifying the critical ones.

The rest of the article is structured as follows: Section 2 reviews different works in the literature related to energy–time modelling and energy-aware computing in HPC systems; an overview of genetic algorithm is provided in Section 3; the proposed mathematical model and the non-linear optimisation method used to fit the model are detailed in Sections 4 and 5, respectively; the description of the modelled application and the corresponding experimental results are shown in Section 6; the paper's conclusions are provided in Section 7. Finally, Appendix A includes a glossary of the mathematical terms involved in the proposed model and Appendix B defines the acronyms that appear throughout this paper.

2. Related work

Given the great importance of a good balance between performance and energy efficiency in computing, different methods and solutions have been proposed to address this problem. The works [4,11] compile an extensive list of papers on HPC and categorises them based on compute device type, optimisation metrics, and energy reduction methods. The compatibility of employing techniques for HPC systems, such as Dynamic Voltage Scaling (DVS), Dynamic Frequency Scaling (DFS), and Dynamic Voltage and Frequency Scaling (DVFS), to find the best combination of computing performance and energy consumption has also been discussed. Studies like [12–17] are well-known examples following this trend.

As mentioned in Section 1, modelling an application in HPC is essential to have control over its energy–time behaviour. In the case of parallel architectures, these simple but wide-ranging models based on extensions of Amdahl's Law serve as a basis for seeing the influence of task distribution on many-core systems [18,19], and cloud computing [20]. Other approaches calculate the runtime and energy consumption as the weighted sum of the frequency of a set of system events, which are measured by hardware counters. Since low-level events such as cache misses, floating-point operations, or interrupts directly cause a significant effect on performance and energy efficiency, this alternative approach has become very attractive [21]. Although such models can be built for a specific application [22], they can become general models for a given set of problems with the correct modelling. For example, in [23], an estimation of the time consumed to evaluate an individual of the GA is used to determine the population size among a heterogeneous cluster without being dependent on the concrete problem or application.

Regarding models that attempt to predict the energy consumption of computing architectures, surveys such as [3,24,25], summarise a substantial set of possible models. Most of them are based on linear

expressions, while only a few use non-linear models to describe the hierarchical nature of system components. This is because linear models generally use linear regression for such a task, while non-linear models use optimisation methods [26]. Since it is more common to find applications in HPC systems executed on homogeneous systems with static workload distributions, linear models have been more widespread. In a broad sense, linear models have the advantage of interpretability so that the designer can understand each component's weight in the HPC system's final costs. This allows the developer to tweak the software to the algorithms to reduce their energy cost and increase performance [27]. In contrast, non-linear models, such as those optimised using metaheuristics, could be less interpretable but more accurate and can be used on various problems.

One of the main features that affect overall efficiency in distributed HPC systems is the scheduling and distribution of the workload. Deciding where the tasks should be assigned is an important factor in heterogeneous clusters with different performance and power requirements [28,29]. It is important to differentiate between procedures that plan the distribution of jobs between devices at the operating system level, considering jobs independent of each other, and procedures that distribute tasks associated with a specific application and whose execution must be carried out in a pre-established and synchronised order. In the latter case, numerous heuristics have been previously proposed [30,31]. Wang et al. [30] propose an energy-aware task scheduling framework to minimise the energy consumption of heterogeneous and geo-distributed MapReduce clusters. The framework constructs a reasonable task list, considering deadlines, the number of assigned task slots, and possible processing times. Tasks are then scheduled into promising slots of rack-local servers, cluster–local servers, and remote servers, which significantly improve data locality. After task and slot assignment, available slots in clusters are updated to improve server resource utilisation by fuzzy logic, based on the current CPU, memory, and bandwidth utilisation. Experimental results showed that the proposed heuristic reduces energy consumption compared to existing algorithms by varying the total number of slots.

In the work [31], the authors address the problem of energy efficiency in heterogeneous edge computing systems for a large number of latency-sensitive applications. Indeed, they present an efficient technique to minimise the energy overhead of time-constrained applications modelled by Directed Acyclic Graphs (DAGs). The technique is developed in three phases: firstly, they design a new method for calculating task priority and propose an energy-aware scheduling algorithm based on ant colonies to obtain a preliminary scheduling result; secondly, taking into consideration the slack time between tasks and their deadlines, they propose a proportional downward recovery slack algorithm to further reduce the energy overhead by using the DVFS technique; finally, taking into consideration the slack time between tasks executed on the same processor, they propose a proportional upward and downward recovery algorithm to reduce the power overhead using the DVFS technique again. The simulated results indicated that the presented technique is highly efficient in reducing power overhead compared to existing techniques using randomly generated and real-world benchmarks with different characteristics.

Despite the above, more precise heuristics can be defined when the tasks are specific or identifiable. For example, in one of our previous works [32], a model is fitted to predict the energy–time behaviour of the application, taking into account both the input parameters of the algorithm, as well as the clock frequencies of the CPU–GPU devices. Later, in the work [33], the model was extended to allow predicting the behaviour of a parallel and distributed genetic algorithm when using heterogeneous clusters to evaluate multiple subpopulations of individuals. From both works, it is revealed that adequate scheduling can simultaneously reduce runtime and energy consumption by efficiently distributing the tasks among the different processing elements. Despite the good results, both models used linear regression to fit the parameters, which, as discussed above, is not ideal for dynamic workload distributions.

3. Overview of genetic algorithms

As stated in a recent work [34], GA is one of the most popular optimisation algorithms currently employed in real-world applications, including image processing, programming, clustering, software engineering, natural language processing, recommendation systems, and scheduling tasks. The interest in and importance of this kind of algorithm and evolutionary computation has remained strong over the years. In the last four years alone, more than 73,000 publications on GA have been included in the Web of Science Platform (Clarivate).

GAs mimic the process of natural selection, where each new generation is expected to be better than its parents. The goal of these algorithms is to find a solution to the problem using the individuals in the population as candidates for the solution. Each of them is represented by a chromosome that encodes the parameters (genes) of the problem to be solved. All individuals are evaluated according to a fitness function, obtaining a score related to the quality of their solutions. The better the score, the greater the degree of approximation to the optimal solution and the probability of reproduction for that individual. There are several ways to determine the individuals (parents) that will generate offspring (children). One of the most common is the binary tournament, in which some individuals are randomly chosen and the two best are selected. After that, the crossover and mutation operators are applied. The crossover operator is responsible for conducting the search across the solution space by combining genes from the parents to create new individuals. On the contrary, the mutation is the modification of some genes in the children's chromosomes. For each gene, the mutation is performed with a fixed probability, usually low, so as not to excessively alter the natural course of evolution. Nevertheless, its use provides diversity in the search space and serves as a measure to avoid falling into local optimum quickly. Finally, the selection operator will choose which individuals will be part of the next generation based on the value of the fitness function. All the above steps are repeated over several generations of individuals (iterations in computational terms) until a stopping condition is met. The objective after each generation is to continuously improve the quality of the solutions until converging on an optimal or near-optimal solution.

A GA can have a more complex structure and involve more operators to improve the search for solutions. For example, there are multi-population models that allow the evolution of several populations simultaneously. In fact, the mathematical model proposed in this work is focused on this type of GAs. The idea is to divide the total population into several subpopulations, or islands, where each one evolves independently. This scheme allows some GAs the possibility of exchanging information between subpopulations every certain number of generations. The process is known as migration, and it is considered as an exclusive genetic operator of multi-population models. The introduction of migration gives the algorithm the ability to exploit the differences between subpopulations, thus allowing genetic diversity. However, the determination of the migration rate and when to migrate is a sensitive matter since an inappropriate value can cause premature convergence. Another concern is the performance. GAs are highly parallelizable since each individual in the population can be independently evaluated. However, the multi-population paradigm allows another level of parallelism since following the same philosophy, the subpopulations evolve independently of each other except at the time of migration. However, the more levels, the more complexity. One of them is the problem of workload balancing: depending on the parallel approach, workload imbalances and consequent loss of performance can occur. Assuming that the time needed to evaluate an individual is the same in all subpopulations, an example of workload imbalance appears when the size of the subpopulations differs. Another imbalance situation is present in heterogeneous computing because the devices responsible for evaluating individuals probably have different computational capabilities. Also the granularity of parallelism must be taken into account since, depending on its type, the cost of communications and workload imbalance can be less or greater, so finding a trade-off between the two is important.

4. The proposed energy–time model

This section exposes the new mathematical model intended to predict the energy–time behaviour of multi-population genetic algorithms, in which a master–worker approach dynamically distributes the evaluation of subpopulations among the CPU–GPU devices of heterogeneous clusters, and performs migrations between subpopulations every certain number of generations. We have taken as a baseline model the one proposed in [33], which we intend to improve. All variables involved in the definition of the new model are summarised in Appendix A.

4.1. Mathematical formulation

The starting hypothesis to create the new model is based on the following differences with the baseline model:

1. The new model is based on considering the computing by CPU–GPU devices instead of by nodes. Modelling a level where the workload granularity is finer provides more accurate results.
2. In the baseline model, the total workload sent to each device was assumed to be known. However, this does not capture the dynamic nature of the application because each device requests workload on demand, so it cannot be exactly known a priori.
3. The baseline model considered that the number of individuals to be evaluated within a device in each generation is equal to the subpopulation size. However, when the genetic algorithm applies the crossover operator to generate new offspring, the number of children will usually be smaller due to the influence of the crossover probability.

It should be noted that although the mutation step modifies the computational workload by varying the number of selected features, it is assumed to be negligible due to its low probability (β), so it has not been included in the new model to simplify its complexity. Taking all of the above into account, the model is developed as follows: as the model proposed here starts from the distribution of subpopulations by devices, the runtime of the jth device of the ith node, necessary to evolve one subpopulation between two migrations, is:

$$\hat{T}^1_{i,j} = \frac{g}{N_{Gm}} \cdot \left\lceil \frac{P_E}{C_{i,j}} \right\rceil \cdot \frac{W_{i,j}}{F_{i,j}} \tag{1}$$

where g is the number of generations and N_{Gm} the number of global migrations. In this way, migration takes place every $\frac{g}{N_{Gm}}$ generations. $C_{i,j}$ is the number of cores in the device running at a clock frequency $F_{i,j}$. Each subpopulation involves P_E evaluations of individuals in parallel, being $W_{i,j}$, the number of clock cycles necessary to evaluate each. As the number of individuals to be evaluated in each generation should be less than the subpopulation size due to the crossover probability, P_E can be estimated as:

$$P_E = S_{Sp} \cdot \alpha = \left(\frac{N}{N_{Sp}} \right) \cdot \alpha \tag{2}$$

where S_{Sp} is the subpopulation size, α is the crossover probability, and N and N_{Sp} are the number of individuals and subpopulations of the genetic algorithm, respectively. Extending Eq. (1), the runtime of the jth device of the ith node necessary to evolve its $N_{Sp_{i,j}}$ subpopulations is:

$$\hat{T}_{i,j} = N_{Sp_{i,j}} \cdot \hat{T}^1_{i,j} \tag{3}$$

Finally, the total runtime taken by the cluster to perform the whole genetic algorithm is modelled as follows:

$$\hat{T}_{cluster} = N_{Gm} \cdot (\hat{T}_{master} + \hat{T}_{com} + \hat{T}_{evo}) \tag{4}$$

being $\hat{T}_{master}$ and $\hat{T}_{com}$ the time overheads that migration imposes on the master node and the switch, respectively. The variable $\hat{T}_{evo}$ corresponds to the time necessary to evolve all N_{Sp} subpopulations between two migrations. In the baseline model, $\hat{T}_{evo}$ was modelled as the time needed by the slowest worker node to evolve its subpopulations: $max\left(\hat{T}_i; \forall i = 1, \ldots, N_{Wk}\right)$, being N_{Wk} the number of available workers:

$$\hat{T}_{cluster} = N_{Gm} \cdot \left(\hat{T}_{master} + \hat{T}_{com} + max\left(\hat{T}_i; \forall i = 1, \ldots, N_{Wk}\right)\right) \tag{5}$$

However, in the new model, parameter $\hat{T}_{evo}$ has been adapted at the device level to capture the dynamic essence of the algorithm. This means that the slowest device between migrations will not always be the same since it depends on the total number of subpopulations it received. For this reason, $\hat{T}_{evo}$ has been estimated using Algorithm 1 presented in Section 4.2 by accumulating the time costs of each device.

Once the time equations have been obtained, the next step is to estimate the energy cost. In this sense, Eqs. (3) and (4) are fundamental in this model since energy consumption corresponds to the product of instantaneous power and runtime. Thus, the energy consumption of the jth device of the ith node between two migrations can be calculated by adding its consumption when active and idle:

$$\hat{E}_{i,j} = Pow_{i,j} \cdot \hat{T}_{i,j} + Pow^{idle}_{i,j} \cdot \left(\hat{T}_{evo} - \hat{T}_{i,j}\right) \tag{6}$$

where $Pow_{i,j}$ and $Pow^{idle}_{i,j}$ are the instantaneous power of the device when active and idle, respectively. The idle time can be calculated by subtracting the time taken by the device, $\hat{T}_{i,j}$, from the total time of computing all the subpopulations, $\hat{T}_{evo}$. If this subtraction result is zero, this device is always computing, which causes a bottleneck for the rest of the devices. Finally, the energy consumption of the entire cluster can be calculated as the sum of the energy of each device and that caused by the overhead of the master node and the switch:

$$\hat{E}_{cluster} = N_{Gm} \cdot \left(Pow_{master} \cdot \hat{T}_{master} + Pow_{switch} \cdot \hat{T}_{com} + \sum_{i,j} \hat{E}_{i,j} \right) \tag{7}$$

being Pow_{master} and Pow_{switch} the instantaneous power of the master node and switch, respectively.

Algorithm 1: Simulated workload distribution implemented in the modelled application.

```
Function WorkloadDistribution(N_Sp, N_Wk, N_D)
    Input  : Number of subpopulations, N_Sp
    Input  : Number of worker nodes in the cluster, N_Wk
    Input  : List with the number of devices of each node, N_D
    Output: Predicted runtime to evolve all subpopulations, T̂_evo
    Output: List with the estimated number of subpopulations
            assigned to each device, N_{Sp_{i,j}}, ∀i = 1,...,N_Wk,
            ∀j = 1,...,N_{D_i}
    N_{Sp_{i,j}} ← 0
    L ← Init list with rows of type [node,device,total_time = 0]
    while N_Sp > 0 do
        i ← getColumn(L[0], "node")
        j ← getColumn(L[0], "device")
        t ← getColumn(L[0], "total_time")
        setColumn(L[0], "total_time", t + T̂^1_{i,j})
        L ← sortAscendingOrder(L, "total_time")
        N_Sp ← N_Sp − 1
        N_{Sp_{i,j}} ← N_{Sp_{i,j}} + 1
    end
    T̂_evo ← getColumn(L[len(L) − 1], "total_time")
    return [T̂_evo, N_{Sp_{i,j}}]
End
```

4.2. Workload distribution computation

The model proposed in this work reproduces the dynamic behaviour of a heterogeneous system. Since the total workload assigned to each device, $N_{Sp_{i,j}}$, is dynamic, it is necessary to implement a simulation

of the workload distribution within the model to estimate its value and that of the variable $\hat{T}_{evo}$. For this purpose, Algorithm 1 has been developed. The algorithm considers a list of tuples, L, which records for each device the accumulated time of evaluating subpopulations (Line 3). The main loop from Lines iterates over the input subpopulations, assigning one subpopulation to the device at the top of the list, L_0 (Line 7). The reason is that this device is the one that computes the least and, therefore, is idle. Then, the runtime $\hat{T}^1_{i,j}$ necessary to evolve the subpopulation is added to the accumulated runtime of the device (Line 8). Next, the list is sorted by the "total_time" column in ascending order, and the subpopulation of this iteration is discarded (Line). After finishing the loop, $\hat{T}_{evo}$ is obtained from the last element of the list, $L\,[len\,(L) - 1]$, since it is the device that has accumulated the maximum time (Line 13). The algorithm ends in Line 14 by returning $\hat{T}_{evo}$ and the number of subpopulations computed by each device, $N_{Sp_{i,j}}$.

Although this method covers dynamic behaviour, one crucial feature appears, namely non-linearity. This is because devices with different performances compete to request new tasks in a heterogeneous cluster with a non-static workload distribution. Therefore, although the energy–time costs can be modelled with near-linear expressions at the device level, the states of each device (active or idle) are strongly related to this distributed competition. In this context, our approach is not limited to efficiently distribute the workload but also aims to accurately understand and predict system behaviour in terms of execution time and energy consumption. The aforementioned variability in the computational capabilities of the devices, along with the nature of genetic algorithms, introduces complexities that a coarse-grain prediction model cannot fully capture. Accurate predictions are essential to avoid bottlenecks, unbalanced workload, and inefficient allocations that could increase energy costs and degrade performance. Thus, the proposed model offers granular predictions that complement the dynamic workload distribution algorithm. This is crucial for designing more efficient systems, as it allows fine-tuning of specific aspects such as node-to-node migrations or the allocation of subpopulations based on the profile of each device.

5. Model fitting method

A genetic algorithm has been used to fit the model's parameters due to its speed in finding acceptable solutions in a reasonable time [35,36]. A total of 22 parameters must be fitted, which are defined as unknowns and are associated with the cost of different cluster resources. The only known input variable is the number of subpopulations used in the evolutionary algorithm, N_{Sp}. The fitting is divided into two independent executions of the GA: first, 2 parameters related to time and 6 to workload are fitted. Subsequently, the 14 energy parameters are fitted using the fitted values of the previous step. Each parameter corresponds to a chromosome gene, which has a variable length since it depends on the type of fitting run. In addition, the number of genes also depends on the platform to be modelled, since for each device, its parameters $W_{i,j}$, $Pow_{i,j}$, and $Pow^{idle}_{i,j}$ must be added. The chromosome representation for time/workload and energy, c_{TW} and c_E, respectively, are defined as:

$$c_{TW} = \left(W_{1,1}, \ldots, W_{N_{Wk},N_{D_{Wk}}}, \hat{T}_{com}, \hat{T}_{master}\right) \tag{8}$$

$$c_E = \Big(Pow_{1,1}, \ldots, Pow_{N_{Wk},N_{D_{Wk}}}, Pow^{idle}_{1,1}, \ldots, Pow^{idle}_{N_{Wk},N_{D_{Wk}}}, Pow_{master}, Pow_{switch} \Big) \tag{9}$$

The cost functions for time and energy, $RMSE_T$ and $RMSE_E$, respectively, are defined as the Root-Mean-Square Error (RMSE) between the predicted and experimental measurements:

$$RMSE_T = \sqrt{\frac{1}{32} \cdot \sum_{N_{Sp}=1}^{32} \left(\hat{T}_{cluster} - T_{cluster}\right)^2} \tag{10}$$

Table 1
Input values of the GA used to fit the model. SBB: Simulated Binary Bounded.

Individuals	Number	120
	Chromosome representation	Real-valued
Evolution	Number of generations	100
	Inconsistency penalty	10^9
Crossover	Type	SBB
	Probability	0.7
Mutation	Type	Polynomial
	Probability	0.01

$$RMSE_E = \sqrt{\frac{1}{32} \cdot \sum_{N_{Sp}=1}^{32} \left(\hat{E}_{cluster} - E_{cluster}\right)^2} \tag{11}$$

where $T_{cluster}$ and $E_{cluster}$ corresponds to the measured experimental data. In this way, both fittings aim to find a set of parameter values such that the predictions made with the fitted model are significantly similar to the real experimental values. To maintain consistency between solutions, the GA only accepts solutions that meet the following restrictions:

$$R_1 : Pow^{idle}_{i,j} < Pow_{i,j} \leq TDP_{i,j} \cdot 1.5 \tag{12}$$

$$R_2 : Pow_{master} \leq TDP_{master} \cdot 1.5 \tag{13}$$

$$R_3 : Pow_{switch} \leq TDP_{switch} \cdot 1.5 \tag{14}$$

The R_1 constraint ensures that the instantaneous power of an idle device is lower than when the device is active. Moreover, all these restrictions have in common that the instantaneous power of a device does not exceed 150% of its Thermal Design Power (TDP). This decision is due to the fact that the TDP of a device can be exceeded under certain circumstances. For example, in Intel [37] processors, this value can be exceeded during Intel® Turbo Boost or certain workload types such as Intel® Advanced Vector Extensions (Intel® AVX) for a limited time, until the processor hits a thermal throttle temperature, or until the processor hits a power delivery limit. However, since manufacturers do not usually indicate the maximum peak energy consumption of their devices, a 150% restriction for the TDP has been established since Hennessy et al. state in [1] that its value could be increased by up to 1.5 times.

Finally, for the sake of reproducibility, the configuration for the GA can be found in Table 1, which uses a real encoding for the optimised model parameters.

6. Experimental work

The main tasks addressed in this section are to evaluate the fitting of the proposed model, to compare it with the one provided in [33], and to validate it. To make a fair comparison, the application to be modelled and the energy–time experimental data used in this work to fit the model are the same as those used in the baseline paper. The source code of the modelled application can be found in [38].

6.1. Experimental methodology

The evaluation pipeline is as follows, where Points 1 and 2, necessary to obtain the experimental data, were already carried out in [33]:

1. Run the modelled application multiple times, varying the number of subpopulations from 1 to 32. The rest of the input values remain constant (see Table 2). All experiments are repeated 20 times to obtain more reliable measurements on the application's behaviour.

Table 2
Input values of the NSGA-II algorithm used in the modelled application.

Individuals	Number (N) Chromosome representation	3,840 Binary
Subpopulations	Number (N_{Sp}) Size (S_{Sp})	1 to 32 N/N_{Sp}
Evolution	Number of generations (g) Global migrations (N_{Gm})	150 5
Crossover	Type Probability (α)	Uniform 0.75
Mutation	Type Probability (β)	Bit-flip 0.0025

2. While the application is running, both the runtime and the energy consumption of the cluster are measured. Time is measured as the time taken by the algorithm to finish the evolution of all the subpopulations after g generations, including the recombination of subpopulations carried out by the master node. Energy has been measured using a physical wattmeter for each cluster node.
3. Finally, the experimental energy–time data collected are used by the GA proposed in Section 5 to fit the model parameters. As in Point 1, the algorithm is executed 20 times due to its stochastic behaviour.

6.2. Experimental setup

The cluster used in [33] to obtain the experimental energy–time data contains four heterogeneous Non-Uniform Memory Access (NUMA) nodes that execute CentOS (v7.4.1708). The modelled application is coded in *C++* and has been compiled with the GNU Compiler Collection (GCC) v4.8.5 and optimisation level `-O2`. The OpenMPI library v1.10.7 supports the Message Passing Interface (MPI) v3.0.0. The procedure evaluates 3840 individuals, distributed between 1 to 32 subpopulations (depending on the run), along 150 generations. During execution, Node 1 is dedicated to the master process while others act as workers. The energy consumption was measured for each node and the switch using a wattmeter that calculates both instantaneous power (W) and accumulated energy (W · h) every second. The GA developed in this work to fit the model with the collected energy–time data has been developed by using the Distributed Evolutionary Algorithms in Python (DEAP) library [39]. It is worth mentioning that the versions of CentOS, MPI and GCC are relatively old. This is because we want to compare the results under the same experimental conditions as those used in the baseline paper, several years ago (2019).

6.3. Use case application

As mentioned at the beginning of Section 6, the application used here to evaluate the new model is the same as the one modelled in [33]. It corresponds to a *wrapper* approach where a Non-dominated Sorting Genetic Algorithm (NSGA-II) [40] evolves one or multiple subpopulations of individuals along several generations. The application includes a parallel master–worker scheduler that dynamically distributes individuals among the CPU–GPU devices of each computing node. It deals with an Electroencephalogram (EEG) classification problem in which the individuals codify different alternatives for feature selection and are evaluated through K-means algorithm. As the fitness evaluation is independent for each individual, the individuals are distributed among the computing devices according to a master–worker scheme too, which provides up to four parallelism levels depending on the device used to perform the task. The different levels of parallelism and workload distribution are summarised below. For more details, see [33]:

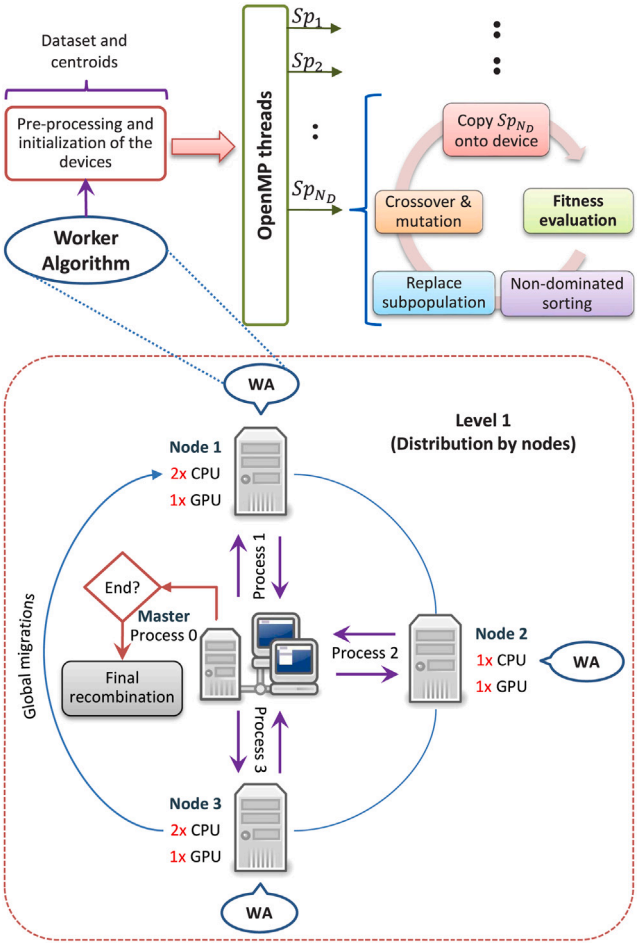


Fig. 1. The use case application. MPI-OpenMP scheme that shows the first and second parallelism levels.

1. **First level**: distribution of subpopulations among the cluster nodes (Fig. 1, bottom). All communications between nodes are done through message-passing with the MPI standard.
2. **Second level**: dynamic distribution of subpopulations among devices of the same node by using OpenMP threads (Fig. 1, top: fitness evaluation, which is detailed in Fig. 2). If there is only one subpopulation, the scheduler will distribute its individuals dynamically to each device. Currently, CPU and GPU devices are supported.
3. **Third level**: dynamic distribution of individuals among CPU cores and the GPU Compute Units (CUs) (Fig. 2). In other words, each CU/core is in charge of evaluating an individual through a K-means algorithm, which has been parallelised with OpenMP on CPU and OpenCL on GPU. Using OpenMP for CPU is motivated by the ease of implementation, since with a single directive, it is possible to distribute the loop that iterates over the list of individuals.
4. **Fourth level**: GPU data parallelism in K-means (Fig. 2). The Euclidean distances between each point and the centroids are parallelised by the work-items (threads) of the same CU.

The application works as follows: a master MPI process is responsible for asynchronously distributing subpopulations among nodes and migrating individuals between subpopulations after a certain number of generations. In addition to this, the workers perform all the evolutionary steps of each subpopulation. After starting the MPI communications with the master, a worker requests the master as many subpopulations

Table 3
Mean and Relative Standard Deviation (RSD) of the 22 fitted parameters after repeating the fitting 20 times. Shaded cells denote highly consistent parameters. Units: **W** (clock cycles); **Pow** (watts); **T** (seconds).

Param	Mean ± RSD (%)	Param	Mean ± RSD (%)	Param	Mean ± RSD (%)	Param	Mean ± RSD (%)
$W_{1,1}$	$7.41 \cdot 10^6 \pm 2.57$	$W_{2,1}$	$1.03 \cdot 10^7 \pm 33.65$	$W_{3,1}$	$2.25 \cdot 10^7 \pm 36.57$	T_{com}	0.28 ± 148.90
$W_{1,2}$	$8.22 \cdot 10^7 \pm 2.37$	$W_{2,2}$	$1.09 \cdot 10^8 \pm 3.40$	$W_{3,2}$	$1.21 \cdot 10^8 \pm 28.84$	T_{master}	4.05 ± 39.08
$Pow_{1,1}$	222.85 ± 2.95	$Pow_{2,1}$	152.43 ± 56.72	$Pow_{3,1}$	165.10 ± 47.31	Pow_{master}	82.18 ± 3.76
$Pow_{1,2}$	79.69 ± 1.38	$Pow_{2,2}$	84.68 ± 1.65	$Pow_{3,2}$	76.97 ± 18.82	Pow_{switch}	4.18 ± 36.75
$Pow^{idle}_{1,1}$	218.61 ± 6.34	$Pow^{idle}_{2,1}$	118.92 ± 79.87	$Pow^{idle}_{3,1}$	85.75 ± 105.69		
$Pow^{idle}_{1,2}$	75.92 ± 9.93	$Pow^{idle}_{2,2}$	83.04 ± 8.37	$Pow^{idle}_{3,2}$	15.29 ± 128.31		

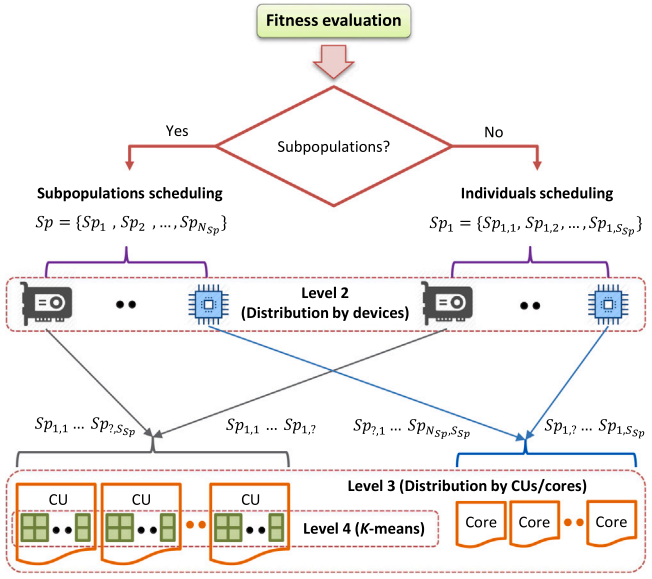


Fig. 2. The use case application. Evaluation of the fitness in the devices, which provides the third and fourth parallelism levels.

as devices are present in its node. The purpose of reducing idle states is to make all nodes busy as soon as possible. However, the worker node could receive a chunk of subpopulations less than or equal to the one indicated in the request if there are not enough subpopulations to be distributed. If this is not the case, once all workers are computing, the master waits for new requests. As each worker has already assigned work, a new request involves receiving only one subpopulation and sending another if available. The loop ends when all subpopulations have evolved. Then, the master merges all subpopulations to perform the next subpopulation set.

6.4. Model fitting results

The best fitting is the vector $(NRMSE_T, NRMSE_E)$ that provides the smallest Euclidean distance concerning the origin point, $(0, 0)$. Here, the NRMSE (Normalised Root-Mean-Square Error) for time and energy is calculated as the RMSE divided by the standard deviation of the measured values when executing the application varying N_{Sp} from 1 to 32 subpopulations:

$$NRMSE_T = \frac{RMSE_T}{std\left(T_{cluster}\ ;\ \forall N_{Sp} = 1, \ldots, 32\right)} \tag{15}$$

$$NRMSE_E = \frac{RMSE_E}{std\left(E_{cluster}\ ;\ \forall N_{Sp} = 1, \ldots, 32\right)} \tag{16}$$

The 20 repetitions of the fitting result in a set of 20 individuals that report low fitness, with mean and deviation values of $NRMSE_T = 0.089 \pm 0.012$ and $NRMSE_E = 0.139 \pm 0.031$. Hence, our method is capable of overall returning fairly well fitted solutions. Moreover, when picking the most accurate solution from the 20 fitted ones, the best individual achieves fitness values of $NRMSE_T = 0.081$ and $NRMSE_E = 0.091$. This means that the fitting exceeds greatly that obtained in the baseline study [33], whose best prediction results were $NRMSE_T = 0.213$ and $NRMSE_E = 0.256$. Both fittings are visually compared in Fig. 3. As it can be seen, the values predicted by the new model closely follow the experimentally measured data for any number of subpopulations. Although the baseline model fitting has a similar trend, it does not correctly predict runtime and energy consumption for subpopulations 18 to 24. From that point on, it also shows an irregular prediction trend. Therefore, using a genetic algorithm and a non-linear model is an accurate approach to capture the heterogeneous behaviour of the cluster. Indeed, we have obtained similar results using the Particle Swarm Optimisation (PSO) algorithm ($NRMSE_T = 0.104$ and $NRMSE_E = 0.113$). However, in order not to extend this paper too much, only the analyses related to the GA will be shown from now on. In summary, our methodology demonstrates its suitability in the path towards more complex models where features such as dynamic workload distribution or programs' hyperparameters play a major role.

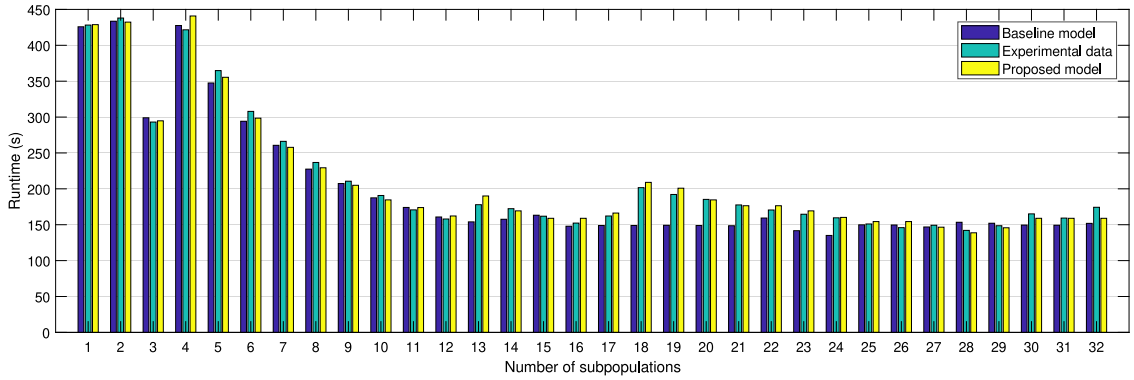
6.5. Model validation results

Since the fitting method explores the parameter space scattered, all fittings have been repeated 20 times. The Relative Standard Deviation (RSD) for the pth parameter of a chromosome, RSD_p, has been used to measure the dispersion:

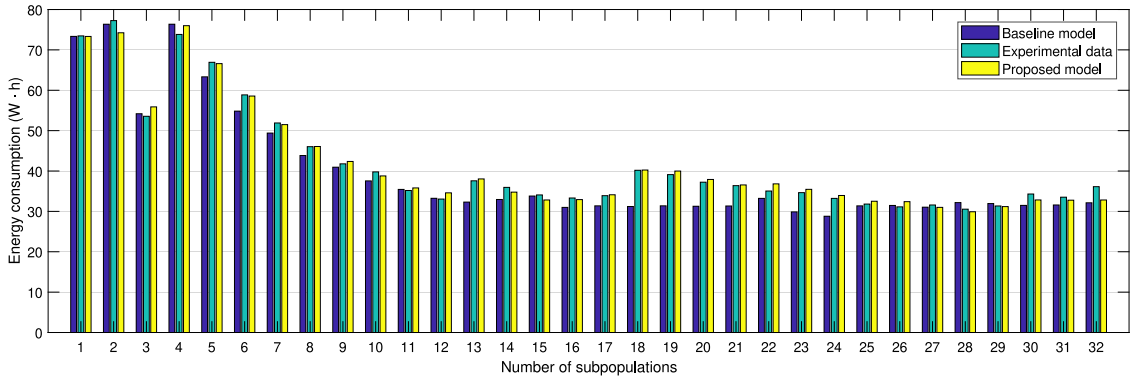
$$RSD_p = \frac{std\left(p_r\ ;\ \forall r = 1, \ldots, 20\right)}{mean\left(p_r\ ;\ \forall r = 1, \ldots, 20\right)} \cdot 100 \tag{17}$$

where p_r represents the rth fitting (repetition) of the pth parameter. The mean values and RSD of the 22 fitted parameters are displayed in Table 3. As it can be seen, most of the deviations are acceptable if the dynamic nature of the cluster is taken into account, except for $Pow^{idle}_{3,1}$, $Pow^{idle}_{3,2}$, and T_{com}. Nevertheless, these three parameters do not seem to have an appreciable impact on $NRMSE_T$ and $NRMSE_E$. It must be considered that the time needed for communication is quite low. Regarding $Pow^{idle}_{3,1}$ and $Pow^{idle}_{3,2}$, they present large deviations. This means that the values of the parameters are spread out in the solution search space, and therefore they may have different optimal values: as the use case application works through a dynamic master–worker scheme, the devices will always be busy and will only be idle during a migration, which is done in a short time. Thus, for this particular application, it could be stated that the instantaneous powers of the devices in an idle state are not essential for the model, although they must be included to obtain the best results. Besides, there is an explanation for the fact that some active instantaneous powers also have considerable deviations: the 20 experimental values collected from the cluster, used to fit the model, present disparate mean values. This could be due to a combination of the following factors:

1. Between iterations, the workload assigned to a device is variable, impacting the time the device is active or idle. The crossover probability, for example, affects the number of individuals to be evaluated in each generation.



(a) Runtime fitting



(b) Energy consumption fitting

Fig. 3. Comparison between the energy–time fitting of the baseline model and the one proposed in this work when increasing the number of subpopulations. For most cases, the proposed model fits the experimental data better than the baseline model [33].

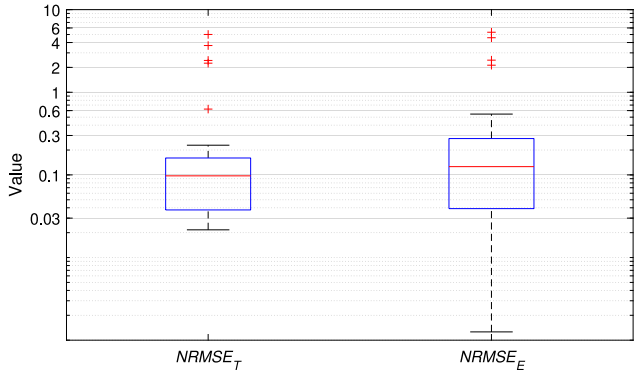
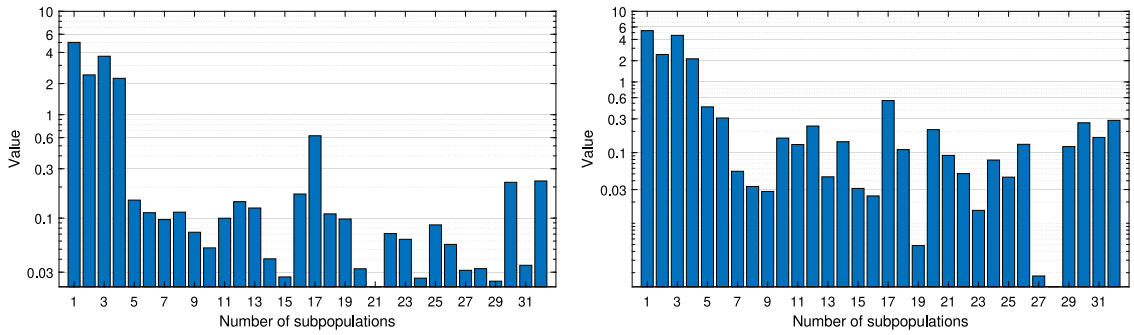


Fig. 4. Boxplot of $NRMSE_T$ and $NRMSE_E$ for the 32 iterations of LOO-CV.

2. It has been observed that the percentage of use of the CPU–GPU devices when executing the case use application is not always 100%, probably due to the sequential parts of the algorithm. Since the percentage is variable, so it will be the instantaneous power.
3. In the GA used to fit the parameters, there is no restriction to establish a minimum instantaneous power higher than zero. Consequently, the model offers very low power values as solutions that will never correspond to the real ones. Values close to the TDP are unrealistic, too. There is no solution to this problem for now since CPU and GPU manufacturers only report the maximum consumption (TDP) and rarely indicate the minimum.

Assuming that small deviation values are those less than 10%, the parameters whose fitted values converge in the same region are: (i) Pow_{master}, (ii) those related to the devices of Node 1, and (iii) those related to CPU of Node 2. The good fitting of these parameters could be due to the lack of good solutions in other regions of the search space. This hypothesis arose after verifying that in almost all the executions of the NSGA-II, the devices associated with these parameters were the fastest in requesting their initial workload. Therefore, they could be declared critical for the algorithm since they condition the subsequent workload distributions. As the rest of the parameters show worse fittings, they are considered less critical, which would imply that some components' roles in our context may overlap and vice versa. Considering a heterogeneous cluster comprising various configurations of nodes and devices, measuring the weight of the costs associated with the different components is not a trivial task. Moreover, since the workload distribution here is dynamic and heterogeneous devices compete to receive subpopulations, it is very difficult to determine which devices are active at any given moment simultaneously. This problem may be aggravated by increasing the number of nodes and devices in the cluster.

Another issue to discuss is the variance of the model. That is, the sensitivity of the fitting when the input variable N_{Sp} changes. For this purpose, a Leave-One-Out Cross-Validation (LOO-CV) has been performed with 32 independent fittings, each with a different number of subpopulations. The results are displayed in Figs. 4 and 5. All normalised errors are around 0.1 except when N_{Sp} takes values between 1 and 4, indicating that the model cannot predict the behaviour of the cluster after being fitted with the rest of the experimental data. This can be explained by the way Algorithm 1 assigns the workload: the subpopulations are initially assigned by node and device order, since it is impossible to know exactly which node will finally receive its subpopulations. When the number of subpopulations is high, there is no problem since several devices overlap their runtime. Conversely, when the number of subpopulations is very low, some devices will be idle. If the prediction of the model has not coincided with the real execution, there will be a fairly high prediction error due to the heterogeneity of



(a) $NRMSE_T$ (b) $NRMSE_E$

Fig. 5. $NRMSE_T$ and $NRMSE_E$ when performing LOO-CV from 1 to 32 subpopulations. The bars for 21 and 28 subpopulations are not visible in the figure because their values are very low.

the devices. Therefore, since the devices play a crucial role, this would confirm that the prediction model should be based on the workload distribution by devices instead of by nodes.

7. Conclusions and future work

By addressing the inherent complexities of dynamic workload distribution, the proposed model diverges from traditional linear approaches, resulting in a non-linear framework that enables more accurate modelling of energy consumption and runtime. The high degree of accuracy shown in the experimental results suggests that the model could be adapted to other bioinspired algorithms, given the structural similarities in how these algorithms operate across distributed systems. However, not all parameters were robustly fitted, as some exhibited a degree of flexibility, allowing for multiple valid solutions. This variability highlighted which parameters play a crucial role in determining the energy–time outcomes, as the final predictions remained largely unaffected by scattered parameter values or outliers. This insight is valuable for refining the model, as it enables a more targeted approach to parameter fitting by focusing on those with the most significant impact. Another possible improvement would be to focus on individual devices first before integrating the system as a whole. This approach could enhance both the precision and interpretability of the model, allowing for a clearer understanding of how each device contributes to the overall system's performance.

In addition, future extensions of the model could incorporate hardware optimisations such as DVFS and Dynamic Concurrency Throttling (DCT). These techniques, which adjust the device frequency, voltage levels, and the number of active cores, represent potential new parameters that could be optimised to minimise energy–time costs. Addressing different hardware configurations or profiles for each device could also resolve issues of solution sparsity, as discussed earlier in Section 6.5. However, expanding the model to account for additional hardware resources presents its own challenges. The increased dimensionality of the solution space (chromosome) could complicate the parameter-fitting process, potentially reducing the effectiveness of the model as the number of devices grows. In our case, each new device entails the fitting of one parameter related to time and two related to energy. As a result, while hardware optimisations hold promise for improving energy efficiency, careful consideration must be given to the trade-offs between model complexity and practical applicability, especially when scaling to larger systems involving dozens or even hundreds of devices. In this sense, a new study is required to know the threshold from which the model could be inconsistent.

CRediT authorship contribution statement

Juan José Escobar: Writing – original draft, Investigation, Funding acquisition, Formal analysis, Conceptualization. **Pablo Sánchez-Cuevas:** Writing – review & editing, Validation, Software, Data curation. **Beatriz Prieto:** Writing – review & editing, Resources, Project administration, Methodology. **Rukiye Savran Kızıltepe:** Writing – review & editing, Visualization, Validation, Methodology. **Fernando Díaz-del-Río:** Writing – review & editing, Supervision, Investigation. **Dragi Kimovski:** Writing – review & editing, Visualization, Validation, Supervision.

Funding

This research is part of the:

- PID2022-137461NB-C32 and PID2023-151065OB-I00 projects, funded by the MICIU/AEI/10.13039/501100011033 and by ESF+ ("NextGenerationEU/PRTR").
- PPJIA2023-025 project, funded by the University of Granada.
- Program of mobility stays for professors and researchers in foreign higher education and research centres, funded by the Spanish Ministry of Universities under grant CAS22/00332.
- P.S.-C. was supported by "Predoctores 2021" (PREDOC_01229) fellowship from the Ministry of Economic Transformation, Industry, Knowledge and Universities of the Regional Government of Andalusia.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgements

The authors thank to Alberto Prieto, from the Department of Computer Engineering, Automation, and Robotics of the University of Granada, for his invaluable support in this work.

Appendix A. Glossary of formal variables

1. α: crossover probability of the NSGA-II.
2. β: mutation probability of the NSGA-II.
3. $C_{i,j}$: number of cores of the jth device of the ith node that are working in parallel.
4. c_E: chromosome representation of energy parameters.
5. c_{TW}: chromosome representation of time/workload parameters.
6. $E_{cluster}$: measured energy consumption of the cluster to perform the whole genetic algorithm.
7. $\hat{E}_{cluster}$: predicted energy consumption of the cluster to perform the whole genetic algorithm.
8. $\hat{E}_{i,j}$: predicted energy consumption of the jth device of the ith node to compute $N_{Sp_{i,j}}$ subpopulations between two migrations.

9. $F_{i,j}$: clock frequency of the jth device of the ith node.
10. g: number of generations of the NSGA-II.
11. N: number of individuals of the NSGA-II.
12. N_D: list with the number of devices of each worker node.
13. N_{D_i}: number of devices of the ith node.
14. N_{Gm}: number of global migrations of the NSGA-II.
15. N_{Wk}: number of worker nodes in the cluster.
16. N_{Sp}: number of subpopulations of the NSGA-II.
17. $N_{Sp_{i,j}}$: list with the predicted number of subpopulations of the NSGA-II assigned to the jth device of the ith node.
18. $NRMSE_E$: normalised $RMSE_E$.
19. $NRMSE_T$: normalised $RMSE_T$.
20. p_r: rth fitting (repetition) of the pth parameter of a chromosome.
21. P_E: number of parallel evaluations of individuals per subpopulation.
22. $Pow_{i,j}$: instantaneous power of the jth device of the ith node.
23. $Pow_{i,j}^{idle}$: instantaneous power in idle of the jth device of the ith node.
24. Pow_{master}: instantaneous power of the master node.
25. Pow_{switch}: instantaneous power of the switch.
26. $T_{cluster}$: measured runtime of the cluster to perform the whole genetic algorithm.
27. $\hat{T}_{cluster}$: predicted runtime of the cluster to perform the whole genetic algorithm.
28. $\hat{T}_{com}$: predicted runtime for communications between nodes of the cluster when performing a global migration.
29. $\hat{T}_{evo}$: predicted runtime of the cluster for computing the evolution of all N_{Sp} subpopulations between two migrations.
30. $\hat{T}_{i,j}$: predicted runtime of the jth device of the ith node to compute $N_{Sp_{i,j}}$ subpopulations between two migrations.
31. $\hat{T}_{i,j}^{1}$: predicted runtime of the jth device of the ith node to compute one subpopulation between two migrations.
32. $\hat{T}_{master}$: predicted runtime overhead of the master node when performing a global migration.
33. $TDP_{i,j}$: TDP of the jth device of the ith node.
34. TDP_{master}: TDP of the master node.
35. TDP_{switch}: TDP of the switch.
36. R_1, R_2, R_3: restrictions of the GA used to fit the model.
37. $RMSE_E$: RMSE between predicted and measured cluster energy consumption.
38. $RMSE_T$: RMSE between predicted and measured cluster runtime.
39. RSD_p: RSD of the pth parameter of a chromosome.
40. S_{Sp}: size of a subpopulation.
41. Sp: list of subpopulations.
42. Sp_i: ith subpopulation.
43. $Sp_{i,j}$: jth individual of the ith subpopulation.
44. $W_{i,j}$: number of clock cycles taken by the jth device of the ith node to compute one individual.

Appendix B. Glossary of acronyms

1. AI: Artificial Intelligence.
2. AVX: Advanced Vector Extensions.
3. CPU: Central Processing Unit.
4. CU: Compute Unit.
5. DAG: Directed Acyclic Graph.
6. DCT: Dynamic Concurrency Throttling.
7. DEAP: Distributed Evolutionary Algorithms in Python.
8. DFS: Dynamic Frequency Scaling.
9. DLP: Data Level Parallelism.
10. DVFS: Dynamic Voltage and Frequency Scaling.
11. DVS: Dynamic Voltage Scaling.
12. EEG: Electroencephalogram.
13. GA: Genetic Algorithm.
14. GCC: GNU Compiler Collection.
15. GPU: Graphics Processing Unit.
16. HPC: High-Performance Computing.
17. ILP: Instruction Level Parallelism.
18. LOO-CV: Leave-One-Out Cross-Validation.
19. MPI: Message Passing Interface.
20. NUMA: Non-Uniform Memory Access.
21. NRMSE: Normalised Root-Mean-Square Error.
22. NSGA-II: Non-dominated Sorting Genetic Algorithm.
23. PSO: Particle Swarm Optimisation.
24. RMSE: Root-Mean-Square Error.
25. RSD: Relative Standard Deviation.
26. SBB: Simulated Binary Bounded.
27. TDP: Thermal Design Power.
28. TLP: Thread Level Parallelism.
29. TPE: Tree-structured Parzen Estimator.

Data availability

Data will be made available on request.

References

[1] J.L. Hennessy, D.A. Patterson, Computer Architecture, Sixth Edition: A Quantitative Approach, Morgan Kaufmann Publishers Inc., 2017.

[2] A. Flores, J.L. Aragon, M.E. Acacio, Sim-PowerCMP: A detailed simulator for energy consumption analysis in future embedded CMP architectures, in: 21st International Conference on Advanced Information Networking and Applications Workshops, AINAW '07, IEEE, Niagara Falls, ON, Canada, 2007, pp. 752–757, http://dx.doi.org/10.1109/AINAW.2007.334.

[3] K. O'brien, I. Pietri, R. Reddy, A. Lastovetsky, R. Sakellariou, A survey of power and energy predictive models in HPC systems and applications, ACM Comput. Surv. 50 (3) (2017) 1–38, http://dx.doi.org/10.1145/3078811.

[4] P. Czarnul, J. Proficz, A. Krzywaniak, Energy-aware high-performance computing: Survey of state-of-the-art tools, techniques, and environments, Sci. Program. 2019 (1) (2019) http://dx.doi.org/10.1155/2019/8348791.

[5] Q. Liu, W. Luk, Heterogeneous systems for energy efficient scientific computing, in: 8th International Symposium on Applied Reconfigurable Computing: Architectures, Tools and Applications, ARC '2012, Springer, Hongkong, China, 2012, pp. 64–75, http://dx.doi.org/10.1007/978-3-642-28365-9_6.

[6] H. Khaleghzadeh, M. Fahad, A. Shahid, R.R. Manumachu, A. Lastovetsky, Bi-objective optimization of data-parallel applications on heterogeneous HPC platforms for performance and energy through workload distribution, IEEE Trans. Parallel Distrib. Syst. 32 (3) (2021) 543–560, http://dx.doi.org/10.1109/TPDS.2020.3027338.

[7] J. Stone, M. Hallock, J. Phillips, J. Peterson, Z. Luthey-Schulten, K. Schulten, Evaluation of emerging energy-efficient heterogeneous computing platforms for biomolecular and cellular simulation workloads, in: International Parallel and Distributed Processing Symposium Workshops, IPDPSW '2016, IEEE, Chicago, IL, USA, 2016, pp. 89–100, http://dx.doi.org/10.1109/IPDPSW.2016.130.

[8] A. Guillén, D. Sovilj, A. Lendasse, F. Mateo, I. Rojas, Minimising the delta test for variable selection in regression problems, Int. J. High Perform. Syst. Archit. 1 (4) (2008) 269–281, http://dx.doi.org/10.1504/IJHPSA.2008.024211.

[9] P. Haghi, A. Guo, T. Geng, A. Skjellum, M.C. Herbordt, Workload imbalance in HPC applications: Effect on performance of in-network processing, in: IEEE High Performance Extreme Computing Conference, HPEC '2021, IEEE, Waltham, MA, USA, 2021, pp. 1–8, http://dx.doi.org/10.1109/HPEC49654.2021.9622847.

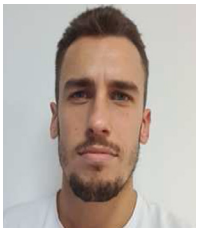
[10] A. Sohail, Genetic algorithms in the fields of artificial intelligence and data sciences, Ann. Data Sci. 10 (4) (2023) 1007–1018, http://dx.doi.org/10.1007/s40745-021-00354-9.

[11] B.o. Kocot, P. Czarnul, J. Proficz, Energy-aware scheduling for high-performance computing systems: A survey, Energies 16 (2) (2023) e890, http://dx.doi.org/10.3390/en16020890.

[12] S. Kumar, S. Pal, S. Singh, V.P. Singh, D. Singh, T.K. Saha, H. Gupta, P. Jaiswal, Energy efficient model for balancing energy in cloud datacenters using dynamic voltage frequency scaling (DVFS) technique, in: 3rd Doctoral Symposium on Computational Intelligence, DoSCI '2022, Springer, Lucknow, India, 2022, pp. 533–540, http://dx.doi.org/10.1007/978-981-19-3148-2_45.

[13] S. Baskiyar, R. Abdel-Kader, Energy aware DAG scheduling on heterogeneous systems, Clust. Comput. 13 (4) (2010) 373–383, http://dx.doi.org/10.1007/s10586-009-0119-6.
[14] E. Rotem, U.C. Weiser, A. Mendelson, R. Ginosar, E. Weissmann, Y. Aizik, H-EARtH: Heterogeneous multicore platform energy management, IEEE Comput. Mag. 49 (10) (2016) 47–55, http://dx.doi.org/10.1109/MC.2016.309.
[15] Z. Wang, H. Wang, W. Zhao, L. Cheng, Energy optimization of parallel programs in a heterogeneous system by combining processor core-shutdown and dynamic voltage scaling, Future Gener. Comput. Syst. 92 (2019) 198–209, http://dx.doi.org/10.1016/j.future.2018.09.039.
[16] G.L. Stavrinides, H.D. Karatza, An energy-efficient, QoS-aware and cost-effective scheduling approach for real-time workflow applications in cloud computing systems utilizing DVFS and approximate computations, Future Gener. Comput. Syst. 96 (2019) 216–226, http://dx.doi.org/10.1016/j.future.2019.02.019.
[17] H. Li, Y. Wei, Y. Xiong, E. Ma, W. Tian, A frequency-aware and energy-saving strategy based on DVFS for spark, J. Supercomput. 77 (10) (2021) 11575–11596, http://dx.doi.org/10.1007/s11227-021-03740-5.
[18] M.A.N. Al-hayanni, F. Xia, A. Rafiev, A. Romanovsky, R. Shafik, A. Yakovlev, Amdahl's law in the context of heterogeneous many-core systems - A survey, IET Comput. Digit. Tech. 14 (4) (2020) 133–148, http://dx.doi.org/10.1049/iet-cdt.2018.5220.
[19] Y. Cao, F. Wu, T. Robertazzi, Integrating Amdahl-like laws and divisible load theory, Parallel Process. Lett. 31 (2) (2021) e2150008, http://dx.doi.org/10.1142/S0129626421500080.
[20] F. Díaz-del-Río, J. Salmerón-García, J.L. Sevillano, Extending Amdahl's law for the cloud computing era, Computer 49 (2) (2016) 14–22, http://dx.doi.org/10.1109/MC.2016.49.
[21] A. Shahid, M. Fahad, R.R. Manumachu, A. Lastovetsky, Improving the accuracy of energy predictive models for multicore CPUs by combining utilization and performance events model variables, J. Parallel Distrib. Comput. 151 (2021) 38–51, http://dx.doi.org/10.1016/j.jpdc.2021.01.007.
[22] D. De Sensi, Predicting performance and power consumption of parallel applications, in: 24th Euromicro International Conference on Parallel, Distributed, and Network-Based Processing, PDP '2016, IEEE, Heraklion Crete, Greece, 2016, pp. 200–207, http://dx.doi.org/10.1109/PDP.2016.41.
[23] A. Guillén, M.I. García Arenas, M. van Heeswijk, D. Sovilj, A. Lendasse, L.J. Herrera, H. Pomares, I. Rojas, Fast feature selection in a GPU cluster using the delta test, Entropy 16 (2) (2014) 854–869, http://dx.doi.org/10.3390/e16020854.
[24] E. García-Martín, C.F. Rodrigues, G. Riley, H. Grahn, Estimation of energy cnsumption in machine learning, J. Parallel Distrib. Comput. 134 (2019) 75–88, http://dx.doi.org/10.1016/j.jpdc.2019.07.007.
[25] M. Fahad, A. Shahid, R.R. Manumachu, A. Lastovetsky, A comparative study of methods for measurement of energy of computing, Energies 12 (11) (2019) e2204, http://dx.doi.org/10.3390/en12112204.
[26] J. Marszał kowski, M. Drozdowski, J. Marszał kowski, Time and energy performance of parallel systems with hierarchical memory, J. Grid Comput. 14 (1) (2016) 153–170, http://dx.doi.org/10.1007/s10723-015-9345-8.
[27] X. Wu, V. Taylor, J. Cook, P. Mucci, Using performance-power modeling to improve energy efficiency of HPC applications, Computer 49 (10) (2016) 20–29, http://dx.doi.org/10.1109/MC.2016.311.
[28] A. Hussain, M. Aleem, M.A. Islam, M.A. Iqbal, A rigorous evaluation of state-of-the-art scheduling algorithms for cloud computing, IEEE Access 6 (2018) 75033–75047, http://dx.doi.org/10.1109/ACCESS.2018.2884480.
[29] K. Geeta, V.K. Prasad, Multi-objective cloud load-balancing with hybrid optimization, Int. J. Comput. Appl. 45 (10) (2023) 611–625, http://dx.doi.org/10.1080/1206212X.2023.2260616.
[30] J. Wang, X. Li, R. Ruiz, J. Yang, D. Chu, Energy utilization task scheduling for MapReduce in heterogeneous clusters, IEEE Trans. Serv. Comput. 15 (2) (2022) 931–944, http://dx.doi.org/10.1109/TSC.2020.2966697.
[31] J. Liu, P. Yang, C. Chen, Intelligent energy-efficient scheduling with ant colony techniques for heterogeneous edge computing, J. Parallel Distrib. Comput. 172 (2023) 84–96, http://dx.doi.org/10.1016/j.jpdc.2022.10.003.
[32] J.J. Escobar, J. Ortega, A.F. Díaz, J. González, M. Damas, Energy-aware load balancing of parallel evolutionary algorithms with heavy fitness functions in heterogeneous CPU-GPU architectures, Concurr. Comput.: Pr. Exp. 31 (6) (2019) e4688, http://dx.doi.org/10.1002/cpe.4688.
[33] J.J. Escobar, J. Ortega, A.F. Díaz, J. González, M. Damas, Time-energy analysis of multi-level parallelism in heterogeneous clusters: The case of EEG classification in BCI tasks, J. Supercomput. 75 (7) (2019) 3397–3425, http://dx.doi.org/10.1007/s11227-019-02908-4.
[34] B. Alhijawi, A. Awajan, Genetic algorithms: Theory, genetic operators, solutions, and applications, Evol. Intell. 17 (3) (2024) 1245–1256, http://dx.doi.org/10.1007/s12065-023-00822-6.
[35] D.E. Goldberg, Genetic Algorithms in Search, Optimization and Machine Learning, Addison-Wesley Longman Publishing, 1989.
[36] S. Katoch, S.S. Chauhan, V. Kumar, A review on genetic algorithm: Past, present, and future, Multimedia Tools Appl. 80 (5) (2021) 8091–8126, http://dx.doi.org/10.1007/s11042-020-10139-6.
[37] Intel, Thermal design power (TDP) in intel processors, 2023, URL https://www.intel.com/content/www/us/en/support/articles/000055611/processors.html?wapkw=tdp (Accessed 02 October 2024).
[38] J.J. Escobar, Hpmoon. a parallel and distributed multi-objective genetic algorithm to EEG classification, 2024, URL https://github.com/rotty11/Hpmoon (Accessed 02 October 2024).
[39] F.A. Fortin, F.M. De Rainville, M.A. Gardner, M. Parizeau, C. Gagné, DEAP: Evolutionary algorithms made easy, J. Mach. Learn. Res. 13 (70) (2012) 2171–2175.
[40] H. Ma, Y. Zhang, S. Sun, T. Liu, Y. Shan, A comprehensive survey on NSGA-II for multi-objective optimization and applications, Artif. Intell. Rev. 56 (12) (2023) 15217–15270, http://dx.doi.org/10.1007/s10462-023-10526-z.

Juan José Escobar received the Ph.D. degree in Computer Science from University of Granada, Spain, in 2020. He is a Permanent Lecturer at the Department of Software Engineering of University of Granada. His research interests include code optimisation, energy-efficient parallel computing, and workload balancing strategies on heterogeneous and distributed systems, specially in issues related to evolutionary algorithms and multi-objective feature selection problems.

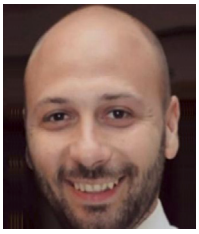


Pablo Sánchez-Cuevas received the B.Sc. degree in Computer Engineering from the University of Seville, Spain, and the M.Sc. degree in Data Science and Computer Engineering in the University of Granada, Spain, in 2020 and 2021, respectively. He is currently a Ph.D. student at the Department of Computer Architecture and Technology of the University of Seville, supported by a research fellowship granted in 2022. His current research interests include computer architecture, parallel computing, field-programmable gate array design, machine learning, deep learning, and computer vision.

Beatriz Prieto has a B.Sc. degree in Electronic Engineering and a Ph.D. degree in Electronic Engineering from University of Granada. She is an Associate professor at the Department of Computer Engineering, Automation, and Robotics of the University of Granada. Her research interests are focused in the field of intelligent systems for signal processing applied to biomedical applications, Green Computing, and parallel and distributed computing.

Rukiye Savran Kiziltepe received the Ph.D. degree in the School of Computer Science and Electronic Engineering at the University of Essex, Colchester, in 2022. She is currently an Assistant Professor at the Department of Software Engineering of Karadeniz Technical University and Ankara University. Her research interests include machine learning, video processing, and computer vision. She is particularly interested in video understanding using deep learning techniques.

Fernando Díaz-del-Río received his Master in Physics (Electronics) and Ph.D. from the University of Seville, Spain, in 1990 and 1997, respectively. He joined the Department of Computer Architecture, University of Seville, where he has been an Associate Professor since 2000. He served as Vice-Dean of the Computer Engineering School (2007–2010). He is author/coauthor of more than 50 papers in refereed international journals and conferences. Mobile robot, parallel computing and image processing are his main research topics. He has participated in or directed more than 25 research projects and contracts.



Dragi Kimovski received the Ph.D. degree from the Technical University of Sofia, Bulgaria, in 2013, and the Habilitation degree from the University of Klagenfurt, Austria, in 2023. Between 2015–2018, he worked as a postdoctoral researcher with the University of Innsbruck, Austria. He is a university docent with ITEC, University of Klagenfurt, Austria. He has co-authored more than 80 international articles in parallel and distributed systems. He participated as a workpackage and scientific coordinator in several European projects.