JOURNAL OF COMPUTATIONAL BIOLOGY
Volume 25, Number 8, 2018

Pp. 882–893
DOI: 10.1089/cmb.2018.0080

A Power–Performance Perspective to Multiobjective Electroencephalogram Feature Selection on Heterogeneous Parallel Platforms

JUAN JOSÉ ESCOBAR, JULIO ORTEGA, ANTONIO FRANCISCO DÍAZ, JESÚS GONZÁLEZ, and MIGUEL DAMAS

ABSTRACT

This article provides an insight on the power–performance issues related with the CPU-GPU (Central Processing Unit-Graphics Processing Unit) parallel implementations of problems that frequently appear in the context of applications on bioinformatics and biomedical engineering. More specifically, we analyze the power–performance behavior of an evolutionary parallel multiobjective electroencephalogram feature selection procedure that evolves subpopulations of solutions with time-demanding fitness evaluation. The procedure has been implemented in OpenMP to dynamically distribute either subpopulations or individuals among devices, and uses OpenCL to evaluate the fitness of the individuals. The development of parallel codes usually implies to maximize the code efficiency, thus optimizing the achieved speedups. To follow the same trend, this article extends and provides a more complete analysis of our previous works about the power–performance characteristics in heterogeneous CPU-GPU platforms considering different operation frequencies and evolutionary parameters, such as distribution of individuals, etc. This way, different experimental configurations of the proposed procedure have been evaluated and compared with respect to a master–worker approach, not only in runtime but also considering energy consumption. The experimental results show that lower operating frequencies does not necessarily mean lower energy consumptions since energy is the product of power and time. Thus, we have observed that parallel processing not only reduces the runtime, but also the energy consumed by the application despite a higher instantaneous power. Particularly, the workload distribution among both CPU and GPU cores provides the best runtime and very low energy consumption compared with the values achieved by the same alternatives executed by only CPU threads.

Keywords: dynamic scheduling, EEG classification, energy-aware computing, heterogeneous parallelism, multiobjective feature selection, subpopulations.

1. INTRODUCTION

EEG CLASSIFICATION PROBLEMS appear in many neurological and bioengineering applications, such as diagnosis of sleep disorders, prediction of epileptic seizures, or brain–computer interface (BCI) tasks (Rupp et al., 2014). Electroencephalogram (EEG) classification involves some hard issues related with the

Department of Computer Architecture and Technology, CITIC, University of Granada, Granada, Spain.

unstable behavior and nonlinearity of the signals, the low signal-to-noise rate, and the high number of features required to represent the information included in the evolution of the signals with respect to time. Moreover, as the experimental work required to register EEG signals (or EEG patterns) for different subjects and situations is quite time consuming, the number of EEG patterns available to train the classifiers is much smaller than the number of features used to describe each EEG. Thus, a so-called curse-of-dimensionality problem (Duin, 2000) arises, unless a feature selection procedure to reduce the dimensionality of the EEG patterns is accomplished.

Nevertheless, finding the optimum set of features has proven to be a Nondeterministic Polynomial-time hard (NP-hard) problem, and thus metaheuristics, such as simulated annealing, genetic algorithms, ant colony optimization, and particle swarm optimization constitute suitable approaches to tackle the problem (Marinaki and Marinakis, 2014). Thus, feature selection can take advantage of evolutionary algorithms. Although these algorithms could require a lot of runtime in high-dimensional problems, as they do not use explicit information about the problem to be solved, they can be accelerated by present parallel computer architectures in several ways. In our previous articles (Escobar et al., 2016a,b, 2017a,c), we described the benefits of CPUs and GPU cores to accelerate EEG classification which, as many other bioinformatics applications, require solving problems with different parallelism types. More specifically, in Escobar et al. (2017a), we accelerated the EEG feature selection problem by a subpopulation-based evolutionary algorithm to take advantage of parallel architectures involving multicore Central Processing Units (CPUs) and Graphics Processing Units (GPUs). Nevertheless, besides speed, energy consumption has become an important issue to evaluate the program efficiency, not only due to economic and environmental reasons, but also as a challenge for the high-performance computing community to allow efficient use of future Exascale systems, and as a requirement for handheld and wearable devices.

Although the relevance of energy consumption in the context of evolutionary algorithms has been pointed out in Cotta et al. (2015), and even taking into account that decreasing energy consumption should be considered at par with decreasing the running time, to the best of our knowledge, there is not any article on parallel evolutionary algorithms that provides a detailed efficiency analysis from the power–performance approach. Fernández-de-Vega et al. (2016) analyzes the energy consumption in different platforms of a sequential evolutionary procedure, but deals more with the energy efficiency of different platforms than with the comparison of the energy consumption of different algorithms in the same heterogeneous platform.

In this article, we provide a detailed analysis of different parallel implementations of our subpopulation-based evolutionary algorithm for EEG feature selection, not only with respect to the quality of the obtained solutions and the speedup achieved by the alternative configurations of parallel platforms, but also considering their energy consumption characteristics. This way, with respect to our previous article (Escobar et al., 2017a), we analyze the quality of the solutions achieved (hypervolume indicator), the speedup, and the energy consumed by our parallel codes in a more complete set of experiments. These experiments correspond to different operating frequencies (1.2 and 2.1 GHz, and the alternative used by the runtime system), subpopulations, and individuals per subpopulations, number of migrations, and platforms (heterogeneous [HET] platforms, including both CPU and GPU cores, or constituted by either CPU or GPU cores). We have also shown the evolution with time of the instantaneous power dissipated by different alternatives on operating frequency and subpopulations.

After this introduction, Section 2 briefly describes the parallel evolutionary multiobjective algorithm alternatives for feature selection in heterogeneous CPU-GPU architectures along with some details for their implementations. Then, Section 3 shows the related works in the literature, Section 4 describes and analyzes the experimental setup and results, and finally, Section 5 summarizes the conclusions.

2. A SUBPOPULATION-BASED MULTIOBJECTIVE FEATURE SELECTION ON CPU-GPU PLATFORMS

A multiobjective evolutionary procedure, in our case the well-known NSGA-II algorithm (Deb et al., 2000), evolves subpopulations of individuals that codify different feature selections. Besides the mutation and crossover operators applied to some selected parent individuals, NSGA-II also includes the so-called Nondomination Sorting to rank the individuals into different levels of nondominance: the first level (Pareto front) includes the individuals nondominated by any other individual.

Given a feature selection (an individual in the subpopulation), the components of the N_P patterns included in the training dataset, *DS*, are determined and correspond to the selected features among the whole set of N_F features. In our approach, the fitness of each feature selection is obtained by applying the *K*-means algorithm to the N_P patterns $P_i = (p_i^1, \ldots, p_i^{N_F})(i = 1, \ldots, N_P)$ to determine the centroids $K^t(j)(j = 1, .., W)$ of the W possible clusters ($W = 3$ because it is equal to the number of classes in our BCI tasks). Once the clusters are built by including each pattern in its nearest centroid, the fitness of each individual in the subpopulation is evaluated by using two clustering validation indices, defined by the intraclass f_1 and the interclass f_2 distances, which are detailed in Escobar et al. (2017c).

The parallel code has been implemented with OpenMP pragmas and OpenCL. OpenMP dynamically distributes the fitness evaluation of the individuals (the cost functions f_1 and f_2) launching OpenCL kernels among both CPU and GPU devices. As many CPU threads as available OpenCL devices, N_D, are created through the corresponding OpenMP pragma to parallelize the loop that iterates over all subpopulations, S_p. To evaluate the fitness in parallel, two dynamic scheduling alternatives can be applied, as the procedure distributes subpopulations or individuals when only one subpopulation is detected. Thus, according to the device, two parallelism levels can be achieved in a CPU, and up to three in a GPU, where the *K*-means data parallelism is also implemented. Algorithm 1 provides a detailed description of our parallel multiobjective evolutionary algorithm, named D2S_NSGAII (Dynamic Distribution of Subpopulations using NSGA-II), which also is summarized in Figure 1.

On the other hand, to execute a GPU kernel, the individuals must be transferred from the host memory to the GPU memory, and vice versa. The required copies per generation between devices could constitute an important bottleneck, as was analyzed in Escobar et al. (2017b). Nevertheless, as the subpopulations are dynamically allocated to the devices available in the platform, the time required for transferring data can be overlapped.

A migration implies to build a new set of subpopulations. To define a new subpopulation, each subpopulation contributes with half of its solutions of its present Pareto front at most. Finally, the solutions obtained by different subpopulations are recombined by the main CPU thread, and returned at the end of the function. These steps are repeated according to the required number of subpopulation generations and migrations as Figure 1 shows.

3. RELATED WORKS

The use of multiobjective optimization in data mining applications has been shown in Mukhopadhyay et al. (2014a,b), and its benefits in both supervised and unsupervised classification have been reported elsewhere (Handl and Knowles, 2006). As GPU architectures constitute one of the present mainstream approaches to take advantage of technology improvements (Collet, 2013), their use has been described in

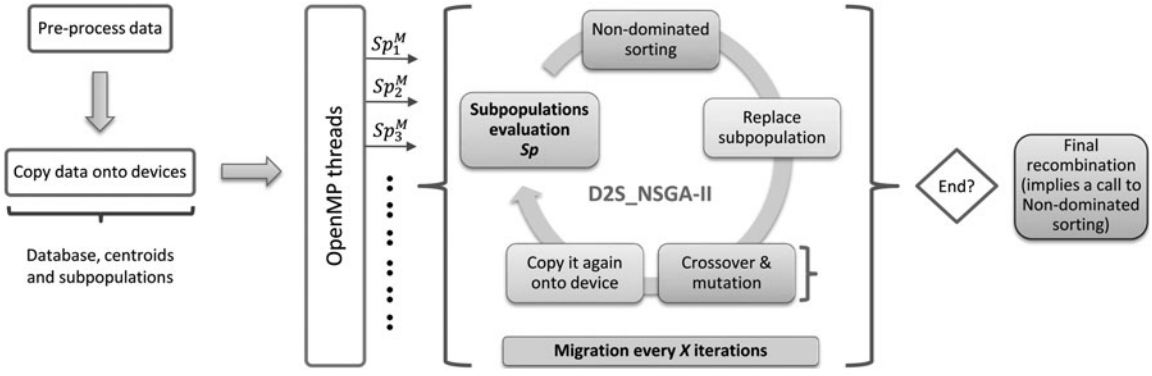

FIG. 1. Scheme of the steps in the D2S_NSGAII procedure. First, the algorithm receives all necessary parameters to perform the evolution of subpopulations. Then, each subpopulation Sp_i^M (assigned to one CPU thread) is evaluated by one OpenCL device, which executes the cycle of steps inside of the blue brackets. If only one subpopulation is detected, all devices cooperate to perform the evaluation. The algorithm uses uniform crossover with a probability of 0.75, mutation by inversion of the selected bit with a probability of 0.025, and selection by binary tournament.

many previous articles involving the analysis of the acceleration rates attained by the GPUs with respect to a sequential implementation that only uses CPU cores (Luong et al., 2010). With respect to evolutionary algorithm implementations, it is

Algorithm 1: Subpopulations scheduler pseudocode. The evaluation of subpopulations is distributed among all OpenCL devices, where each of them is assigned to one OpenMP thread

```
1  Function D2S_NSGAII(S_p, N_D, D, N_Spop, M, DS, K DS^t)
   |  Input : The initial subpopulations, Sp_i, ∀i = 1, ..., N_Spop
   |  Input : Number of available OpenCL devices, N_D
   |  Input : Objects D_j containing the OpenCL devices, ∀_j = 1, ..., N_D
   |  Input : Number of subpopulations N_Spop to be evolved
   |  Input : Number of individuals in each subpopulation, M
   |  Input : Dataset DS: N_P training patterns of N_F features
   |  Input : Set of K of W centroids randomly chosen from DS
   |  Input : Dataset DS^t is DS in column-major order
   |  Output : S, the new solution for the problem
2  |  repeat
   |  |  // OpenMP parallel section with N_D devices
3  |  |  repeat
   |  |  |  // Start the evolution process
4  |  |  |  repeat
5  |  |  |  |  Off spr ← UniformCrossover(Sp_i)
6  |  |  |  |  if D_j is a CPU then
7  |  |  |  |  |  Off spr ← evaluationsCPU(Off spr, M, DS, K)
8  |  |  |  |  else
9  |  |  |  |  |  Off spr ← evaluationGPU(Off spr, M, DS, K, DS^t)
10 |  |  |  |  end
   |  |  |  |  // Replacement process
11 |  |  |  |  Aux ← Join Sp_i and Off spr in one array
12 |  |  |  |  Aux ← nonDominatedSorting(Aux, M + N_Off spr)
13 |  |  |  |  Sp_i ← Copy the first M individuals from Aux
14 |  |  |  until the number of subpopulations generations is reached;
15 |  |  until all N_Spop subpopulations are evaluated;
16 |  |  Sp ← migration(Sp, N_Spop, M)
17 |  until the number of desired migrations is reached;
   |  // Recombination process
18 |  Sp ← nonDominatedSorting (Sp, N_Spop, ×M)
19 |  S ← Copy the first M individuals for Sp
20 |  return S
21 End
```

possible to use the GPU only to evaluate the fitness of the individuals in the population, taking advantage of the data parallelism present in that fitness function. Another approach is to implement the whole evolutionary algorithm in the GPU (Jähne, 2016).

An alternative GPU implementation of the nondominance rank used in NSGA-II, the Archived-based Stochastic Ranking Evolutionary Algorithm (ASREA), is provided in Sharma and Collet (2013). Article (Wong and Cui, 2013) provides a parallel NSGA-II implementation for a data mining application that executes all steps of the algorithm in the GPU except for the nondominated selection of a multi-objective evolutionary algorithm. Nevertheless, works analyzing the effect in the parallel performance of heavy fitness functions requiring high-volume datasets and the parallelization on a heterogeneous platform of a whole data mining application with similar characteristics to our target application are less frequent. In our previous article (Escobar et al., 2016a), we proposed a multiobjective feature selection that implements both functional and data parallelism, which can be executed either in a CPU or in a GPU. Moreover, in Escobar et al. (2016b, 2017c), the effect of memory access optimization on GPU implementations has been demonstrated.

The main approaches to the development of energy-efficient parallel and distributed codes can be grouped into two alternatives. Several approaches propose scheduling procedures that take into account not

only running time but also energy consumption of the program (Zhang et al., 2002; Baskiyar and Abdel-Kader, 2010; Lee and Zomaya, 2011; Nesmachnow et al., 2013; Dorronsoro et al., 2014; Rotem et al., 2016). Other approaches investigate the effect of different implementations for a specific application in energy consumption and try to derive energy-aware strategies and power models from the corresponding experimental results (Aliaga et al., 2014). In this study, we follow this approach.

With respect to energy consumption efficiency of hybrid CPU-GPU platforms, articles (Ma et al., 2012; Marowka, 2012; Allen and Ge, 2016) provide some results on this topic. For example, Marowka (2012) provides analytical models to get insight into performance gains and energy consumption and concludes that a greater parallelism allows opportunities for energy-saving parallel applications. We demonstrate this from the energy consumption we have measured in our computing node.

4. EXPERIMENTAL RESULTS AND DISCUSSION

In this section, we analyze the performance of our OpenMP-OpenCL codes running on Linux CentOS 6.7 operating system, in a node with two Intel Xeon E5-2620 v4 processors at 2.1 GHz, including eight cores per socket with two Hyper-Threading threads per core, thus comprising 32 threads. The node also has a GPU NVIDIA Tesla K40m with 288 GB/s as maximum memory bandwidth and 2880 CUDA cores at 745 MHz. In our experiments, we have used three datasets from the BCI Laboratory at the University of Essex and described in Asensio-Cubero et al. (2013). They correspond to subjects coded as 104, 107, and 110, and each includes 178 EEG patterns with 3600 features per pattern. The measures have been obtained considering three different alternatives: two correspond to the use of fixed operation frequencies at 1.2 and 2.1 GHz in CPU, and the third one is the so-called *Syst* alternative, in which the runtime system modifies the operation frequency according to its strategy to optimize the code efficiency.

4.1. Hypervolume results

The quality of the solution obtained by our procedure is evaluated through their corresponding Pareto front hypervolume (Fonseca et al., 2018), computed here with (1,1) as reference point, and the minimum values of the cost functions f_1 and f_2 are, respectively, 0 and −1. Thus, the maximum value for the hypervolume is 2. We have made 10 repetitions of each experiment to analyze, through Kolmogorov–Smirnov and Kruskal–Wallis tests, the statistical significance of the observed differences among alternatives. Figure 2 shows hypervolume results for some of the parallel alternatives we have compared. They correspond to good-enough solutions included in Pareto fronts with average hypervolumes between 1.881 and 1.978 and standard deviations among 0.007 and 0.03. The statistical analysis does not show significant hypervolume differences for any pair of alternatives. Thus, although our parallel algorithms are not equivalent to the corresponding sequential procedure with only one subpopulation, they provide solutions with similar classification quality.

4.2. Running time performance

Figure 3 provides the averages of the speedups obtained for the dataset of subject 110 by different platform configurations using 32 CPU threads and/or GPU. In Figure 3a–c, a population of 480 individuals is distributed into 2, 4, 8, and 16 subpopulations of 240, 120, 60, and 30 individuals, respectively. Each subpopulation independently executes generations among migrations. The figures show results for 1 to 5 migrations and, as all algorithms execute 60 generations, 60, 30, 20, 15, and 12 generations of independent evolutions, respectively, are executed by each subpopulation between migrations, also showing improvements in the speedups as the number of subpopulations decreases, or as the number of individuals in the subpopulations increases (the same in all cases, i.e., 480).

With respect to changes in the number of migrations, the speedups remain approximately constant. A migration implies send individuals and cost functions among subpopulations and thus, its cost increases with the number of subpopulations and individuals per subpopulation. Nevertheless, communications should not be more costly than the replacement process because communications are indeed done through the shared memory that stores the information about individuals and their fitness. The main changes shown in the speedups of Figure 3a–c seem to be determined by the number of subpopulations and their size (as more subpopulations mean less individuals per subpopulation). As the number of subpopulations grows, the

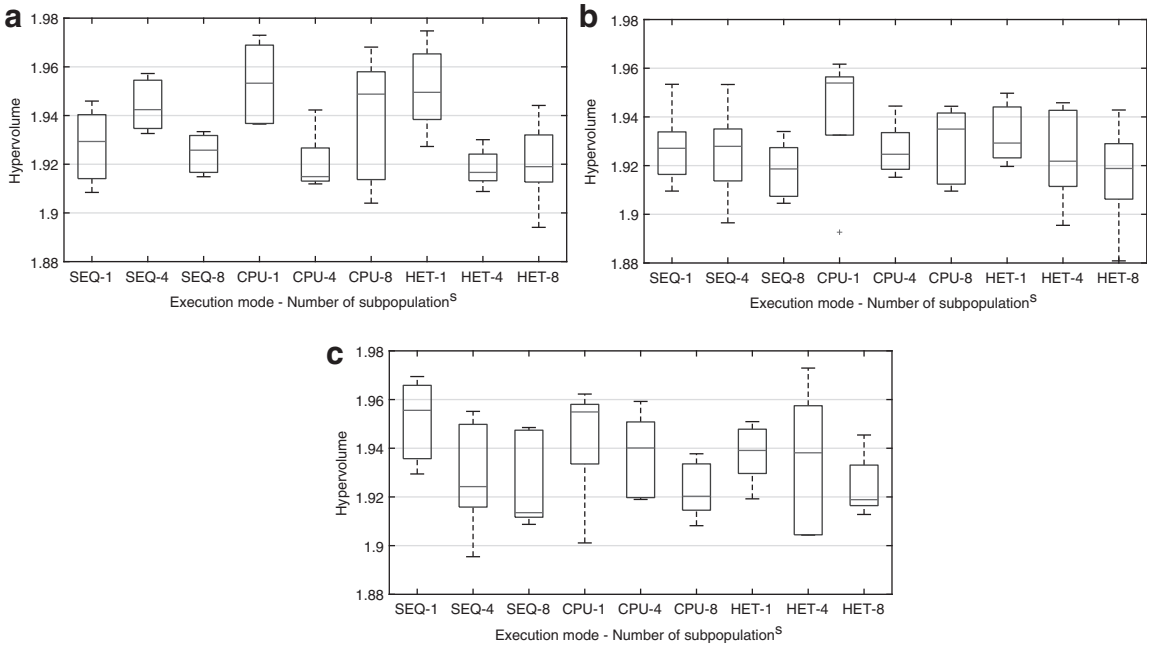

FIG. 2. Hypervolumes obtained using multiple CPU frequencies, and different number of subpopulations and execution modes (SEQ: 1 thread; CPU: 32 threads; HET: CPU+GPU): **(a)** 1.2 GHz; **(b)** 2.1 GHz; **(c)** *Syst* alternative. We only show the results obtained for subject 110, as they are similar to those obtained for the rest of subjects. HET, heterogeneous.

number of calls to the CPU or GPU kernels that allocate a subpopulation to the corresponding device also grows, being costly because a call to the kernel implies to initialize it, and to copy the data to and from the device.

By comparing Figure 3a–c is apparent that the speedups obtained by using only 32 CPU threads are quite similar to those obtained by the GPU. The effect of using both CPU and GPU cores is shown in Figure 3c and d. As Figure 3d shows, the speedup grows as the number of subpopulations decreases, except in the

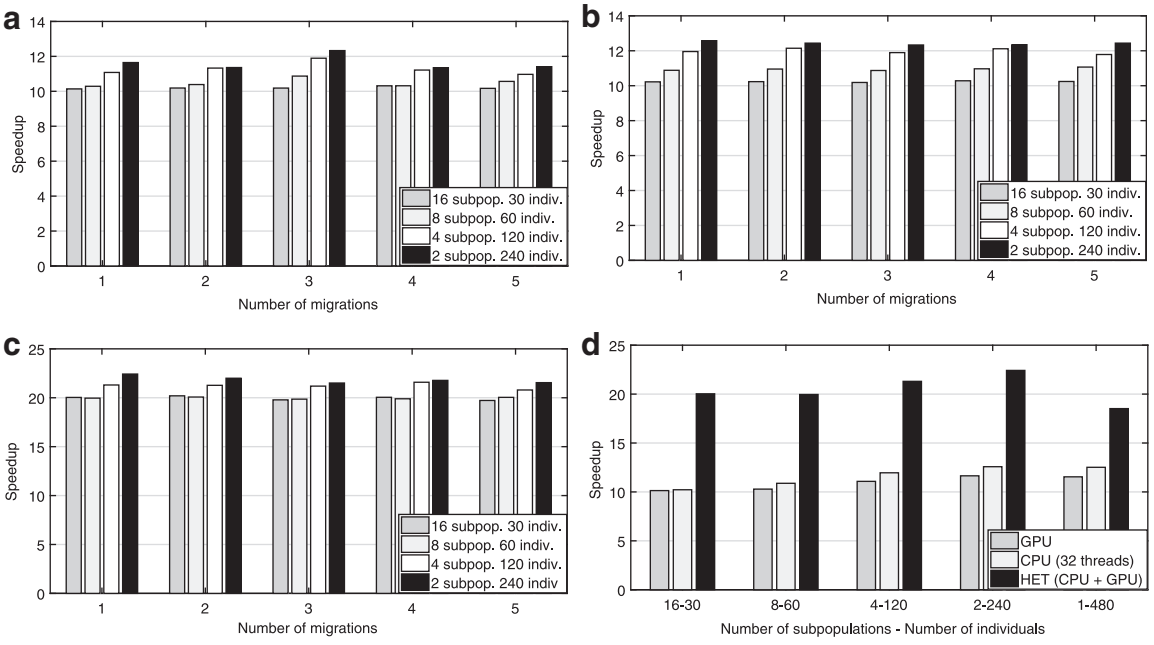

FIG. 3. Averages of speedups achieved with different number of subpopulations and execution modes: **(a)** GPU; **(b)** CPU: 32 threads; **(c)** HET: CPU+GPU; **(d)** Comparison of platforms for different number of subpopulations and individuals per subpopulation. The speedup characteristics for subjects 104 and 107 are quite similar.

case of using only one subpopulation. In this case, the CPU and GPU kernels, respectively, take 32 and 15 individuals (480/32 = 15) and (480/15 = 32), and thus, calls to the CPU or GPU kernels are required. Consequently, the number of calls is higher in the one subpopulation case than in the case of multiple subpopulations.

Figure 4 provides the average of the running time for alternatives corresponding to different values of subpopulations, number of migrations, operating frequencies, and platform configurations. The running time for the parallel alternatives are clearly lower than those corresponding to the sequential executions. It is also clear that the times also decrease in case of using an operating frequency of 2.1 GHz and the *Syst* strategy.

4.3. Energy consumption behavior

The power and the energy consumption in the node has been measured by using a data acquisition system, which we have devised, based on *Arduino Mega*, which gives four real-time measures per second of power and energy consumption. In what follows, we analyze the energy–power behavior of our approach for frequencies of 1.2 and 2.1 GHz, and the *Syst* strategy, as previously described. Figure 5a and b provides the evolution of the instantaneous power consumed along the execution of our procedure in two different parallel configurations: one running in the CPU and the other using the HET mode. These figures show that the lowest instantaneous power values correspond to the 1.2 GHz alternative. The 2.1 GHz and *Syst* alternatives present similar values. Figure 5c and d clearly show the advantage of using the HET mode instead of only CPU. In addition, it is also clear that the highest instantaneous power consumption corresponds to the HET mode, followed by the configuration, which only includes CPU. Nevertheless, the energy consumption depends on the time required to complete the task.

The behavior with respect to energy consumption for different alternatives is shown in Figure 6. From this figure, it is also clear that the parallel alternatives consume less energy. Although the instantaneous power consumed is higher, the achieved speedup allows better consumption figures. Moreover, the energy consumption is also less in case of an operating frequency of 2.1 GHz and in the *Syst*

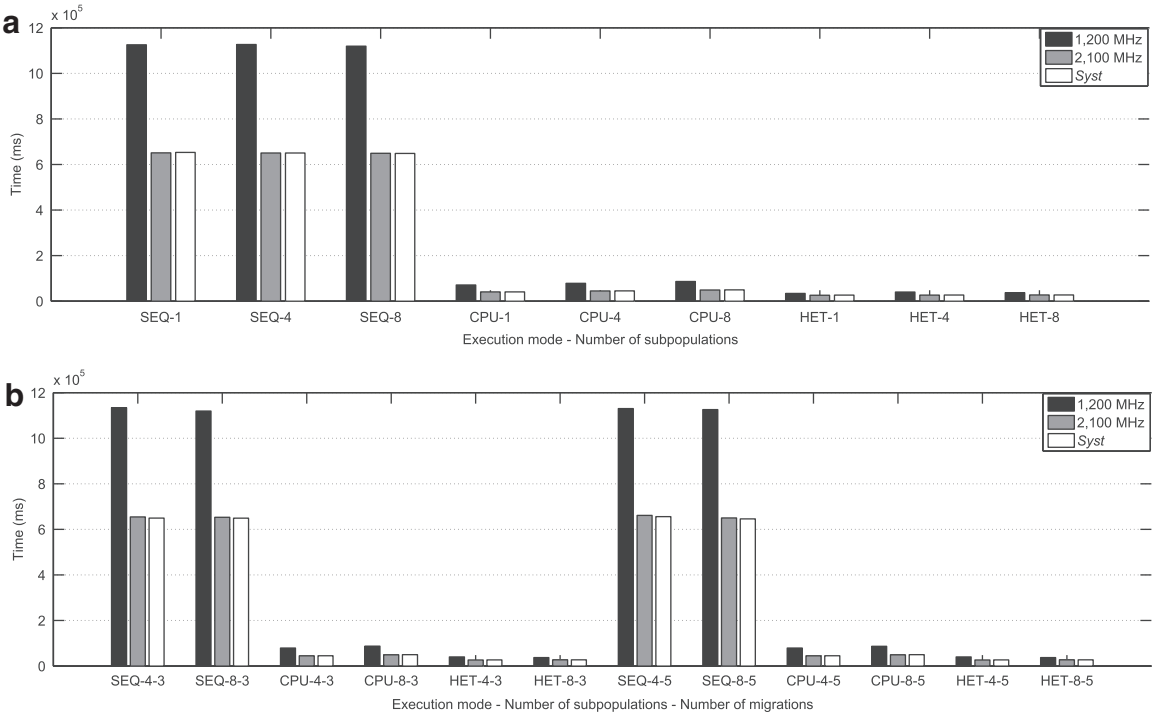

FIG. 4. Running time for different parallel alternatives and operating frequencies, using 60 generations and 480 individuals distributed into multiple number of subpopulations: **(a)** 1 migration; **(b)** 3 and 5 migrations.

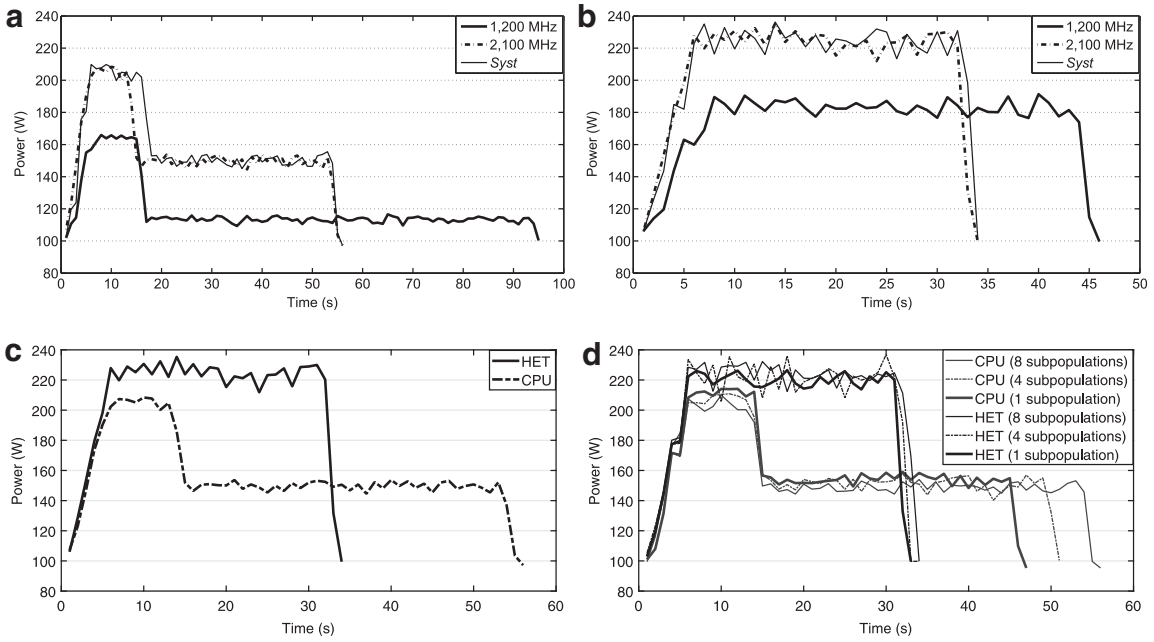

FIG. 5. Power measured along the execution of different parallel alternatives with 60 generations and 480 individuals: **(a)** CPU mode with 8 subpopulations, 5 migrations; **(b)** HET mode with 8 subpopulations, 5 migrations; **(c)** CPU and HET modes with 8 subpopulations, 5 migrations at 2100 MHz; **(d)** CPU and HET modes with 1 migration at 2100 MHz.

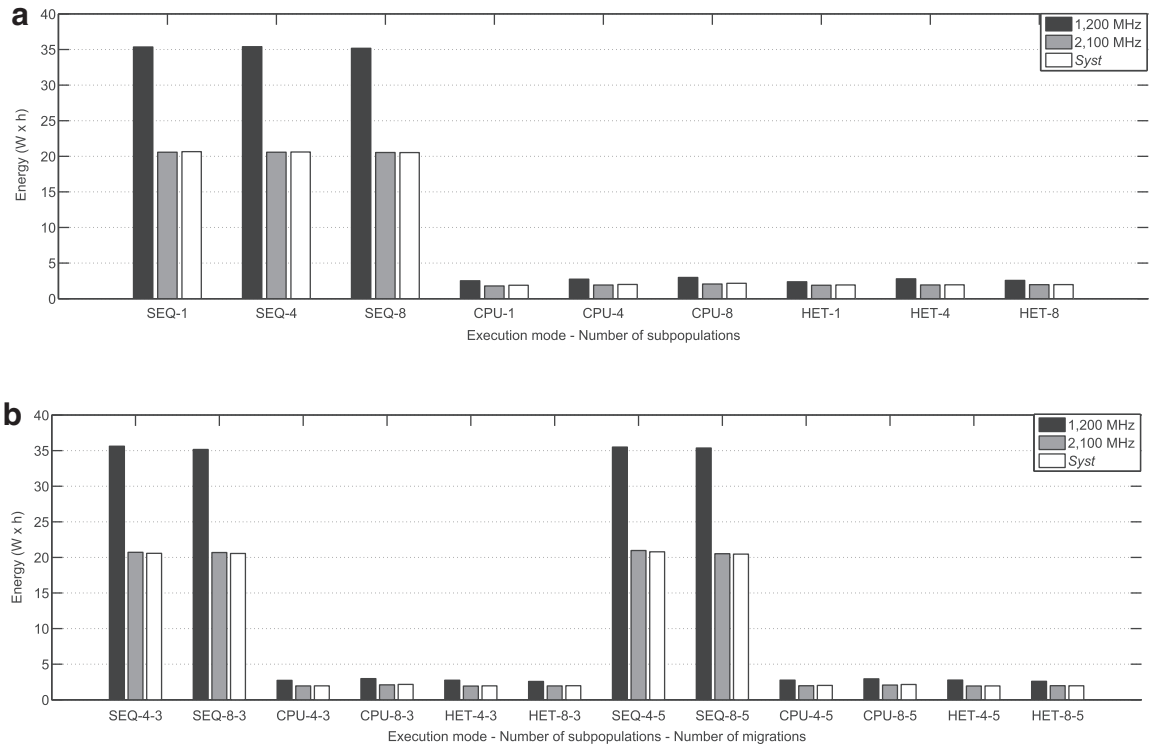

FIG. 6. Energy consumption for different parallel alternatives and operating frequencies, using 60 generations and 480 individuals distributed into multiple number of subpopulations: **(a)** 1 migration; **(b)** 3 and 5 migrations.

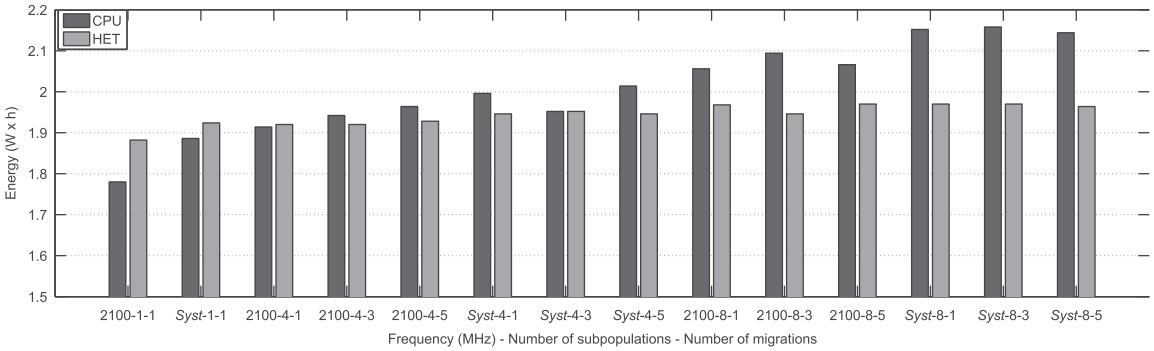

FIG. 7. Comparison of energy consumption for configurations using only CPU threads and HET mode.

strategy. In those two last alternatives, the energy consumption is almost the same. It has to be noticed that the energy measures correspond to the energy consumed by the whole node, including the consumption of buses and memories. Nevertheless, differences are still apparent for different processing architectures.

Figure 7 compares the energy consumption of the CPU mode with respect to the HET mode. It shows that the CPU alternative only implies less energy consumption in case of one subpopulation for both 2.1 GHz and *Syst* alternatives. The Kruskal–Wallis test shows statistically significant differences in these two cases. Instead, the application of the Kruskal–Wallis test does not detect statistically significant differences between CPU and HET alternatives in case of four subpopulations. Finally, the HET mode shows lower energy consumption than the CPU one in case of eight subpopulations. In this case, the differences are statistically significant according to the Kruskal–Wallis test. Thus, although the HET mode shows higher instantaneous power consumptions than the CPU alternative, the lower execution time of the HET mode in conjunction with an increase in the energy consumed by the CPU alternative when more subpopulations are evolved would explain this behavior. A similar behavior can also be seen in Figure 8.

From Table 1, it can be seen that there are statistically significant differences in all cases considered when the parallel code is executed only by CPU threads. Instead, when the parallel code is executed by both, CPU threads and GPU, there are only statistically significant differences between four and eight subpopulations in the 2.1 GHz case. Even in these cases with significant differences, the *p*-values are near

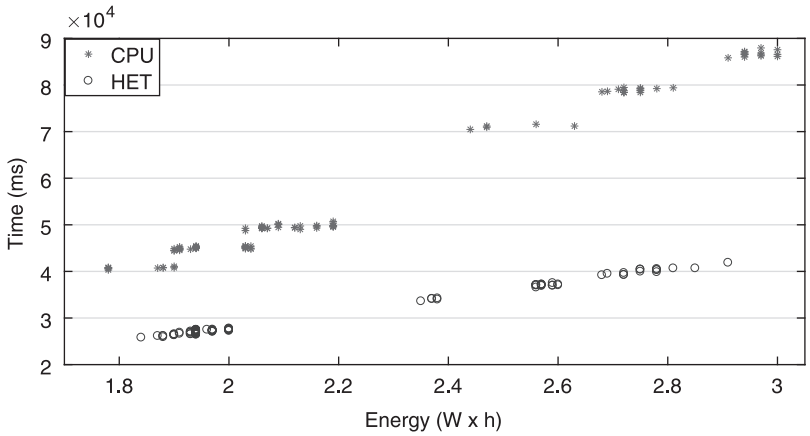

FIG. 8. Time versus energy for different parallel configurations. The alternatives, which use CPU threads and GPU, provide the lowest values of running time. One of the heterogeneous alternatives even shows a very low energy consumption, despite that the lowest energy consumption values correspond to parallel configurations that only involve CPU threads.

Table 1. *p*-Values Obtained from Kruskal–Wallis Test to Analyze the Statistical Significance of Differences in the Energy Consumption for Different Parallel Configurations with Multiple Number of Subpopulations and Migrations, N_M ($p < 0.05$ Means Statistically Significance)

N_M	*No. of subpopulations*	*CPU (2.1 GHz)*	*CPU (Syst)*	*HET (2.1 GHz)*	*HET (Syst)*
1	1 vs. 4	0.005	0.008	**0.091**	**0.093**
	4 vs. 8	0.007	0.008	0.030	**0.155**
3	4 vs. 8	0.012	0.007	0.045	**0.282**
5	4 vs. 8	0.007	0.008	0.045	**0.155**

CPU, Central Processing Unit; HET, heterogeneous.
Bold indicates not statistically significant.

($p = 0.05$). In all cases, the statistically significant differences correspond to increases with the number of subpopulations.

5. CONCLUSIONS

This article proposes and analyzes parallel heterogeneous implementations of a multiobjective feature selection procedure applied to EEG classification on BCI tasks, which take advantage of both CPU and GPU architectures. It uses OpenMP threads to distribute the workload and OpenCL kernels to perform the fitness evaluation of the individuals in each subpopulation. The *K*-means algorithm to evaluate the individual fitness is also parallelized through the GPU cores, with the objective of taking advantage of the data parallel GPU capabilities.

The experimental evaluation of our approach has been done in terms of speedup and energy consumption for different alternatives of subpopulations, migrations, and platform configurations. It has been shown that, for some subpopulation and migration alternatives, the configuration, including both CPU and GPU devices provides not only better speedup results, but also lower energy consumption than the corresponding alternatives. Compared with a master–worker parallel implementation (the alternative corresponding to only one subpopulation), the heterogeneous configuration still provides the highest speedups, and depending on their specific values, its energy consumption could be higher or lower than the corresponding CPU configuration. Moreover, although the instantaneous power is higher for HET configurations than for the corresponding ones using only CPU threads, the runtime are lower for the HET mode. On the other hand, the energy consumption measured for the HET configuration almost does not show any significant changes with the number of subpopulations. In this case, the energy consumed by components not included in the CPU (buses, RAM memory, etc.) could mask the differences in the energy consumption.

Among other approaches that should be explored to take advantage of both CPU and GPU architectures, a message-passing implementation could offer new insights about speed and energy behavior of heterogeneous architectures for applications that demand a high amount of heterogeneous parallelism. Moreover, building accurate quantitative models for energy consumption will also be very useful to identify energy–performance-efficient workload distributions.

ACKNOWLEDGMENTS

The work was funded by project TIN2015-67020-P (Spanish ''Ministerio de Economía y Competitividad'' and ERDF funds). The authors would like to thank the BCI laboratory of the University of Essex, especially Prof. John Q. Gan, for allowing us to use their databases. The authors also thank the reviewers for their useful comments and suggestions.

AUTHOR DISCLOSURE STATEMENT

The authors declare that no competing financial interests exist.

REFERENCES

Aliaga, J., Barreda, M., Dolz, M., et al. 2014. Assessing the impact of the CPU power-saving modes on the task-parallel solution of sparse linear systems. *Clust. Comput*. 17:1335–1348.

Allen, T., and Ge, R. 2016. Characterizing power and performance of GPU memory access, 46–53. *In Proceedings of the 4th International Workshop on Energy Efficient Supercomputing*, E2SC'2016, Salt Lake City, UT. IEEE Press.

Asensio-Cubero, J., Gan, J., and Palaniappan, R. 2013. Multiresolution analysis over simple graphs for brain computer interfaces. *J. Neural Eng*. 10:046014.

Baskiyar, S., and Abdel-Kader, R. 2010. Energy aware DAG scheduling on heterogeneous systems. *Clust. Comput*. 13:373–383.

Collet, P. 2013. Why GPGPUs for evolutionary computation? 3–14. *In* Tsutsui, S., and Collet, P., eds. *Massively Parallel Evolutionary Computation on GPGPUs*, Natural Computing Series. Springer: Berlin-Heidelberg.

Cotta, C., Fernández-Leiva, A., Fernández-de-Vega, F., et al. 2015. Ephemeral computing and bioinspired optimization: Challenges and opportunities, 319–324. *In Proceedings of the 7th International Conference on Evolutionary Computation Theory and Applications*, ECTA'2015, Lisbon, Portugal. IEEE.

Deb, K., Agrawal, S., Pratap, A., et al. 2000. A fast elitist non-dominated sorting genetic algorithm for multi-objective optimization: NSGA-II, 849–858. *In Proceedings of the 6th International Conference on Parallel Problem Solving from Nature*, PPSN VI, Paris, France. Springer.

Dorronsoro, B., Nesmachnow, S., Taheri, J., et al. 2014. A hierarchical approach for energy-efficient scheduling of large workloads in multicore distributed systems. *Sustain. Comp. Inf. Syst*. 4:252–261.

Duin, R. 2000. Classifiers in almost empty spaces, 1–7. *In Proceedings of the 15th International Conference on Pattern Recognition*, ICPR'2000, Barcelona, Spain. IEEE.

Escobar, J. Ortega, J., Díaz, A., et al. 2017a. Power-performance evaluation of parallel multi-objective EEG feature selection on CPU-GPU platforms, 580–590. *In Proceedings of the 17th International Conference on Algorithms and Architectures for Parallel Processing*, ICA3PP'2017, Helsinki, Finland, Springer.

Escobar, J., Ortega, J., González, J., et al. 2016a. Assessing parallel heterogeneous computer architectures for multi-objective feature selection on EEG classification, 277–289. *In* Ortuño, F., and Rojas, I., eds. *Proceedings of the 4th International Conference on Bioinformatics and Biomedical Engineering*, IWBBIO'2016, Granada, Spain. Springer.

Escobar, J., Ortega, J., González, J., et al. 2016b. Improving memory accesses for heterogeneous parallel multi-objective feature selection on EEG classification, 372–383. *In Proceedings of the 4th International Workshop on Parallelism in Bioinformatics*, PBIO'2016, Grenoble, France. Springer.

Escobar, J., Ortega, J., González, J., et al. 2017b. Parallel high-dimensional multi-objective feature selection for EEG classification with dynamic workload balancing on CPU-GPU. *Clust. Comput*. 20:1881–1897.

Escobar, J., Ortega, J., González, J., et al. 2017c. Issues on GPU parallel implementation of evolutionary high-dimensional multi-objective feature selection, 773–788. *In Proceedings of the 20th European Conference on Applications of Evolutionary Computation, Part I*, EVOSTAR'2017, Amsterdam, The Netherlands. Springer.

Fernández-de-Vega, F., Chávez, F., Díaz, J., et al. 2016. A cross-platform assessment of energy consumption in evolutionary algorithms, 548–557. *In Proceedings of the 14th International Conference on Parallel Problem Solving from Nature*, PPSN'2016, Edinburgh, UK. Springer.

Fonseca, C., López-Ibáñez, M., Paquete, L., et al. 2018. Computation of the hypervolume indicator. Available at: http://lopez-ibanez.eu/hypervolume. Last accessed: February 2, 2018.

Handl, J., and Knowles, J. 2006. Feature subset selection in unsupervised learning via multiobjective optimization. *Int. J. Comput. Intell. Res*. 2:217–238.

Jähne, P. 2016. Overview of the current state of research on parallelisation of evolutionary algorithms on graphic cards, 2163–2174. *In GI-Jahrestagung*, INFORMATIK'2016, Bonn, Germany. LNI.

Lee, Y., and Zomaya, A. 2011. Energy conscious scheduling for distributed computing systems under different operating conditions. *IEEE Trans. Parallel Distrib. Syst*. 22:1374–1381.

Luong, T., Melab, N., and Talbi, E.-G. 2010. GPU-based island model for evolutionary algorithms, 1089–1096. *In Proceedings of the 12th Annual Conference on Genetic and Evolutionary Computation*, GECCO'2010, Portland, OR. ACM.

Ma, K., Li, X., Chen, W., et al. 2012. GreenGPU: A holistic approach to energy efficiency in GPU-CPU heterogeneous architectures, 48–57. *In Proceedings of the 41st International Conference on Parallel Processing*, ICPP'2012, Pittsburgh, PA. IEEE.

Marinaki, M., and Marinakis, Y. 2014. An island memetic differential evolution algorithm for the feature selection problem, 29–42. *In Proceedings of the 6th International Workshop on Nature Inspired Cooperative Strategies for Optimization*, NICSO'2013, Canterbury, UK. Springer.

Marowka, A. 2012. Energy consumption modeling for hybrid computing, 54–64. *In Proceedings of the 18th International Conference on Parallel Processing, Euro-Par 2012*, Euro-Par'2012, Rhodes Island, Greece. Springer.

Mukhopadhyay, A., Maulik, U., Bandyopadhyay, S., et al. 2014a. A survey of multiobjective evolutionary algorithms for data mining: Part II. *IEEE Trans. Evol. Comput*. 18:4–19.

Mukhopadhyay, A., Maulik, U., Bandyopadhyay, S., et al. 2014b. A survey of multiobjective evolutionary algorithms for data mining: Part II. *IEEE Trans. Evol. Comput*. 18:20–35.

Nesmachnow, S., Dorronsoro, B., Pecero, J., et al. 2013. Energy-aware scheduling on multicore heterogeneous grid computing systems. *J. Grid Comput*. 11:653–680.

Rotem, E. Weiser, U., Mendelson, A., et al. 2016. H-EARTH: Heterogeneous multicore platform energy management. *IEEE Comput. Mag*. 49:47–55.

Rupp, R., Kleih, S., Leeb, R., et al. 2014. Brain-computer interfaces and assistive technology, 7–38. *In* Grbler, G., and Hildt, E., eds. *Brain-Computer-Interfaces in their Ethical, Social and Cultural Contexts*, The International Library of Ethics, Law and Technology. Springer: Dordrecht, The Netherlands.

Sharma, D., and Collet, P. 2013. Implementation techniques for massively parallel multi-objective optimization, 267–286. *In* Tsutsui, S., and Collet, P., eds. *Massively Parallel Evolutionary Computation on GPGPUs*, Natural Computing Series. Springer: Berlin-Heidelberg.

Wong, M., and Cui, G. 2013. Data mining using parallel multi-objective evolutionary algorithms on graphics processing units, 287–307. *In* Tsutsui, S., and Collet, P., eds. *Massively Parallel Evolutionary Computation on GPGPUs*, Natural Computing Series. Springer: Berlin-Heidelberg.

Zhang, Y. Hu, X., and Chen, D. 2002. Task scheduling and voltage selection for energy minimization, 183–188. *In Proceedings of the 39th Annual Design Automation Conference*, DAC'2002, New Orleans, LA. ACM.

Address correspondence to:
Juan José Escobar, MSc
Department of Computer Architecture and Technology
CITIC
University of Granada
Calle Periodista Rafael Gómez Montero, 2
Granada 18014
Spain

E-mail: jjescobar@ugr.es