Considerando el consumo energético en los Algoritmos Evolutivos

Francisco Fernández de Vega, Josefa Díaz, Juan A. García, and Francisco Chávez

Grupo de Evolución Artificial, Universidad de Extremadura
fcofdez@unex.es, fchavez@unex.es, mjdiaz@unex.es, jangelgm@unex.es

Resumen Los algoritmos evolutivos se han medido tradicionalmente en función de la calidad de soluciones que suministran y el tiempo necesario para llegar a la misma. Aunque con frecuencia esta medida del tiempo se realiza indirectamente mediante el número de generaciones que hay que computar hasta llegar a la solución, en otras ocasiones, y debido a la influencia de parámetros como el número de individuos en la población, o al uso de cromosomas de tamaño variable, es necesario realizar una medida del tiempo de cómputo hasta la solución a la hora de comparar diferentes alternativas. En este trabajo presentamos una nueva medida que en este dominio no se tiene habitualmente en cuenta: el consumo energético. Como mostraremos experimentalmente, y del mismo modo que es necesario considerar no sólo el número de generaciones computadas dependiendo del algoritmo que se estudie, veremos que un mismo algoritmo puede presentar valores muy distintos de consumo energético dependiendo del hardware en que se ejecute; y que este valor en combinación con el tiempo de ejecución son útiles para determinar el modo más adecuado de ejecución del algoritmo.

PALABRAS CLAVE: Algoritmos Evolutivos, Consumo energético

1. Introducción

Tradicionalmente, el análisis de algoritmos se ha realizado desde el punto de vista de la complejidad computacional. Desde que Turing definiera la capacidad computacional mediante un modelo teórico, el orden de los algoritmos permiten entender sus capacidades [14]. No obstante, esto no ha impedido que los fabricantes de hardware hayan intentado ofrecer nuevas soluciones que permitan ejecutar cualquier algoritmo en el menor tiempo posible. El tiempo de cálculo, y en definitiva el *rendimiento*, que cada arquitectura ofrece, ha sido caballo de batalla en la mejora progresiva de las últimas décadas, lo que ha permitido ofrecer a los usuarios arquitecturas *multi-core* y *many-core* a precios competitivos. Los usuarios han tenido así a su disposición en los últimos años, equipos con capacidades de cálculo impensables en otras épocas, pero a su vez se han visto en la necesidad de aplicar nuevos modelos de programación paralela, aprender nuevas metodologías y utilizar librerías especializadas.

Actas de la XVII Conferencia de la Asociación Española para la Inteligencia Artificial, pp. 367-376

A pesar de todas estas ventajas, hay un elemento primordial y necesario para ejecutar cualquier proceso que ha sido frecuentemente olvidado: el consumo energético. Aunque las nuevas arquitecturas hardware, si son bien aprovechadas, permiten una reducción significativa de los tiempos de cómputo, en pocas ocasiones se ha analizado el coste energético -y en definitiva el coste económico- que implica su uso.

El término *Green Computing* surgió en la última década para enfocar adecuadamente el problema del consumo energético asociado a la computación, particularmente en las grandes instalaciones (Data centers) [1]. Pero esta nueva sensibilidad hacia problemas medioambientales, y económicos, es aplicable igualmente a sistemas de cómputo personales y algoritmos de cualquier naturaleza.

En el caso particular de los Algoritmos Evolutivos (AE), se ha prestado esfuerzo notable a la mejora de su rendimiento mediante la utilización de hardware paralelo combinado con modelos distribuidos [15]. Las mejoras siempre han tratado de analizar la calidad global de la solución obtenida frente al tiempo de cálculo necesario. Pero de igual modo que un viajero no siempre elige el medio de transporte más rápido para llegar a su destino, sino que muy al contrario, frecuentemente, y por cuestiones de economía, elige medios de transporte más lentos, vuelos con varias escalas en lugar de trayectos directos, o vehículo propio en lugar de taxista profesional, en la ejecución de los algoritmos deberíamos tener también en cuenta las cuestiones del consumo energético cuando se busca llegar a una solución para un problema.

Hasta donde sabemos, y aunque la literatura en el área ha reconocido ya la importancia del consumo energético en los algoritmos evolutivos [5], todavía no se ha analizado el impacto que tiene los modelos de AE disponibles. Este es el objetivo principal de este trabajo, analizar el consumo energético como un nuevo parámetro importante en la toma de decisiones sobre el diseño de Algoritmos Evolutivos.

2. Algoritmos evolutivos y consumo energético

El interés hacia la eficiencia energética en la computación se despertó hace ya algún tiempo, dando origen en primer lugar al conocido término *Green Computing* [1], y por otro al *energy-aware* [6], que podríamos en castellano traducir como *Computación verde* (o ecológica) y *sensibilidad al consumo energético* respectivamente. Incluso los fabricantes de procesadores contemplan ya la posibilidad de adaptar la velocidad de ejecución de los mismos, y por tanto su consumo energético, a las necesidades de los algoritmos, en lo que se ha denominado *escalado dinámico de la veolocidad* [8], e investigaciones recientes tratan de aprovechar esta capacidad para que los algoritmos regulen la velocidad del procesador en función de las necesidades de cada instante [9], [10].

Los algoritmos evolutivos, como métodos generales de optimización, han sido utilizados en contextos asociados a gestión energética. Así, podemos encontrar problemas de optimización asociados a HVAC (Energy management of heating, ventilating and air-conditioning) que han sido abordados mediante AE [2], [3].

Existen otros trabajos que tratan el problema de la distribución de la energía eléctrica [4]. Pero tanto este como los anteriores trabajos, son tangenciales con respecto al problema que aquí queremos abordar: el consumo energético del algoritmo evolutivo, en relación a los parámetros que lo definen y el hardware en el que se ejecutan.

El concepto principal que nos interesa aquí es la capacidad del propio algoritmo para poder adaptarse a entornos cambiantes en los que el consumo energético es un factor importante de diseño [**?**]. Esta capacidad, que podríamos inscribir en el dominio de las propiedades *auto** de un algoritmo, incluido los evolutivos [5], ha sido considerada ya por los investigadores en otros tipos de algoritmos. Así la importancia de la capacidad adaptativa desde el punto de vista energético se tiene ya en cuenta en el diseño de algoritmos y arquitecturas [6], y es un factor clave en todas las escalas, tanto en el diseño de los grandes centros de procesos de datos, como en las arquitecturas destinadas a pequeños dispositivos móviles, en los que la duración de la batería es crucial para el usuario. También en este último contexto, los algoritmos evolutivos se han tenido en cuenta para el diseño de arquitecturas de procesador que consigan un menor consumo energético reduciendo a la vez la disipación de energía calorífica [7].

No obstante, hasta donde sabemos, el algoritmo evolutivo nunca se ha estudiado desde el punto de vista de sus necesidades energéticas. Dada la naturaleza estocástica del mismo, el conjunto de parámetros que afecta a su comportamiento en los procesos de búsqueda de soluciones a problemas difíciles, y la cantidad de plataformas hardware que se han utilizado para ejecutarlos, sería conveniente conocer cúal es el la energía consumida para encontrar una solución, y cómo los parámetros y hardware utilizado afectan el proceso de búsqueda y el consumo energético.

Este trabajo es una primera aproximación al estudio del consumo energético para el Algoritmo Genético, y mostrará los resultados encontrados en diferentes situaciones, permitiendo un primer análisis del comportamiento general y conclusiones sobre los mejores escenarios para la utilización de los mismos.

El resto de artículo se organiza del siguiente modo: en la sección 3 describimos la metodología que hemos seguido para analizar el consumo energético de un algoritmo evolutivo. La sección 4 muestra los resultados obtenidos, y finalmente en la sección 5 presentamos las conclusiones de este primer trabajo en la temática.

3. Metodología

El estudio preliminar que queremos abordar consiste en realizar una estimación del consumo energético requerido por un algoritmo evolutivo para encontrar una solución para un problema dado.

Teniendo en cuenta la naturaleza estocástica de este tipo de algoritmos, lo primero que se decidió es que cada posible experimento sería repetido 30 veces para obtener valores promedio. Por otro lado, y dado que el mismo algoritmo queremos ejecutarlo en diferentes plataformas hardware, decidimos utilizar valores fijos para la semilla aleatoria del algoritmo, así se preseleccionaron 30

semillas, para ser utilizadas cada una de ellas en una de las 30 ejecuciones a realizar de cada experimento.

Por otro lado, y dada la influencia que tendrá el tiempo de ejecución en el consumo total de energía del algoritmo, se decidió configurar el bucle principal del algoritmo evolutivo para que finalizara cuando un determinado nivel de calidad fuera alcanzado, evitando así que el algoritmo continúe trabajando cuando la solución ha sido ya encontrada. El dato que más nos interesa es el tiempo medio necesario para encontrar la solución a lo largo de las 30 ejecuciones en cada plataforma hardware, además por supuesto del consumo energético del algoritmo en ese hardware. Dado que las semillas aleatorias serán las mismas, y el código fuente del algoritmo y sus parámetros exactamente los mismos, sabemos que las operaciones ejecutadas en alto nivel son exactamente las mismas, y las diferencias obtenidas en el tiempo de ejecución serían debidas exclusivamente al hardware: arquitectura de instrucciones, velocidad del procesador, etc, elementos que no son objeto de estudio en este trabajo.

Por último, cabe destacar que aunque son muchos los parámetros que influyen en el proceso de convergencia de los algoritmos evolutivos, y por tanto del tiempo necesario para llegar a una solución, en este estudio preliminar nos hemos fijado solamente en el tamaño de la población, y hemos experimentado con tamaños que van desde 50 individuos hasta un millar. Hemos utilizado como problema de test un problema de regresión, al que se le ha aplicado el algoritmo genético estándar.

Para simplificar los procesos de compilación en las diferentes plataformas elegidas, hemos utilizado un algoritmo genético simple, basado en las especificaciones presentadas en [11] y tomando como punto de partida la implementación disponible en [13], el cual se ha modificado y adaptado convenientemente. En la Tabla 1 se describen las diferentes arquitecturas consideradas para la experimentación.

Cuadro 1. Característica de los dispositivos.

Dispositivo	Procesador	Núcleos	RAM
Raspberry Pi	Cortex-A7 (900 Mhz)	4	1GB
Laptop	Intel(R) Core(TM) i5-2450M (2.50GHz)	4	8GB
Xilinx SoC	ZYNQ-7000 ZC702 (50Mhz)		
Smartphone	Qualcomm		
Samsung Galaxy Note 3	Snapdragon 800 (2.3Ghz)	4	3Gb

Dado que el principal objetivo es la medición del consumo energético, para realizar una estimación adecuada hemos procedido del siguiente modo: Utilizando un voltímetro hemos medido en la toma de corriente eléctrica del equipo (en el caso del PC, por ejemplo) la intensidad de corriente cuando el algoritmo *no* se está ejecutando, y posteriormente cuando el algoritmo *sí* se ejecuta. Hemos almacenado el consumo instantáneo que nos proporciona el voltímetro durante todo el tiempo que dura la ejecución del algoritmo, para con este valor de intensidad de corriente que puede variar a lo largo del tiempo, además del voltaje al que trabaja el equipo, poder hacer el cálculo lo más exacto posible del consumo total energético del algoritmo en ese periodo. Así, al realizar la resta de los dos valores promedios obtenido, las medidas eliminan los componentes de consumo

del resto de elementos de cada equipo, ya sean periféricos, sistema operativo, etc.

En el caso del Smartphone, hemos utilizado la aplicación denominada *PowerTutor* [16],[17]. Esta aplicación instalada en el dispositivo móvil nos permite monitorizar el consumo de cada uno de los procesos lanzados en el dispositivo, permiténdonos a su vez calcular el consumo energético de la ejecución del algoritmo genético.

Los datos que mostremos a continuación de consumo se muestran en unidades de Kilovatios hora (Kwh). Esta será nuestra medida básica que permita comparar los consumos en diferentes circunstancias y sacar las conclusiones que correspondan.

Por otro lado, el problema a resolver es la búsqueda del valor máximo en una función escalar **F** definida por cinco variables, siguiendo las directrices presentadas en [11]. Cada uno de estos valores, proporcionados como parámetro, deben estar comprendidos entre un valor mínimo y uno máximo. El objetivo es encontrar los valores óptimos de las cinco variables para que la función F alcance un valor máximo dado. La función de fitness se especifica en la Ecuación 1. La descripción de los operadores genéticos se muestra en la Tabla 2. El número máximo de generaciones se ha establecido a 10000, con un criterio de parada definido por el porcentaje que representa el fitness de la mejor solución respecto a la solución óptima $(0,999\,\%)$, conocida anticipadamente. Los resultados experimentales se realizan para cuatro tamaños de población diferentes y a los operadores de cruce y mutación se les han asignado valores dentro de los rangos generalmente recomendados.

En el caso de la FPGA de Xilinx, utilizando el kit de evaluación indicado en la Tabla 1 se implementa el algoritmo con algunas modificaciones. Dadas las características del dispositivo, el algoritmo programado no incluye una finalización con criterio de parada, sino que se ha utilizado el valor medio del número de generaciones obtenido al aplicar dicho criterio en el dispositivo Raspberry, para cada tamaño de población. Asi pues, se generan cuatro implementaciones, una para cada tamaño de población, y en cada implementación se limita el número máximo de generaciones al valor medio obtenido. Para los tamaños de población elegidos, de 50, 100, 500 y 1000 individuos, se han elegidos un número máximo de generaciones de 6505, 5859, 2732 y 1684, respectivamente. Las características particulares del entorno donde se implementa el algoritmo son Versión: Vivado v.2015.4 (win64) Build 1412921 Wed Nov 18 09:43:45 MST 2015, Dispositivo: xc7z020clg484-1.

Para implementar el algoritmo en la FPGA, en primer lugar se depura el código C, se transforma en VHDL y se crea un módulo IP, que en el diseño al SoC xc7z020 de Xilinx, requiere utilizar protocolo AMBA para comunicarse mediante los buses de E/S. Completado el diseño a incluir en el SoC (PS+AMBA+modulo IP del algoritmo) se realiza la verificación, que proporciona el número de ciclos de reloj que necesita el SoC para determinar el valor de ajuste final a partir de un valor de una semilla, y que son los ciclos de reloj (o tiempo) que el SoC necesita para estar en disposición de admitir otro valor de entrada (otra semilla)

para dar otro valor de salida. Este número de ciclos, multiplicado por la duración (en segundos) de un ciclo, multiplicado por el número de veces que se repite (30 semillas) da un valor en tiempo que se puede comparar (en forma, es decir, para los mismos cálculos que se hacen en los casos de Android y la Raspberry Pi) con los otros resultados.

Al proceso de verificación le sigue el proceso de síntesis, que proporciona en watios el consumo del SoC completo en la placa Zynq.

$$F = x_1^4 + (x_2^3 * x_3^2) - (x_3 * x_4) + x_5 \quad (1)$$

Cuadro 2. Parámetros para GA.

Número máximo generaciones	10000
Tamaño población	50,100,500,1000
Probabilidad cruce	0,8
Probabilidad mutación	0,15

4. Resultados

Para cada experimento realizado, se muestra información relativa al consumo en vatios (w), el tiempo de ejecución (seg) y el consumo total en Kilovatios hora (kwh). Este último es la medida estándar de facturación utilizada por las compañías eléctricas.

La Tabla 3 muestra los resultados obtenidos para un tamaño de población de 50, 100, 500 y 1000 individuos respectivamente. Por otro lado, la figura 1 muestra gráficamente los datos de la tabla 3, donde podemos observar la evolución del tiempo frente al consumo de los diferentes dispositivos utilizados en este trabajo.

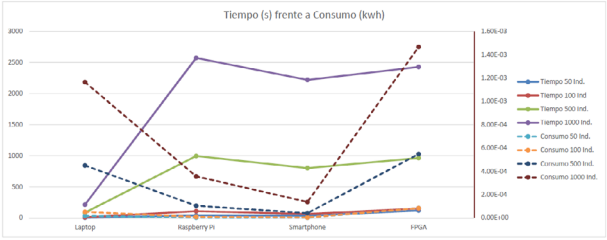

Figura 1. Tiempo (s) frente a Consumo (kwh) de los diferentes dispositivos utilizados

La primeras conclusión que podemos obtener, es que efectivamente, el hardware influye notablemente en la energía necesaria para encontrar una solución. Observamos que el equipo tipo PC consume mucha más potencia eléctrica que el

Cuadro 3. Resultados experimentales

Dispositivo	Consumo (W)	Tiempo (seg.)	Consumo (Kwh)
Tamaño de población = 50			
Raspberry Pi	0,32	41,66	$0,370 * 10^{-5}$
Laptop	12,76	5,1	$1,81 * 10^{-5}$
Smartphone	0,27	35,48	$0,272 * 10^{-5}$
FPGA	1,982	60,523	$3,332 * 10^{-5}$
Tamaño de población = 100			
Raspberry Pi	0,32	109,89	$0,977 * 10^{-5}$
Laptop	15,76	12,11	$5,3 * 10^{-5}$
Smartphone	0,26	65,75	$0,48 * 10^{-5}$
FPGA	1,990	153,06	$8,4608 * 10^{-5}$
Tamaño de población = 500			
Raspberry Pi	0,38	996,61	$10,520 * 10^{-5}$
Laptop	18,6	87,16	$45,034 * 10^{-5}$
Smartphone	0,18	802,26	$4,185 * 10^{-5}$
FPGA	2,050	964,27	$54,909 * 10^{-5}$
Tamaño de población = 1000			
Raspberry Pi	0,5	2572,8	$35,733 * 10^{-5}$
Laptop	19,49	215,33	$116,475 * 10^{-5}$
Smartphone	0,22	2219,80	$13,807 * 10^{-5}$
FPGA	2,172	2429,97	$146,60 * 10^{-5}$

resto de dispositivos. Si nos fijamos por ejemplo en las ejecuciones con tamaños de población de 50 individuos el consumo es de 12,76 vatios en el laptop, frente a menos de 2 vatios para el resto; en algunos casos la diferencia está en el orden de 40 veces más energía en cada instante de tiempo para el PC (frente a la raspberry, por ejemplo). Sin embargo, y debido a la mayor potencia computacional del PC, el tiempo necesario hasta la solución es bastante más pequeño, como cabría esperar: 5 segundos en el caso del PC frente a los 41 y 35 respectivamente para la raspberry y el teléfono móvil, y más de 60 para la FPGA. Pero si tenemos en cuenta ambos elementos, potencia consumida y tiempo de ejecución, para hacer un cálculo de kilovatios/hora, que es un dato objetivo resumido del consumo total para el algoritmo en cuestión, los datos siguen a favor de los dispositivos con menor capacidad de cómputo $0,370 * 10^{-5}$ y $0,272 * 10^{-5}$ Kwh para raspberry y Smartphone respectivamente frente a $1,81 * 10^{-5}$ Kwh en el PC y $3,33 * 10^{-5}$ en el caso de la FPGA. Descartando la FPGA como dispositivo de interés, dado en primer lugar su más difícil utilización, y por otro lado su mayor consumo cuando se sintentiza directamente un algoritmo de alto nivel, y fijándonos en el resto de dispositivos, podemos concluir que es, aproximadamente, 6 veces más

costoso encontrar una solución para el problema en cuestión en un PC que en el resto de dispositivos.

Similares conclusiones podemos obtener si analizamos los experimentos con mayor tamaño de población. En el experimento con la mayor población utilizada, 1000 individuos, y descartando el interés de la FPGA por las razones ya expuestas, el dispositivo que menos ejergía ha consumido para encontrar la solución es el Smartphone, con un consumo casi 10 veces inferior al PC, aunque el tiempo total ha sido 10 veces superior al del PC. La raspberry ha sido el dispositivo más lento, algo más lento que el Smartphone, pero quedando en segundo lugar en cuanto a consumo, con $35,733 * 10^{-5}$ Kwh, doble que el Smartphone pero aproximadamente 3,3 veces inferior al consumo del PC.

Pero un análisis más profundo de los resultados nos suministra información interesante. Si comparamos el consumo instantaneo de energía para un mismo dispositivo, con los diferentes tamaños de población vemos que ese consumo varía, creciendo con el tamaño de población utilizado (salvo en el caso del smartphone, que habrá que estudiar en el futuro con más detenimiento). Así, el consumo varía desde 12,76 a 19,49 vatios cuando pasamos de 50 a 1000 individuos en el PC. Teniendo en cuenta que el algoritmo es el mismo, y es un algoritmo secuencial, la causa hay que buscarla en algún componente externo. Es lógico que el tiempo de cómputo pueda variar entre ejecuciones con diferente tamaño de población, aunque la influencia de este parámetro en el tiempo hasta encontrar una solución no puede preveerse de antemano, dado el modo en que el algoritmo genético realiza la búsqueda de la misma en el espacio de búsqueda. Así, el tamaño de la población puede influir positivamente en el proceso de búsqueda, aunque todo depende de la forma del *fitness landscape*, y a la vez puede provocar un mayor tiempo de ejecución si esta ventaja en la búsqueda no es suficiente para compensar el mayor tiempo requerido para evaluar cada generación, elementos ambos que han sido estudiados en profundidad para diferentes algoritmos evolutivos [**?**],[12]. Pero lo que no se había estudiado hasta ahora, es que sin cambios en el algoritmo pudiera haber diferencias en el *consumo instantaneo* de potencia con un simple cambio en un parámetro del mismo.

Aunque es prematuro achacar la razón de estas diferencias a algún elemento concreto que interviene en el proceso, creemos que no debe la razón estar originada en el algoritmo, sino más bien en la arquitectura del sistema que ejecuta el mismo. Como en todas las arquitecturas basadas en ARM-INTEL encontramos el mismo comportamiento, y los sistemas operativos que gestionan los procesos son basados en Linux, es posible que las diferencias se deban a la mejor o peor utilización de la jerarquía de memoria; en particular es posible que el uso de la memoria caché sea más adecuado cuando el tamaño de la población es menor. No obstante, esperamos poder estudiar esta y otras posibles fuentes de consumo de energía en próximos trabajos.

5. Conclusiones

Este artículo presenta un primer estudio del consumo energético de un algoritmo genético simple. La idea es estudiar si la ventaja que ofrecen equipos de más altas prestaciones en cuanto al menor tiempo necesario para encontrar soluciones tiene un coste razonable en cuanto a energía consumida, o por el contrario, al tener en cuenta este elemento las ventajas de las arquitecturas de procesador más avanzadas se diluyen. Hasta ahora, esta cuestión no se había planteado en la literatura de los Algoritmos Evolutivos, y por defecto, el objetivo siempre ha sido reducir el tiempo de ejecución, sin tener en cuenta el coste energético.

Utilizando diferentes plataformas hardware, que incluyen PC y raspberry pi ejecutando Linux, Smartphone android, y SoC Xilinx (FPGA Zynq xc7z020clg484-1), se ha realizado una medición tanto de tiempo de ejecución para el mismo algoritmo como el consumo energético a lo largo de cada experimento.

Los resultados muestran que aunque los dispositivos más potentes necesitan menos tiempo para encontrar la solución al problema de optimización mediante el algoritmo genético, cuando tenemos en cuenta el consumo energético, las ventajas se diluyen. En primer lugar hemos visto que la FPGA es el dispositivo menos interesante en las condiciones estudiadas: cuando un algoritmo ya disponible en un lenguaje de alto nivel, se traslada al hardware utilizando las herramientas que la propia tarjeta suministra. Centrándonos en el resto de equipos, hemos comprobado que aunque el PC es el equipo más rápido, es también el que globalmente consume más energía cuando el algoritmo genético se está ejecutando. Pero además, la energía necesaria para encontrar una solución medida en Kilovatios/hora es mucho menor cuando se utilizan equipos con menores prestaciones, como la raspberry pi o el Smartphone. Esto muestra que la tendencia habitual a ejecutar los algoritmos evolutivos en equipos de altas prestaciones es la que mayor huella energética produce, y va en contra de la línea actual que promueve el respeto por el medio ambiente produciendo bienes y equipos con reducido consumo energético.

Por otro lado, hemos comprobado que algunos parámetros del algoritmo evolutivo pueden influir notablemente en el consumo energético. Incluso en el caso del algoritmo genético simple y secuencial, mediante pruebas experimentales hemos constatado que la modificación del tamaño de población produce un cambio en el consumo instantáneo de energía. Aunque es prematuro indicar el origen de ese cambio, y esperamos en próximos trabajos hacer un análisis riguroso, creemos que el tamaño de la población puede influir en el mejor o peor uso de la jerarquía de memoria del sistema, y por ello en la energía necesaria para ejecutar cada paso del algoritmo. Aunque esto por supuesto puede depender de la estructura de datos utilizada por implementar la población en cada herramienta, y convendrá comparar diferentes herramientas de Algoritmos Evolutivos.

6. Agradecimientos

Este trabajo se ha realizado con el apoyo del 7º Programa Marco de la Unión Europea, mediante el proyecto FP7-PEOPLE-2013-IRSES, ACoBSEC 612,689

Grant; del Ministerio de Educación Cultura y Deporte, proyecto UEX:EPHEMEC (TIN2014-56494-C4-2-P), y Gobierno de Extremadura, Consejería de Economía Comercio e Innovación and FEDER, proyecto GRU10029.

Referencias

1. Hooper, A. (2008). Green computing. Communication of the ACM, 51(10), 11-13.
2. Fong, K. F., Hanby, V. I., Chow, T. T. (2006). HVAC system optimization for energy management by evolutionary programming. Energy and Buildings, 38(3), 220-231.
3. Lee, W. S., Chen, Y. T., Kao, Y. (2011). Optimal chiller loading by differential evolution algorithm for reducing energy consumption. Energy and Buildings, 43(2), 599-604.
4. Fadaee, M., Radzi, M. A. M. (2012). Multi-objective optimization of a stand-alone hybrid renewable energy system by using evolutionary algorithms: A review. Renewable and Sustainable Energy Reviews, 16(5), 3364-3369.
5. Cotta, C. Fernández, A. J., Fernández de Vega, F., Chávez, F., Merelo, J.J.,Castillo, P.A., Camacho, D., Bello, G: Ephemeral Computing and Bioinspired Optimization. IJCCI (ECTA) 2015: 319-324
6. Yoo, C. M. K. S. (2011). Energy-Aware System Design. Springer.
7. Díaz, J., Risco, J.L., Colmenar, J.M., Multi-objective optimization of energy consumption and execution time in a single level cache memory for embedded systems.
8. Bansal, N., Kimbrel, T., Pruhs, K. (2004, October). Dynamic speed scaling to manage energy and temperature. In Foundations of Computer Science, 2004. Proceedings. 45th Annual IEEE Symposium on (pp. 520-529). IEEE.
9. Albers, S. (2011). Algorithms for dynamic speed scaling. In LIPIcs-Leibniz International Proceedings in Informatics (Vol. 9). Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik.
10. Albers, S. (2010). Energy-efficient algorithms. Communications of the ACM, 53(5), 86-96.
11. Zbigniew Michalewicz,(1996). Genetic Algorithms + Data Structures = Evolution Programs. Third Edition, Springer. ISBN: 3-540-60676-9.
12. Fernández, F., Tomassini, M., Vanneschi, L. (2003). An empirical study of multipopulation genetic programming. Genetic Programming and Evolvable Machines, 4(1), 21-51.
13. Dennis Cormier and Sita Raghavan. http://people.sc.fsu.edu/ jburkardt/cpp_src/ simple_ga/simple_ga.html.
14. Papadimitriou, C. H. (2003). Computational complexity (pp. 260-265). John Wiley and Sons Ltd..
15. Fernández de Vega, F., Pérez, J. I. H. (2012). Parallel Architectures and Bioinspired Algorithms (Vol. 122). Springer.
16. Gordon, M., Zhang, L., Tiwana, B., Dick, R., Mao, Z. M., Yang, L. (2013). PowerTutor: a power monitor for android-based mobile platforms.
17. Zhang, L., Tiwana, B., Qian, Z., Wang, Z., Dick, R. P., Mao, Z. M., Yang, L. (2010, October). Accurate online power estimation and automatic battery behavior based power model generation for smartphones. In Proceedings of the eighth IEEE/ACM/IFIP international conference on Hardware/software codesign and system synthesis (pp. 105-114). ACM.