Dávila-Guzmán, Maria A.; Alfonso-Morales, Wilfredo; Caicedo-Bravo, Eduardo F.
Arquitectura heterogénea para el procesamiento de los algoritmos de enjambres
Tecno Lógicas, vol. 17, núm. 32, enero-junio, 2014, pp. 11-20
Instituto Tecnológico Metropolitano
Medellín, Colombia

Tecno Lógicas,
ISSN (Versión impresa): 0123-7799
tecnologicas@itm.edu.co
Instituto Tecnológico Metropolitano
Colombia

Tecno Lógicas
ISSN 0123-7799
Vol. 17, No. 32, pp. 11-20
Enero-junio de 2014

Arquitectura heterogénea para el procesamiento de los algoritmos de enjambres

Heterogeneous architecture to process swarm optimization algorithms

Maria A. Dávila-Guzmán[1], Wilfredo Alfonso-Morales[2], Eduardo F. Caicedo-Bravo[3]

Recibido: 07 de marzo de 2013,
Aceptado: 26 de noviembre de 2013

Como citar / How to cite
M. A. Dávila-Guzmán, W. Alfonso-Morales y E. F. Caicedo-Bravo, "Arquitectura heterogénea para el procesamiento de los algoritmos de enjambres", *Tecno Lógicas*, vol. 17, no. 32, pp. 11-20, 2014.

1 M.Sc en ingeniería con énfasis en electrónica, Grupo de investigación en percepción y sistemas inteligentes PSI, Universidad del Valle, Cali-Colombia, maria.a.davila@correounivalle.edu.co
2 M.Sc en ingeniería con énfasis en electrónica, Grupo de investigación en percepción y sistemas inteligentes PSI, Universidad del Valle, Cali-Colombia, wilfredo.alfonso@correounivalle.edu.co
3 Ph.D Informática Industrial, Grupo de investigación en percepción y sistemas inteligentes PSI, Universidad del Valle, Cali-Colombia, eduardo.caicedo@correounivalle.edu.co

Resumen

Desde años recientes, el paralelismo hace parte de la arquitectura de las computadoras personales al incluir unidades de co-procesamiento como las unidades de procesamiento gráfico, para conformar así una arquitectura heterogénea. Este artículo presenta la implementación de algoritmos de enjambres sobre esta arquitectura para resolver problemas de optimización de funciones, destacando su estructura inherentemente paralela y sus propiedades de control distribuido. En estos algoritmos se paralelizan los individuos de la población y las dimensiones del problema gracias a la granuralidad del sistema de procesamiento, que además proporciona una baja latencia de comunicaciones entre los individuos debido al procesamiento embebido. Para evaluar las potencialidades de los algoritmos de enjambres sobre la plataforma heterogénea, son implementados dos de ellos: el algoritmo de enjambre de partículas y el algoritmo de enjambre de bacterias. Se utiliza la aceleración como métrica para contrastar los algoritmos en la arquitectura heterogénea compuesta por una GPU NVIDIA GTX480 y una unidad de procesamiento secuencial, donde el algoritmo de enjambre de partículas obtiene una aceleración de hasta 36,82x y el algoritmo de enjambre de bacterias logra una aceleración de hasta 9,26x. Además, se evalúa el efecto al incrementar el tamaño en las poblaciones donde la aceleración es significativamente diferenciable pero con riesgos en la calidad de las soluciones.

Palabras clave

Algoritmos de enjambre, algoritmo de enjambre de bacterias, algoritmo de enjambre de partículas, GPU, paralelo.

Abstract

Since few years ago, the parallel processing has been embedded in personal computers by including co-processing units as the graphics processing units resulting in a heterogeneous platform. This paper presents the implementation of swarm algorithms on this platform to solve several functions from optimization problems, where they highlight their inherent parallel processing and distributed control features. In the swarm algorithms, each individual and dimension problem are parallelized by the granularity of the processing system which also offer low communication latency between individuals through the embedded processing. To evaluate the potential of swarm algorithms on graphics processing units we have implemented two of them: the particle swarm optimization algorithm and the bacterial foraging optimization algorithm. The algorithms' performance is measured using the acceleration where they are contrasted between a typical sequential processing platform and the NVIDIA GeForce GTX480 heterogeneous platform; the results show that the particle swarm algorithm obtained up to 36.82x and the bacterial foraging swarm algorithm obtained up to 9.26x. Finally, the effect to increase the size of the population is evaluated where we show both the dispersion and the quality of the solutions are decreased despite of high acceleration performance since the initial distribution of the individuals can converge to local optimal solution.

Keywords

Swarm intelligence algorithm, bacterial foraging optimization, particle swarm optimization, and graphic processing unit.

1. INTRODUCCIÓN

Los sistemas de cómputo convencional han evolucionado a sistemas de procesamiento paralelo para realizar operaciones de cómputo intensivo y mejorar la velocidad de procesamiento. En las computadoras personales el paralelismo hace parte de la arquitectura incluyendo instrucciones vectoriales (SIMD), procesadores multinúcleo y unidades de co-procesamiento como las *many-core* y las FPGAs [2]; este conjunto es el que recibe el nombre de arquitectura heterogénea. Cada uno de los procesadores de la arquitectura heterogénea puede manejar una carga de cómputo diferente, que distribuida de forma adecuada, puede ejecutar las aplicaciones de forma eficiente.

Estas nuevas capacidades de procesamiento permiten eliminar parte de incompatibilidad de los algoritmos enjambre con el cómputo convencional, donde, las características heredadas de la computación natural son modeladas de forma cualitativa y no cuantitativa sobre ordenadores clásicos como se muestra en la Tabla 1.

Tabla 1. Computación convencional versus computación natural. Fuente: [1]

Convencional	Natural
Determinístico	Estocástico
Síncrono	Asíncrono
Serial	Paralelo
Susceptible	Robusto
Intolerante a fallas	Tolerante a fallos
Humano-dependiente	Autónomo
Centralizado	Distribuido
Preciso	Aproximado
Simple	Complejo

Por ello, se ha generado una tendencia creciente al cambio de tecnología de procesamiento, enfocándose principalmente en las características paralelas y de control distribuido de estos algoritmos.

Para exaltar el paralelismo de los algoritmos de enjambres se ha optado por plataformas *many-core* como las unidades de procesamiento gráfico (GPU), FPGAs y clusters de computadoras. En estos sistemas se tiene mayor capacidad de procesamiento y se busca obtener una mayor velocidad en las rutinas a procesar.

Las GPUs se han convertido en una plataforma atractiva para el procesamiento por su comparativo bajo costo. Sobre ésta se ha evaluado el algoritmo de enjambre de partículas [3]-[9] y el algoritmo de hormigas [10]-[15]. Sobre un clúster de computadoras y FPGA se ha implementado el algoritmo de partículas [16], y solo sobre FPGAs variantes los algoritmos de hormigas [17], [18]. Estas implementaciones demuestran aceleración y, a pesar de ser parte de una misma familia de algoritmos, las implementaciones paralelas de cada variante son presentadas de forma aislada. Cuando se tiene mayor capacidad de procesamiento, en el caso de PSO, el aumento desmedido de los parámetros de población es presentado en términos de la aceleración, sin tomar en cuenta las capacidades del algoritmo para dar solución al problema de optimización.

Este artículo presenta un planteamiento de los algoritmos de enjambres, donde se busca maximizar las prestaciones de procesamiento haciendo uso de una plataforma paralela, realizando un análisis de las arquitecturas más usadas para destacar las características inherentes de estos algoritmos.

Usando una arquitectura heterogénea conformada por una CPU que aloja una GPU, son implementados el algoritmo PSO, uno de los más populares y el algoritmo BFO reconocido por la calidad de las soluciones en problemas de aplicación y alta carga computacional [19], [20]. Se evalúa el rendimiento que produce el cambio de tecnología, determinado por las ganancias en tiempo de procesamiento de estos algoritmos usando como métrica la aceleración entre los sistemas CPU-GPU y la CPU. Se analizan las variaciones de la aceleración con la cantidad de dimensiones del problema y el tamaño de la población y así, mostrar las ventajas con sistemas de bajo costo como las arquitecturas heterogéneas.

2. ALGORITMOS DE ENJAMBRES

Entre los animales sociales se pueden distinguir las bandadas de aves, los cardúmenes, e insectos como las hormigas, abejas, avispas, termitas, entre otros. Estos animales resuelven problemas de su vida diaria como la consecución de comida, construcción de nidos, división de labores y más, que tienen su equivalente en problemas científicos y de ingeniería [21]. Individuos con poca inteligencia y capacidades limitadas y un bajo nivel de comunicación con sus vecinos, exhiben un comportamiento colectivo inteligente y auto-organizado de carácter global donde típicamente no se sigue a un líder, o un plan global [22], [23].

La inteligencia de enjambres toma las características de estos animales sociales como inspiración para resolver problemas, donde surgen los algoritmos de enjambres. Comparados con las técnicas tradicionales para la resolución de problemas de optimización, proporcionan mejoras en la robustez y efectividad [22]. Se caracterizan por las estrategias de intensificación y diversificación, resumidas en la Tabla 2, que permite al algoritmo encontrar soluciones de alta calidad y moverse por áreas sin exploración del espacio de búsqueda.

Tabla 2. Mecanismo de intensificación y diversificación de los principales algoritmos de enjambres. Fuente: Autores.

Metaheurística	Componente de intensificación y diversificación
ACO (*Ant Colony Optimization*)	Rastro de feromona. Construcción probabilística.
PSO (*Particle Swarm Optimization*)	Parámetro cognitivo. Parámetro social.
BFO (*Bacteria Foraging Optimization*)	Evento reproductivo. Evento de eliminación/dispersión.
Algoritmos de abejas	Comunicación por danza. División de labores. Mecanismo de abandono.
Algoritmo de luciérnagas	Función de atracción.

Una descripción general de alto nivel de los algoritmos de enjambres se presenta en el Algoritmo 1. Basados en este esquema, se plantean los algoritmos como capas de niveles iterativos como se muestra en la Fig. 1. En la capa interna, cada individuo es procesado de forma independiente, cuya información es compartida entre todos los individuos. El mecanismo de comunicación produce la emergencia de un comportamiento colectivo que conforma la capa del enjambre.

El enjambre puede estar conformado por todos los individuos o un subconjunto de estos que, finalmente, está sometido a una condición de parada que puede ser el trascurso del tiempo, un máximo error permitido para la solución, un número de iteraciones fijo o cualquier otra estrategia que emule el comportamiento natural y permita soluciones óptimas.

Algoritmo 1. Descripción de alto nivel para los algoritmos de enjambres

```
for Cada individuo del enjambre do
    generar la población inicial
end for
repeat
    for cada individuo del enjambre do
        generar solución local
        selección de mejor solución local {opcional}
    end for
    comunicación entre individuos
    Selección de la mejor solución global
until condición de parada
return Mejor solución encontrada
```

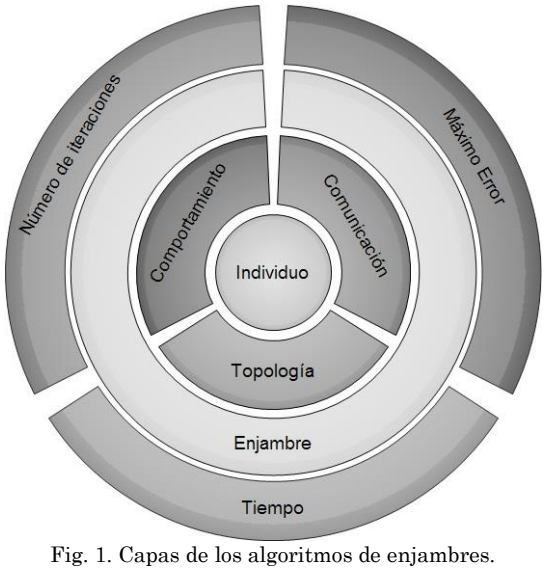

Fig. 1. Capas de los algoritmos de enjambres. Fuente: Autores

Tal como se observa en la Fig. 1, en cada nivel se tienen componentes que se pueden procesar de forma independiente;

sin embargo, los algoritmos de enjambres han sido usualmente implementados como simulaciones seriales de procesos paralelos [24].

3. PLATAFORMAS PARALELAS

El factor económico es una de las referencias a la hora de proponer plataformas para el desarrollo e investigación. El costo de un equipo de cómputo además de su precio en el mercado, se debe establecer por las prestaciones que se pueden obtener o por la cantidad de usuarios que pueden acceder a este.

Para mejorar los tiempos de procesamiento en los algoritmos se puede optar por la implementación en plataformas con mayor potencia de cálculo como las supercomputadoras o *mainframe*, que permiten seguir el esquema clásico de implementación de los algoritmos, pero los costos de estos equipos son elevados, lo que dificulta su acceso en ciertas comunidades científicas.

En la taxonomía Flynn los sistemas de procesamiento paralelo son clasificados en MIMD (*Multiple instruction, multiple data*) y SIMD (*Single instruction, multiple data*). Entre las plataformas paralelas más comunes se encuentran las arquitecturas con memoria distribuida como el cómputo en malla y en la nube, los sistemas multinúcleo, *many-core* y los procesadores sistólicos.

La computación en la nube se presenta como un sucesor de los sistemas distribuidos, donde, de forma virtual los recursos de cómputo son compartidos a través de la web. Es considerada económica para el procesamiento, aunque tener acceso a este sistema podría acarrear problemas de seguridad, además que se está expuesto a la latencia de la red de datos y la heterogeneidad de los sistemas, introduciendo en los procesos componentes de desempeño adicional difíciles de predecir. Por ello, se opta por las mallas de cómputo, que también comparten recursos a través de la red, que en la realidad se encuentran en el mismo sitio para obtener la máxima velocidad de comunicación (*clúster*).

Por otra parte, cuando se tiene un sistema integrado, las latencias de las trasferencias internas y la sincronización de los datos son, en general, pequeñas fracciones de tiempo, que pueden ser despreciadas dependiendo de la estructura del proceso. En el campo de los dispositivos integrados para el procesamiento paralelo se pueden mencionar los procesadores multinúcleo, las FPGAs (*Field Programmable Gates Arrays*) y los procesadores *many-core*.

Los procesadores multinúcleo son representados por procesadores de propósito general, cada núcleo ejecuta las instrucciones de forma independiente y los recursos son administrados por medio de un sistema operativo. Son dispositivos que se encuentran incluidos en las CPUs, que al ser comerciales y de uso general son asequibles con mayor facilidad [25]. Usan gran parte de los transistores y la potencia en operaciones no-computacionales como la lógica de control y caché, es por esto que se ha optado por combinar los sistemas de procesamiento introduciendo núcleos de aceleración para maximizar el desempeño. Estos co-procesadores pueden ser plataformas como los procesadores *many-core.*

Los procesadores *many-core,* representados por las unidades de procesamiento gráfico (GPU), han sido diseñadas para el tratamiento de imágenes, con especial interés en el mercado de los vídeo juegos y adaptadas para el procesamiento de propósito general GPGPU (General-Purpose computing on Graphics Processing Unit). Por su composición de cientos de núcleos elementales y la competencia de las GPUs en el mercado dado su comparativo bajo costo y disponibilidad en las computadoras personales las convierten en una alternativa atractiva para el procesamiento de datos en paralelo. Cada núcleo permite la ejecución de cientos de hilos de procesamiento, presentan un control de flujo me-

nor que las plataformas multinúcleo enfocando más área de transistores en las unidades de procesamiento.

Por último, las FPGAs permiten reconfigurar los recursos de procesamiento y su programación, realizada por medio de una CPU, es de propósito específico siendo versátil y usada principalmente para el prototipado de dispositivo. Además, es eficiente en potencia y velocidad de procesamiento [26].

Realizar una comparación de desempeño y capacidades entre las arquitecturas expuestas puede ser difícil. Desde el punto de vista de dispositivos integrados, las GPUs son vistas como rivales de las FPGAs y estudios comparativos muestran que el precio de las FPGAs es más elevado y con desempeños en tiempo, en algunos casos, inferior al demostrado por la GPU [27]. En cuanto a las arquitecturas multinúcleo actualmente son vistas como complementarias con las GPUs y FPGAs más que competidores [25].

4. IMPLEMENTACIÓN SOBRE LA PLATAFORMA HETEROGÉNEA

Tener un mejor desempeño de los algoritmos puede referirse al costo computacional que implica la velocidad, el consumo de hardware y el precio del mismo, o el desempeño propio del algoritmo reflejado en la calidad de la solución. En esta búsqueda, los algoritmos de enjambres se presentan como un candidato natural para mejorar las prestaciones en tiempo de procesamiento y una alternativa para lograrlo es implementarlo sobre una plataforma de procesamiento paralela.

Los algoritmos de enjambres en su capa externa (ver Fig. 1) son enteramente secuenciales, por tanto, el paralelismo en esta etapa es irrelevante. Pero en la capa interna están compuestos por un conjunto de individuos simples que cooperan para hallar una solución. Estos individuos requieren de una alta comunicación entre los procesos para conservar su estructura original. En consecuencia, para logar un buen desempeño sobre una plataforma paralela se requiere una baja latencia de comunicación, como la suministrada por plataformas totalmente embebidas.

La estructura de individuos de los algoritmos de enjambre permite pensar en el paralelismo masivo que se puede lograr sobre unidades como las GPUs, que pueden procesar de forma simultánea el núcleo del algoritmo representado por los individuos. En cuanto a la arquitectura heterogénea, debido a que las transferencias en la memoria interna de la GPU comparadas con las realizadas a la CPU son hasta 28 veces más rápidas, se evitan estas transferencias, lo que mejora la aceleración de los algoritmo [5], [6].

Gracias a la granularidad que proporciona la GPU con arquitectura CUDA [28]. La paralelización es realizada a nivel de la capa del individuo, donde la unidad mínima de procesamiento del sistema o hilo se encarga de cada una de las n dimensiones del problema y el individuo está conformado por un bloque de procesamiento compuesto por n hilos. El límite de los bloques de procesamiento está dado por dos factores: primero, la asignación de memoria, donde la memoria compartida genera mayores prestaciones por la capacidad y el ancho de banda; y segundo, el manejador de instrucciones, que define la cantidad de hilos por bloque.

Para evaluar algunas de las características de los algoritmos de enjambres, usamos una plataforma de procesamiento GPU NVIDIA GTX480 compuesta por 480 núcleos con arquitectura CUDA, donde se implementan el algoritmo de enjambre de partículas (PSO) [29] y el algoritmo de enjambre de bacterias (BFO) [30] totalmente embebido sobre la GPU. Los resultados son obtenidos de 6 problemas de minimización numérica sin restricciones mostradas en la Tabla 3 del benchmark CEC 2005 [31]. Dada la baja dispersión de los tiempos medidos en una etapa de expe-

rimentación inicial, con un máximo de 1,13%, para variaciones del parámetro de dimensiones y poblaciones con 50 muestras, para el experimento donde se varían las dimensiones se obtienen de una única muestra; y en el segundo experimento donde se analiza la variación de la población, se realiza con base a 25 muestras.

Tabla 3. Funciones del CEC2005 para evaluar los algoritmos. Fuente: Autores

Función	F_{bias}	Rango
$F_1 = \sum_{i=1}^{n} Zi_i^2 + F_{bias}$	-450	[-100, 100]
$F_2 = \sum_{i=1}^{n} Z_i \sin\sqrt{\lvert Z_i \rvert} + F_{bias}$	n*418,98	[-500, 500]
$F_3 = \sum_{i=1}^{n} (10^6)^{\frac{i-1}{n-1}} Z_i^2 + F_{bias}$	-450	[-100, 100]
$F_4 = \sum_{i=1}^{n-1} 100(Zi_i^2 - Z_{i+1})^2 + (Z_i - 1)^2 + F_{bias}$	390	[-100, 100]
$F_5 = \sum_{i=1}^{n} Zi_i^2 - 10\cos(2\pi Z_i) + 10 + F_{bias}$	-330	[-5, 5]
$F_6 = 20\exp\left(-0{,}2\sqrt{\frac{1}{n}\sum_{i=1}^{n} Zi_i^2}\right) - exp\left(\frac{1}{n}\sum_{i=1}^{n}\cos(2\pi Z_i)\right) + 20 + e + F_{bias}$	-140	[-32, 32]

Para obtener la aceleración es implementado el equivalente secuencial de los algoritmos en lenguaje C sobre la CPU con procesador Intel Xeon X5660, memoria RAM de 12GB y sistema operativo Linux Ubuntu 10.04, donde reside la GPU. Para el algoritmo PSO se configuran los parámetros con peso inercial *w* y factores cognitivo c1 y social c2 constantes (w=0,729, c1=c2=1,496) [32], 20 partículas en un intervalo de 3 a 1024 dimensiones y 100000 iteraciones, donde se obtienen las aceleraciones de la Fig. 2.

El algoritmo BFO es configurado con los parámetros dados por [30]. Se define el movimiento de las bacterias con unas componentes: atractivas wa y da, repelentes wr y hr, el tamaño del paso c y la probabilidad de eliminación-dispersión P (wa=0,2, wr=10,0, da=hr=0,1, c=0,1 y P=0,25). Este requiere más recursos de cómputo que el algoritmo PSO, por ello el intervalo de evaluación está entre 3 y 544 dimensiones con un número de pasos reproductivos Nre=12, numero de eventos de eliminación dispersión Ned=10, número de eventos quimiotacticos Nc=300 y número de pasos quimiotacticos Ns=12. La aceleración para este algoritmo se muestra en la Fig. 3.

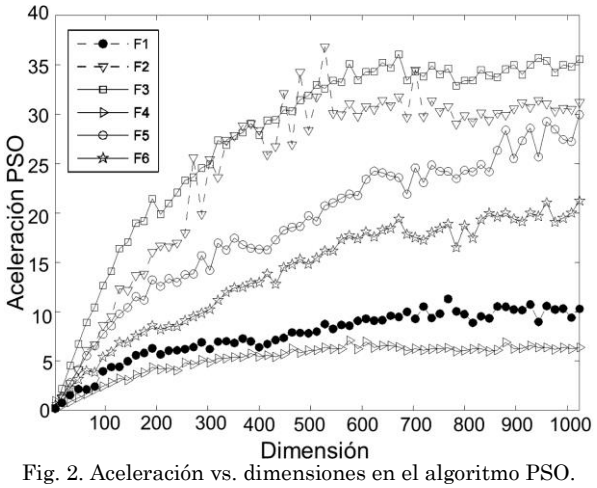

Fig. 2. Aceleración vs. dimensiones en el algoritmo PSO. Fuente: Autores

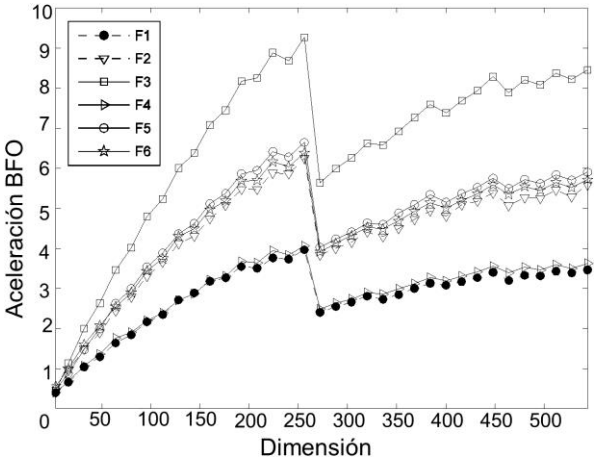

Fig. 3. Aceleración vs. dimensiones en el algoritmo BFO. Fuente: Autores

Tal como se observa en las Fig. 2 y 3, por debajo de las 30 dimensiones el algoritmo secuencial es superior a la implementación sobre la GPU. Sin embargo, una vez el número de dimensiones supera este umbral, la GPU tiene una mayor ocupación de recursos de cómputo haciéndolo superior hasta en 36,82x para el algoritmo PSO y 9,26x para el algoritmo BFO. En cuanto a la calidad de la solución, los resultados son

equivalentes al de la implementación secuencial porque se conserva la estructura original del algoritmo.

Hasta el momento se ha demostrado el comportamiento de la velocidad del algoritmo variando las dimensiones del problema de minimización, pero factores como la cantidad de individuos que conforman la población, afectan la carga computacional del algoritmo y generan comportamientos diferentes en los resultados de la optimización, pero en general no son evaluados sobre las CPUs clásicas por la carga que ello implica.

En términos biológicos, mantener la diversidad (variedad y abundancia) de la población se asocia directamente a la capacidad de exploración, que significa encontrar nuevas soluciones en el espacio de búsqueda, benéfico para evitar la convergencia prematura. Debido a las limitaciones de memoria de las plataformas de procesamiento y la reducción de la velocidad por las múltiples evaluaciones de la función objetivo, incrementar el número de agentes en los algoritmos de enjambres es descartado.

Una de las preocupaciones al tener un enjambre de gran tamaño es el tiempo de procesamiento que se requiere, ya que por cada individuo adicional, se incrementa en aproximadamente $I \times P$ veces el tiempo de procesamiento, donde I representa la cantidad de iteraciones y P el tiempo total de las instrucciones que ejecuta cada individuo. Esta preocupación en tiempo deja de ser crítica en las plataformas paralelas, dado que cada individuo representará una unidad de procesamiento que, teóricamente, se ejecuta de forma simultánea como enjambre.

De forma experimental se determina el comportamiento de los algoritmos con el incremento de la población en 20, 80, 140 y 200 individuos y n=30. Se realizan 25 muestras por cada función para PSO y BFO. En la Fig. 4 se presentan los resultados del tiempo de procesamiento para el algoritmo PSO con 1000 iteraciones y BFO con Nre=8, Ned=4, Nc=200 y Ns=8.

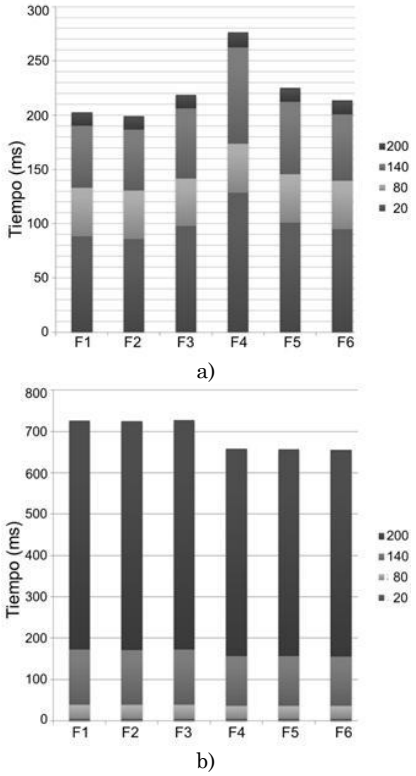

Fig. 4. Tiempo promedio de procesamiento para n=30 y variaciones de la población: a) Algoritmo PSO. b) Algoritmo BFO. Fuente: Autores

Por otra parte, demasiados agentes puede hacer el sistema poco robusto al ruido y producirse el sobreajuste (*overfitting*). Para el algoritmo PSO un tamaño de población mayor produce un desequilibrio en el algoritmo generando mayor dispersión en algunos casos y soluciones alejadas del óptimo en otros, requiriendo más iteraciones [33]; en el caso de la función F1, en el diagrama de cajas de la Fig. 5 se presentan los dos casos: mayor dispersión en la solución y soluciones sub-óptimas con 1000 iteraciones.

En cambio, para el algoritmo BFO con el incremento del tamaño de la población se obtienen mejoras en la calidad de la solución y en la dispersión de los resultados, los resultados obtenidos para F1 son presentados en el diagrama de cajas de la Fig. 6.

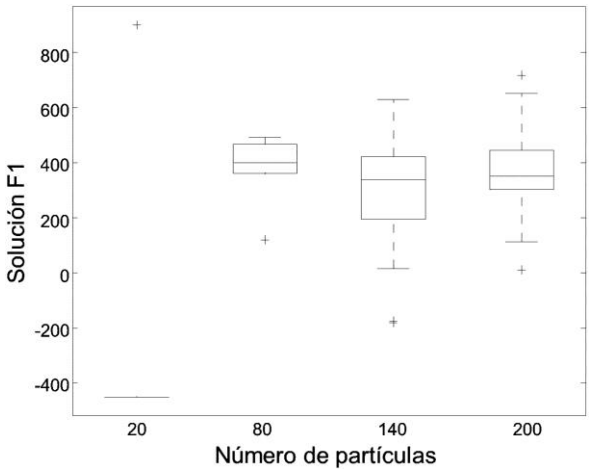

Fig. 5. Diagrama de cajas de la solución de F1 con el algoritmo PSO, n=30 y 1000 iteraciones. Fuente: Autores

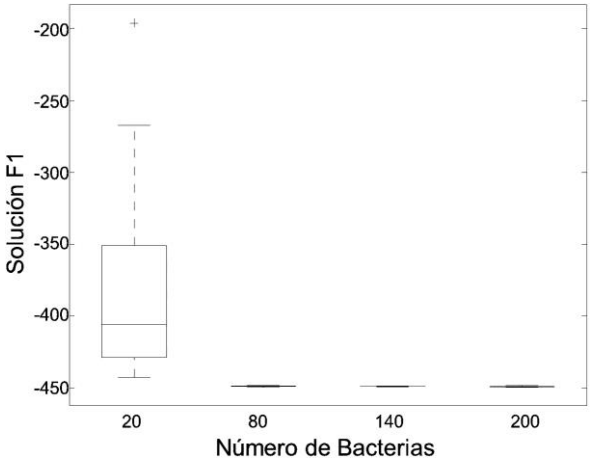

Fig. 6. Diagrama de cajas de la solución de F1 con el algoritmo BFO, n=30; Nre=8, Ned=4, Nc=200, Ns=8. Fuente: Autores

5. CONCLUSIONES

Los algoritmos de enjambres con procedimientos de intensificación y diversificación implican operaciones complejas y/o matriciales, por esta razón su implementación sobre plataformas paralelas -como las que se encuentran en arquitecturas heterogéneas- se ve favorecida. Esto permite extender su aplicación en áreas donde el desempeño en tiempo es un factor determinante para su aplicación en problemas de optimización. Además, se puede evaluar el sistema en mayor cantidad de iteraciones, que en la mayoría de los algoritmos de enjambres permite explorar mucho más el espacio de búsqueda y encontrar soluciones de mayor calidad, minimizando los impactos del tiempo de procesamiento.

Dada la estructura simple que conforma cada individuo en estos algoritmos, el uso de procesadores elementales generan un mejor desempeño en cuanto a tiempo de procesamiento comparado con el equivalente secuencial como en el algoritmo PSO con una aceleración máxima de 36,82x y en el algoritmo BFO de 9,26x en una implementación realizada sobre una plataforma GPU GTX480 con arquitectura CUDA. La aceleración de las implementaciones del algoritmo PSO y BFO es significativa cuando las dimensiones del problema son superiores a 50 dimensiones. Mostrando que los beneficios de este tipo de plataformas dependen en gran medida de parámetros como las dimensiones del problema, la cantidad de individuos que conforman la población y la capacidad del hardware.

Finalmente, como trabajo futuro se propone generar estrategias y metodologías que permitan reducir los tiempos de implementación y mejorar la experiencia del programador para compartir las experiencias con la comunidad de investigación con el fin de estimular la migración de plataforma dado los buenos desempeños obtenidos y la asequibilidad de las arquitecturas heterogéneas como computadoras personales.

6. REFERENCIAS

[1] P. J. Bentley, "Systemic computation: A model of interacting systems with natural characteristics," *Int. J. Parallel, Emergent Distrib. Syst.*, vol. 22, no. 2, pp. 103-121, Apr. 2007.

[2] M. McCool, J. Reinders, and A. Robison, *Structured Parallel Programming: Patterns for Efficient Computation*. Morgan Kaufmann, 2012, p. 406.

[3] D. L. Souza, G. D. Monteiro, T. C. Martins, V. A. Dmitriev, and O. N. Teixeira, "PSO-GPU: accelerating particle swarm optimization in CUDA-based graphics processing units," in *Proceedings of the 13th annual conference companion on Genetic and evolutionary computation*, 2011, pp. 837-838.

[4] M. Cardenas-Montes, M. a. Vega-Rodriguez, J. J. Rodriguez-Vazquez, and A. Gomez-Iglesias, "Accelerating Particle Swarm Algorithm with

GPGPU," in *2011 19th International Euromicro Conference on Parallel, Distributed and Network-Based Processing*, 2011, pp. 560-564.

[5] IBISlab, "CUDA-PSO 2.0." Intelligent Bio-Inspired Systems Laboratory, Universita' degli Studi di Parma, Italy, p. 67, 2011.

[6] L. Mussi, F. Daolio, and S. Cagnoni, "Evaluation of parallel particle swarm optimization algorithms within the CUDA™ architecture," *Inf. Sci. (Ny).*, vol. 181, no. 20, pp. 4642-4657, Oct. 2011.

[7] L. D. P. Veronese and R. A. Krohling, "Swarm's flight: Accelerating the particles using C-CUDA," in *2009 IEEE Congress on Evolutionary Computation*, 2009, pp. 3264-3270.

[8] Y. Zhou and Y. Tan, "GPU-based parallel particle swarm optimization," in *2009 IEEE Congress on Evolutionary Computation*, 2009, no. 1, pp. 1493-1500.

[9] W. Zhu and J. Curry, "Particle Swarm with graphics hardware acceleration and local pattern search on bound constrained problems," in *2009 IEEE Swarm Intelligence Symposium*, 2009, no. 0737173, pp. 1-8.

[10] H. Bai, D. OuYang, X. Li, L. He, and H. Yu, "MAX-MIN Ant System on GPU with CUDA," in *2009 Fourth International Conference on Innovative Computing, Information and Control (ICICIC)*, 2009, pp. 801-804.

[11] J. Fu, L. Lei, and G. Zhou, "A parallel Ant Colony Optimization algorithm with GPU-acceleration based on All-In-Roulette selection," in *Third International Workshop on Advanced Computational Intelligence*, 2010, no. 2, pp. 260-264.

[12] W. Jiening, D. Jiankang, and Z. Chunfeng, "Implementation of Ant Colony Algorithm Based on GPU," in *2009 Sixth International Conference on Computer Graphics, Imaging and Visualization*, 2009, pp. 50-53.

[13] Y. You, "Parallel Ant System for Traveling Salesman Problem on GPUs," in *GECCO 2009-Genetic and Evolutionary Computation*, 2009, pp. 1-2.

[14] J. M. Cecilia, J. M. García, A. Nisbet, M. Amos, and M. Ujaldón, "Enhancing data parallelism for Ant Colony Optimization on GPUs," *J. Parallel Distrib. Comput.*, vol. 73, no. 1, pp. 42-51, Jan. 2013.

[15] A. Delévacq, P. Delisle, M. Gravel, and M. Krajecki, "Parallel Ant Colony Optimization on Graphics Processing Units," *J. Parallel Distrib. Comput.*, vol. 73, no. 1, pp. 52-61, Jan. 2013.

[16] S.-A. Li, C.-C. Hsu, C.-C. Wong, and C.-J. Yu, "Hardware/software co-design for particle swarm optimization algorithm," *Inf. Sci. (Ny).*, vol. 181, no. 20, pp. 4582-4596, Oct. 2011.

[17] B. Scheuermann, K. So, M. Guntsch, M. Middendorf, O. Diessel, H. ElGindy, and H. Schmeck, "FPGA implementation of population-based ant colony optimization," *Appl. Soft Comput.*, vol. 4, no. 3, pp. 303-322, Aug. 2004.

[18] M. A. Dávila, W. Alfonso, and E. Caicedo, "Implementación de un algoritmo basado en colonia de hormigas usando el software DSPBuilder," B.Sc Thesis, Ing. Electrónica, Universidad del Valle, Cali-Colombia, 2010.

[19] B. Niu, Y. Fan, L. Tan, J. Rao, and L. Li, "A Review of Bacterial Foraging Optimization Part I: Background and Development," *Adv. Intell. Comput. Theor. Appl.*, vol. 93, pp. 535-543, 2010.

[20] B. Niu, Y. Fan, L. Tan, J. Rao, and L. Li, "A Review of Bacterial Foraging Optimization Part II: Applications and Challenges," *Adv. Intell. Comput. Theor. Appl.*, vol. 93, pp. 544-550, 2010.

[21] A. Abraham, H. Guo, and H. Liu, "Swarm Intelligence: Foundations Perspectives and Applications," in *Swarm Intelligent Systems*, N. Nedjah and L. de Macedo, Eds. Springer Berlin Heidelberg, 2006, pp. 3-25.

[22] E. Bonabeau, G. Theraulaz, and M. Dorigo, *Swarm Intelligence: From Natural to Artificial Systems*. Oxford University Press, USA, 1999, p. 307.

[23] Y. F. Zhu and X. M. Tang, "Overview of swarm intelligence," in *2010 International Conference on Computer Application and System Modeling (ICCASM 2010)*, 2010, pp. V9-400-V9-403.

[24] M. Rouhipour, P. J. Bentley, and H. Shayani, "Fast bio-inspired computation using a GPU-based systemic computer," *Parallel Comput.*, vol. 36, no. 10-11, pp. 591-617, Oct. 2010.

[25] F. J. Esteban, D. Díaz, P. Hernández, J. a. Caballero, G. Dorado, and S. Gálvez, "Direct approaches to exploit many-core architecture in bioinformatics," *Futur. Gener. Comput. Syst.*, vol. 29, no. 1, pp. 15-26, Jan. 2013.

[26] A. Brodtkorb, C. Dyken, and T. Hagen, "State-of-the-art in heterogeneous computing," *Sci. Program.*, vol. 18, no. 1, pp. 1-33, 2010.

[27] D. H. Jones, A. Powell, C.-S. Bouganis, and P. Y. K. Cheung, "GPU Versus FPGA for High Productivity Computing," in *2010 International Conference on Field Programmable Logic and Applications*, 2010, pp. 119-124.

[28] NVIDIA, "CUDA C, Programming Guide." p. 185, 2010.

[29] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proceedings of ICNN'95 - International Conference on Neural Networks*, 1995, vol. 4, pp. 1942-1948.

[30] K. M. Passino, "Biomimicry of bacterial foraging for distributed optimization and control," *IEEE Control Syst. Mag.*, vol. 22, no. 3, pp. 52-67, Jun. 2002.

[31] P. N. Suganthan, N. Hansen, J. J. Liang, K. Deb, A. Auger, and S. Tiwari, "Problem Definitions and Evaluation Criteria for the CEC 2005 Special Session on Real-Parameter Optimization," 2005.

[32] I. C. Trelea, "The particle swarm optimization algorithm: convergence analysis and parameter selection," *Inf. Process. Lett.*, vol. 85, no. 6, pp. 317-325, Mar. 2003.

[33] Y. Noa, "Estrategias para mejorar el Balance entre Exploración y Explotación en Optimización de Enjambre de Partículas," M.Sc Thesis, Facultad de Matemática y Computación, Universidad de La Habana, La Habana-Cuba, 2011.