Propuesta de una Metodología Generalizada para Diseñar Micro Algoritmos Bioinspirados

Juan C. Herrera-Lozada[1], Hiram Calvo[2], Hind Taud[3], Edgar A. Portilla-Flores[4]

[1, 2] Centro de Investigación en Computación, CIC – IPN.
U. P. Adolfo López Mateos, Av. Juan de Dios Bátiz s/n casi esq. Miguel Othón de Mendizábal, Edificio del CIC, Col. Nueva Industrial Vallejo,
Del. Gustavo A. Madero, C.P. 07738, México D.F.
[3,4] Centro de Innovación y Desarrollo Tecnológico en Cómputo, CIDETEC IPN.
[1]jlozada@ipn.mx, [2]hiramcalvo@gmail.com, [3]htaud@ipn.mx, [4]aportilla@ipn.mx

Resumen. Este trabajo sugiere una metodología para diseñar algoritmos bioinspirados que funcionen con una población de tamaño reducido, para resolver problemas de optimización. Nuestra idea principal versa en considerar tres aspectos fundamentales para generalizar la propuesta: una convergencia nominal, un proceso de elitismo y un mecanismo de reinicialización del algoritmo. Para validar los resultados se diseñó un algoritmo de evolución diferencial con una población de sólo 5 individuos.

Palabras Clave: Algoritmos Bioinspirados, Evolución Diferencial, Micro Algoritmo, Optimización Numérica.

1 Introducción

Los algoritmos evolutivos y bioinspirados han adquirido gran importancia dentro del área de la inteligencia artificial, debido a que han demostrado ser exitosos en la solución de ciertos problemas complejos de aprendizaje de máquina, clasificación y optimización [1]. Este tipo de técnicas son poblacionales, es decir, utilizan una población de soluciones potenciales, lo que permite una exploración amplia del espacio de búsqueda. La población se genera de manera aleatoria y se somete a un proceso iterativo utilizando diferentes estrategias y operadores de variación para mejorar las soluciones.

La simplicidad de un algoritmo es una de las tendencias actuales en el área de la computación evolutiva, aunque en la mayoría de los casos se sacrifica el desempeño en aras de un menor coste computacional [2], [3]. Debido a la manipulación simultánea de un gran conjunto de soluciones, la ejecución de un algoritmo requiere un espacio suficiente en memoria de datos, además de considerar un tiempo de procesamiento generalmente elevado. Para reducir estos factores se han diseñado algoritmos con poblaciones extremadamente pequeñas y que para la mayoría de las aplicaciones abordadas se desempeñan de manera similar a los algoritmos de población estándar. No obstante que los micro algoritmos han demostrado ser

M.A Cruz-Chávez, J.C Zavala Díaz(Eds):CICos2009, ISBN:978-607-00-1970-8, pp. 189 - 199, 2009

competitivos, el tamaño reducido de su población puede obligar a una convergencia prematura derivada de una limitada exploración del espacio de búsqueda, por lo que el número de individuos, así como los mecanismos para mantener la diversidad son factores definitorios. Para demostrar cómo se utiliza la metodología generalizada aquí propuesta, de la literatura especializada se eligió el algoritmo de Evolución Diferencial (ED), el cual es muy representativo en los algoritmos bioinspirados contemporáneos y ampliamente conocido por su sencillez de implementación y su eficiencia.

1.1 Fundamentación Básica

El problema de la optimización global se establece como:
encontrar $\vec{x}$ tal que optimice $f(\vec{x})$, en donde $\vec{x} \in \Re^n$ representa el vector de soluciones $\vec{x} = [x_1, x_2, ..., x_n]^T$ y cada x_i, $i = 1,...,n$ tiene límites inferior y superior $L_i \leq x_i \leq U_i$.

1.2 Trabajo Previo

En [4], Goldberg experimentó con un Algoritmo Genético (AG) simple de representación binaria, utilizando una población de sólo 3 individuos y afirmó que éstos eran suficientes para asegurar la convergencia sin importar el tamaño del cromosoma. En su trabajo, Goldberg aplicó los operadores genéticos hasta alcanzar una convergencia nominal conceptualizada como la generación en la cual los individuos son muy similares o se alcanza cierto número predefinido de iteraciones, obteniendo un nuevo individuo (el de mejor aptitud), para posteriormente generar de manera aleatoria los otros dos individuos que completarán la nueva población.

Krishnakumar en [5] diseñó un AG con una población reducida a 5 individuos, a su algoritmo de representación binaria lo nombró Micro Algoritmo Genético (*Micro-Genetic Algorithm*). Al igual que Goldberg, Krishnakumar utilizó elitismo para preservar la mejor cadena encontrada al término de la convergencia nominal, como uno de los individuos obligatorios para la siguiente generación. Al comparar el desempeño del micro-AG contra un AG simple con una población de 50 individuos, se obtuvieron mejores resultados sobre funciones de un solo objetivo, además de que se comprobó que el AG de población reducida convergía más rápido.

Coello y Toscano, diseñaron un micro AG para resolver problemas de optimización de múltiples objetivos [6], aportando criterios para el manejo de restricciones de igualdad y desigualdad, además de proponer un esquema dominancia de pareto con un posicionamiento geográfico para mantener la diversidad y distribuir uniformemente las soluciones del frente de pareto. Este algoritmo trabaja con una población de 4 individuos y utiliza una memoria secundaria que almacena las soluciones potenciales a lo largo de la búsqueda.

Recientemente, Fuentes y Coello en [7], diseñaron un micro algoritmo PSO (*Particle Swarm Optimization – Optimización por Cúmulo de Partículas*) para resolver problemas de optimización de un solo objetivo con manejo de restricciones. Utilizan 5 partículas (individuos) y se apoyan de una convergencia nominal. El primer autor de este mismo algoritmo, en [8] realizó una extensión de su primera propuesta y diseñó un micro algoritmo PSO para resolver problemas de optimización multiobjetivo.

2 Propuesta Generalizada de un Micro Algoritmo Bioinspirado

Realizando una revisión de los trabajos anteriores, es posible encontrar similitudes en el diseño de un algoritmo con un tamaño de población reducido:

1. Población de 3 a 5 individuos.
2. Se requiere de una convergencia nominal y de un proceso de reinicialización.
3. Es necesario considerar elitismo para preservar, al menos, al mejor individuo obtenido al término de la convergencia nominal.

Otro punto en común que comparten estas realizaciones es que el micro algoritmo que se utiliza para optimización mono-objetivo puede adecuarse para manejar restricciones y para optimización con múltiples objetivos [9]. En base a lo anteriormente expuesto, presentamos el esquema general de la Figura1 para diseñar (o adaptar en su caso) un algoritmo estándar a un modelo de micro población. Se puede apreciar que existen dos ciclos de funcionamiento: uno interno que se ejecutará mientras no se haya alcanzado la convergencia nominal y uno externo que se efectuará hasta que se alcance el criterio de paro del algoritmo.

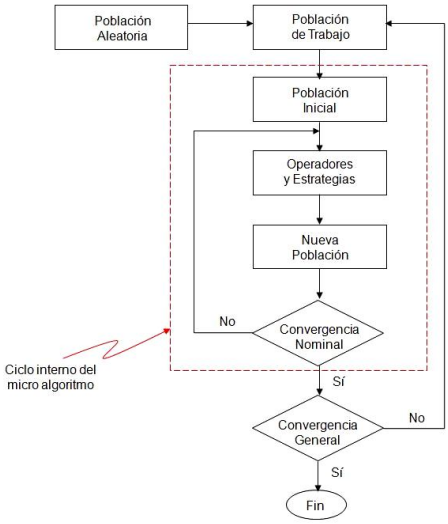

Fig. 1. Esquema general de un micro algoritmo bioinspirado.

La metodología generalizada consta de 3 partes fundamentales:

1. **Definir el número de individuos de la población.** Resulta evidente que en dependencia a la complejidad del problema a resolver, será necesario realizar ajustes en el tamaño de la micro población, no obstante, para cumplir con la metodología que proponemos el número de 5 individuos es asequible para resolver las funciones que se probaron experimentalmente. En la generación inicial, los individuos de la población se obtienen de manera aleatoria, éstos se copian a la población de trabajo y a la población inicial del ciclo interno del micro algoritmo (refiérase a la Figura 1).
   Durante el proceso de evolución, generación tras generación, la población se mantendrá con un número estático, es decir, el tamaño de la población no crece ni se disminuye de manera dinámica, aunque se debe considerar la inclusión de individuos nuevos en cada generación como se mencionará posteriormente.

2. **Definir el criterio de la convergencia nominal y aplicar los operadores y estrategias del algoritmo bioinspirado en su versión estándar.** El criterio más utilizado para la convergencia nominal es definir un número de generaciones máximas a realizar; en este rubro ejecutamos pruebas funcionales que permitieron precisar que 5 generaciones son suficientes para los experimentos con el micro algoritmo de Evolución Diferencial. Al alcanzar la convergencia nominal es necesario un proceso de reinicialización para conformar una nueva población de trabajo. En el ciclo interno controlado por

la convergencia nominal se utilizan los operadores y estrategias particulares definidas para el algoritmo bioinspirado en su versión estándar siendo posible implementarlo sin cambios o en otro caso realizar modificaciones que mejoren el desempeño del algoritmo con respecto a un problema en particular. Debido a que la calidad de las soluciones generadas al término de la convergencia nominal depende de los operadores y estrategias definidas por cada algoritmo estándar, el número de generaciones para alcanzar la convergencia nominal seguramente será diferente al cambiar del algoritmo de ED a algún otro algoritmo bioinspirado.

3. **Aplicar elitismo para garantizar la convergencia.** Algunos de los individuos obtenidos al término de la convergencia nominal, generalmente los mejores o al menos el mejor (elitismo) en base a la aptitud, deben ser copiados a la población de trabajo que se completa con individuos generados aleatoriamente (para mantener la diversidad) y se incrementa una generación más en el ciclo externo hasta que se alcance la condición de paro del algoritmo. Es significativo mencionar que depende mucho de la naturaleza de los algoritmos bioinspirados, con respecto a los operadores y estrategias internas, para determinar cuántos individuos deben ser copiados como parte del elitismo a la población de trabajo, además de que éste garantiza la convergencia del micro algoritmo. En la particularidad de esta propuesta y del algoritmo elegido para los experimentos, cuando se alcanza la convergencia nominal en el ciclo interno, se copian los 4 mejores individuos (seleccionados a través de un *ranking*) a la nueva población del ciclo externo para reinicializar todo el proceso.

3 Micro Algoritmo de Evolución Diferencial

La Evolución Diferencial es un algoritmo evolutivo propuesto por Storn y Price [10] para resolver problemas de optimización principalmente en espacios continuos y en el cual las variables se representan mediante números reales. Actualmente se ha utilizado en problemas de alta complejidad con muy buenos resultados [11].

En la ED se genera de forma aleatoria una población inicial de individuos (vectores originales), de los cuales se seleccionan tres, también aleatoriamente y que sean distintos entre sí. A cada uno de estos individuos se le denomina vector objetivo y se denotan por *r1*, *r2* y *r3*, en donde *r3* es el vector base o principal. Con ayuda de un operador especial (*F*) se realiza una combinación lineal con las diferencias entre *r1*, *r2* y *r3*, con la intención de generar nuevos vectores ruidosos, a esta etapa se le conoce como mutación. *F* es entonces, un factor que escala la diferencia entre vectores. Posteriormente se generan vectores de prueba a partir de la recombinación de vectores ruidosos y vectores originales, lo anterior se logra a través de un parámetro denominado tasa de recombinación (*CR*). Para finalizar el algoritmo se procede a seleccionar el vector que se copiará a la generación siguiente, comparando los vectores de prueba con los originales en base a su aptitud.

La versión más común de la ED es la denominada **DE/rand/1/bin**; que fue la que se abordó en este trabajo y se lista en Algoritmo 1, se recomienda referirse a [12] para mayores detalles del mismo. La cantidad de generaciones, el tamaño de la población y los parámetros *CR* y *F*, son definidos por el usuario.

Algoritmo 1. Evolución Diferencial estándar, versión DE/rand/1/bin

Begin
$G = 0$
Crear aleatoriamente la población inicial $\vec{x}_G \forall i, i = 1, \ldots, NP$
Evaluar $f(\vec{x}_G) \forall i, i = 1, \ldots, NP$
For $G = 1$ to G_{max} **Do**
For $i = 1$ to NP **Do**
Seleccionar aleatoriamente $r_1 \neq r_2 \neq r_3$
j_{rand} = randint(1,D)
For $j = 1$ to D **Do**
If (rand$_j$[0,1)<CR or j=j_{rand}) **then**
$u^i_{j,G+1} = x^{r_3}_{j,G} + F\left(x^{r_1}_{j,G} - x^{r_2}_{j,G}\right)$
Else
$u^i_{j,G+1} = u^i_{j,G}$
End if
End for
If $(f(\vec{u}^i_{G+1}) \leq f(\vec{x}^i_G))$ **then**
$\vec{x}^i_{G+1} = \vec{u}^i_{G+1}$
Else
$\vec{x}^i_{G+1} = \vec{x}^i_G$
End if
End For
$G = G + 1$
End For
End

La versión con población reducida de este mismo algoritmo se observa modularmente en la Figura 2. En nuestros experimentos con ED utilizamos una población de 5 individuos, ya que se comprobó funcionalmente que con un menor número de individuos se obtiene una convergencia prematura y utilizar más de 5 individuos no mejoran los resultados. Se determinó que el criterio para alcanzar la convergencia nominal fuera al cabo de 5 generaciones, refiriéndose al ciclo interno del algoritmo. En este rubro también se realizaron pruebas con 10 generaciones, sin observarse alguna mejoría en los resultados.

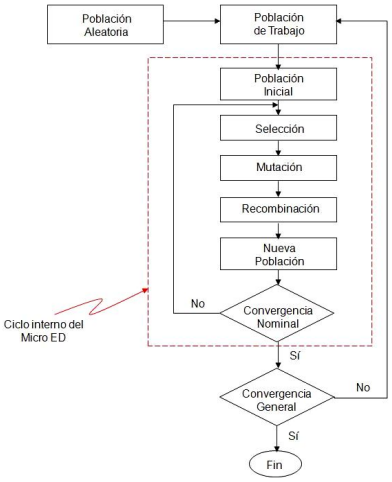

Fig. 2. Diagrama a bloques del micro algoritmo de ED.

En la generación cero, utilizando una población aleatoria de 5 individuos, se procedió a copiar los mismos en la población de trabajo y en la población inicial. Nótese que para el ciclo interno en donde aplica la convergencia nominal, el algoritmo ED se implementa sin cambios en sus operadores y estrategias para evaluar la metodología de manera básica. Al alcanzarse la convergencia nominal es necesario determinar cuántos individuos se copiarán a la población de trabajo y aplicar el proceso de reinicialización. Después de una serie de experimentos, se decidió copiar los 4 mejores individuos y el restante se genera aleatoriamente. Los experimentos realizados contemplaron copiar 1, 2, 3 y 4 individuos con la mejor aptitud, sin embargo con 4 individuos se obtuvieron los mejores resultados. En este micro algoritmo la diversidad se mantiene gracias a las etapas de mutación y recombinación del propio algoritmo base de ED y al individuo aleatorio incorporado en la población de trabajo.

4 Experimentos y Resultados

Para los experimentos se utilizaron las siguientes funciones de alta dimensionalidad, referidas en [12]. ***f1*** y ***f2*** son funciones unimodales y separables. ***f5*** es una función multimodal y no separable.

f1 – Esfera de De Jong

$$f(x) = \sum_{i=1}^{30} (x_i)^2 \qquad (1)$$

$-100 \leq x_i \geq 100$

$min\ (f_1) = f_1\ (0, \ldots, 0) = 0$

f2 – Función techo

$$f(x) = \sum_{i=1}^{30} (\lfloor x_i + 0.5 \rfloor)^2 \quad (2)$$

$-100 \leq x_i \geq 100$

$min\ (f_2\ ) = f_2\ (0, \ldots, 0) = 0$

f5 – Función generalizada de Rosenbrock

$$f(x) = \sum_{i=1}^{29} \left| 100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2 \right| \quad (3)$$

$-30 \leq x_i \geq 30$

$min\ (f_3\ ) = f_3\ (1, \ldots, 1) = 0$

Las tres funciones fueron probadas en el micro algoritmos ED, realizando 20 ejecuciones para cada una y obteniendo los resultados que a continuación se discutirán. Se definieron los parámetros indicados en la Tabla 1, los cuales son sugeridos en [12]:

Tabla 1: Parámetros utilizados para el micro algoritmo de ED.

Función	CR	F
f1	0.9	$\in [0.3,0.9]$
f2	0.0	$\in [0.3,0.9]$
f5	0.0	$\in [0.3,0.9]$

En la Tabla 2 se indican las generaciones utilizadas para cada función, así como los resultados experimentales alrededor del valor óptimo.

Tabla 2: 20 ejecuciones del micro algoritmo de ED.

Función	Ciclo Externo	Ciclo interno	Mejor	Peor	Media
f1	300	5	0.0	0.0064	0.00010
f2	200	5	0.0	1.0	0.20
f5	300	5	0.0	0.0010	0.00012

En la Figura 3, se aprecia el comportamiento del algoritmo de ED estándar en su versión **DE/rand/1/bin** con respecto a la aptitud, para ***f2***. Se utilizó una población de 60 individuos, 1000 generaciones y los mismos valores de *CR* y *F* listados en la Tabla 1.

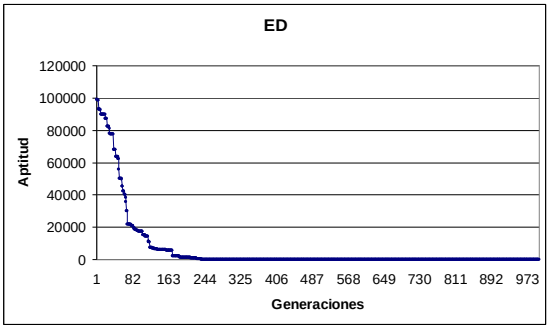

Fig. 3. Aptitud para ***f2*** con ED estándar.

Utilizando una población de 5 individuos y los parámetros indicados en la Tabla 2, se obtuvo la gráfica exhibida en la Figura 4, para ***f2*** probada en el micro algoritmo de ED.

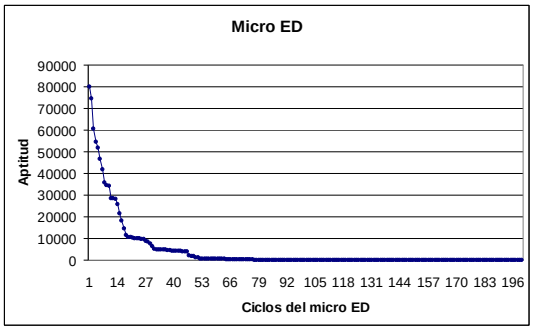

Fig. 4. Aptitud para ***f2*** con micro algoritmo de ED.

Es importante notar que para las funciones experimentales, el desempeño del algoritmo de población reducida es tan eficiente como el del algoritmo en su versión estándar. La cantidad de evaluaciones a la función objetivo es similar, considerando el ciclo interno del micro algoritmo.

5 Conclusiones y Trabajos Futuros

En este artículo se presentó una metodología generalizada que permite diseñar (o en su caso, adaptar) un algoritmo bioinspirado para que funcione con una población de tamaño reducido. Los elementos principales en esta metodología consisten en incorporar una convergencia nominal que se alcance después de un número fijo de iteraciones, un elitismo que preserve al mejor o mejores individuos encontrados al

término de la convergencia nominal y un criterio para incorporarlos como parte de una nueva población al momento de reinicializar el algoritmo. Mantener la diversidad es uno de los mayores retos en los micro algoritmos, por lo que en este trabajo se aportaron algunos criterios para cumplir con este requisito.

El micro algoritmo de ED (en su versión *DE/rand/1/bin*) generó buenos resultado con base en la metodología aplicada. En los resultados experimentales se obtuvo el óptimo para las tres funciones de prueba de alta dimensionalidad, lo que denota resultados alentadores para continuar con la investigación. Es posible afirmar que el desempeño del micro algoritmo de ED es tan bueno como el de su contraparte estándar, aunque con menores requerimientos de espacio en memoria de datos, lo que permite enfocar el diseño hacia aplicaciones con un bajo costo computacional o a implementaciones en circuitos embebidos (p. e. dispositivos de lógica programable y microcontroladores convencionales) refiriéndose al denominado *hardware evolutivo*. El manejo de espacios restringidos y la optimización con múltiples objetivos se pueden incorporar al esquema básico planteado, seguramente éste será un trabajo a futuro, al igual que explorar la adaptación de otros algoritmos bioinspirados comunes en la resolución de problemas de optimización.

6. Referencias

1. D. Ashlock. Evolutionary Computation for Modeling and Optimization. Springer, first edition, 2005.
2. M. Munetomo, Y. Satake. Enhancing Model-building Efficiency in Extended Compact Genetic Algorithms. Systems, Man and Cybernetics, 2006. ICSMC '06. IEEE International Conference on Volume 3, 8-11 Oct. 2006 Page(s):2362 – 2367.3.
3. J. Torresen. Possibilities and limitations of applying evolvable hardware to real-world application. In Field-Programmable Logic and Applications: 10th International Conference on Field Programmable Logic and Applications (FPL-2000), volume 1896 of Lecture Notes in Computer Science, R. W. Hartenstein et al. (eds.), Springer-Verlag, 2000, pp. 230–239.
4. D. E. Goldberg. Genetic Algorithms in Search, Optimization and Machine Learning. Addison Wesley, Reading, MA, 1989.
5. K. Krishnakumar. Micro-genetic algorithms for stationary and non-stationary function optimization. In SPIE Proceedings: Intelligent Control and Adaptive systems, pages 289–296, 1989.
6. G. Toscano, Carlos. A. Coello. A Micro-Genetic Algorithm for multiobjective optimization. First International Conference on Evolutionary Multi-criterion Optimization. Springer – Verlag, Lecture Notes in Computer Science number 1993, 2001, pp. 126 – 140.
7. J. C. Fuentes, C. A. Coello, Handling Constraints in Particle Swarm Optimization Using a Small Population Size, in Lecture Notes in Computer Science, Springer. MICAI 2007: Advances in Artificial Intelligence, Volume 4827/2007.
8. J. C. Fuentes. Un nuevo Algoritmo de optimización basado en optimización mediante cúmulos de partículas utilizando tamaños de población pequeños. Tesis de Maestría. CINVESTAV- IPN. México, 2008.
9. T. Bäck, D. B. Fogel, and Z. Michalewicz, editors. Handbook of Evolutionary Computation. Institute of Physics Publishing and Oxford University Press, New York, 1997.

10.R. Storn, K. Price. Differential Evolution - a Simple and Efficient Adaptive Scheme for Global Optimization over Continuous Spaces. Technical Report TR-95-012, ICSI, March 1995, ftp.icsi.berkeley.edu. Accedido por última vez el 20 de junio de 2009.
11.E. Mezura, E. A. Portilla, C. A. Coello, J. Alvarez, C. A. Cruz Villar. An Evolutionary Approach to Solve a Novel Mechatronic Multiobjective Optimization Problem. Advances in Metaheuristics for Hard Optimization 2008: 329-351
12.E. Mezura, J. Velázquez, C. A. Coello. A comparative study of differential evolution variants for global optimization. ACM, GECCO 2006: 485-492.