OPTIMIZACIÓN PARAMÉTRICA DE SISTEMAS DE IDENTIFICACIÓN DIFUSA, MEDIANTE ALGORITMOS GENÉTICOS DISTRIBUIDOS CON PERÍODO MIGRATORIO DINÁMICO

Ing. Marco Antono Castro Liera[1]
Dr. Francisco Herrera Fernández[2]
[1] Instituto Tecnológico de La Paz, México mcastroliera@acm.org
[2] Universidad Central "Martha Abreu" de las Villas, Cuba herrera@fie.uclv.edu.cu

RESUMEN

Se propone un tipo de algoritmo genético distribuido con determinación dinámica del período migratorio. El algoritmo se adapta especialmente a la optimización de los parámetros de un sistema de identificación difusa y su implementación en clusters heterogéneos. Se presentan los resultados de la optimización de un sistema Takagi-Sugeno-Kang (TSK) para la identificación de un proceso biotecnológico (fermentativo). Se incluyen el análisis de la calidad de la solución, la aceleración que se obtiene al agregar nodos al cluster y la comparación del desempeño del algoritmo usando un periodo migratorio estático y dinámico.

.

Palabras claves: Algoritmos Genéticos Distribuidos, Clusters, Identificación de Sistemas, Sistemas Difusos, Modelo TSK.

1. INTRODUCCIÓN

Cuando se combinan sistemas difusos y algoritmos genéticos, se crea una sinergia en la que, como lo expresa Zadeh (Zadeh 2001), la principal contribución de la lógica difusa es lo que puede llamarse el cálculo de reglas difusas, mientras que los algoritmos genéticos contribuyen con la metodología de búsqueda aleatoria sistematizada inspirada en los procesos evolutivos en especies vivas y propuesta originalmente por John H. Holland (Holland 2001).

El presente trabajo se centra en el uso colaborativo de un algoritmo genético distribuido, y un modelo difuso TSK (Cordón, et al. 2001) para resolver un problema de identificación altamente no lineal, en este caso, un proceso fermentativo. Una de las principales motivaciones para el desarrollo mismo, es la de poner en uso alternativas de bajo costo en cómputo paralelo, que permitan abordar problemas computacionalmente complejos sin tener que adquirir computadoras paralelas que, para instituciones educativas con presupuestos limitados son demasiado onerosas.

Los algoritmos genéticos distribuidos, también conocidos como algoritmos genéticos de grano grueso, o algoritmos de modelo de islas, actualmente son estudiados como una de las formas más escalables de paralelización de algoritmos genéticos. (Nowostawski y Ricardo 1999; Alba y Toya 2001; Alba y Tomassini 2002)

El algoritmo se diseñó teniendo en cuenta las limitaciones que los clusters presentan cuando se comparan con maquinas paralelas reales, en particular el problema del bajo ancho de banda de interconexión entre los procesadores, por lo que resulta ser

especialmente adaptado para ser ejecutado en una arquitectura paralela de bajo costro conocida como cluster.

Se puede definir un cluster como un conjunto de computadoras (nodos) conectadas a través de una red, que mediante el paso de mensajes emulan a una máquina paralela. Un cluster heterogéneo se define como aquel cuyos nodos difieren entre si en poder computacional.

2. EL PROCESO FERMENTATIVO

Para llevar a cabo el proceso de fermentación, se cuenta con cierta cantidad de microorganismos (biomasa) que se encuentran suspendidos en un medio rico en alimento (sustrato).

Este tipo de procesos presentan un comportamiento altamente no lineal que corresponden a un sistema dinámico con la estructura general: (Herrera Fernández et al. 2003)

$$\frac{dX}{dt} = f(X) + bu \tag{1}$$

Donde X es el vector formado por las dos variables de estado $[x,s]$ (concentración de biomasa y sustrato respectivamente), u es el sustrato de entrada y b es la razón constante de dilución D.

3. LA ESTRUCTURA DEL MODELO DIFUSO

Nos encontramos con un modelo con tres variables de entrada, x_t, s_t y s_{in}, que representan respectivamente la concentración de biomasa, sustrato y sustrato de entrada en el instante t. El modelo deberá determinar la concentración de biomasa en el siguiente instante $x_{(t+1)}$.

Las funciones de membresía propuestas para el sistema difuso en las variables de entrada son del tipo campana generalizada de la forma:

$$\Phi(x,\alpha,\beta,\gamma) = \frac{1}{1+\left|\frac{x-\gamma}{2\alpha}\right|^{2\beta}} \tag{2}$$

La base de reglas difusas tiene la siguiente estructura:

$$\text{IF } x_t \text{ IS } \Phi_j \text{ AND } s_t \text{ IS } \Phi_k \text{ THEN}$$

$$x_{i(t+1)} = a_{i0} + a_{i1}x_t + a_{i2}s_t + a_{i3}s_{in} \tag{3}$$

Con j variando desde 1 hasta el número de conjuntos difusos que dividen a x_t (en nuestro caso 3), k variando desde 1 hasta el número de conjuntos difusos que dividen a s_t (en nuestro caso 3) e i variando desde 1 hasta el numero de reglas difusas en la base de reglas (en nuestro caso 9).

Para aquellos procesos donde el flujo de entrada de biomasa se mantiene constante, los términos a_{i0} y $a_{i3}s_{in}$pueden consolidarse, lo cual resulta en una forma simplificada de la base de reglas:

IF x_t IS Φ_j AND s_t IS Φ_k THEN

$$x_{i(t+1)} = a_{i0} + a_{i1}x_t + a_{i2}s_t \quad \textbf{(4)}$$

La biomasa predicha se calcula como el promedio ponderado de la salida de las reglas mediante:

$$x_{(t+1)} = \frac{\sum_{i=1}^{9} h_i x_{i(t+1)}}{\sum_{i=1}^{9} h_i} \quad \textbf{(5)}$$

Donde el grado en que cada regla se cumple es calculado mediante el operador T, en nuestro caso el producto.

$$h_i = T(\Phi_j(x_t), \Phi_k(s_t)) \quad \textbf{(6)}$$

4. EL ALGORITMO GENÉTICO.

La estructura del cromosoma propuesta para el algoritmo genético es el vector real:

$$[\alpha_j, \beta_j, \gamma_j, \alpha_k, \beta_k, \gamma_k | a_{i0}, a_{i1,} a_{i2}] \quad \textbf{(7)}$$

Donde α_j, β_j y γ_j representan los parámetros de la función de membresía del j-ésimo conjunto difuso en x_t. α_k, β_k y γ_k representan los parámetros de la función de membresía para el k-ésimo conjunto difuso en s_t. Asimismo, a_{i0}, a_{i1} y a_{i2} representan los coeficientes de la función de salida de la i-ésima regla difusa.

Por tanto, el problema consiste en encontrar un total de 18 parámetros para la parte antecedente del modelo TSK y 27 coeficientes para la parte consecuente que minimicen el error de predicción.

Se decidió utilizar un algoritmo genético distribuido, debido a que este enfoque de paralelización puede ajustarse para limitar el uso de la comunicación entre procesos lo cual mejorará su rendimiento en la arquitectura paralela de bajo costo que se quiere utilizar por otra parte, la escalabilidad de este tipo de algoritmos ha sido comprobada en varios trabajos (Alba y Toya 2001; Alba y Tomassini 2002).

Este tipo de algoritmo cuenta con un conjunto de sub-poblaciones que ejecutan de manera independiente una cantidad de generaciones del algoritmo genético,

Una tarea maestra esta a cargo de la porción crítica del algoritmo, mientras que cada sub-población es procesada por una tarea esclava. Esta forma de división de las tareas se eligió ya que permite a la tarea maestra conocer los tiempos de ejecución de cada tarea esclava y ajustar acorde a los mismos los parámetros de cada sub-población.

5. CLUSTER HETEROGÉNEO

Definimos un cluster heterogéneo como aquel en el que los nodos que lo componen tienen diferentes arquitecturas y velocidades de procesamiento.

Si el algoritmo utilizado asume un cluster homogéneo (uno en el que todos los nodos tienen el mismo poder computacional), y asigna la misma carga de trabajo a cada tarea, cuando el algoritmo se ejecuta en un cluster heterogéneo, los nodos con un poder computacional mayor tendrán que esperar hasta que aquellos que resulten más lentos hayan terminado sus cálculos antes de poder efectuar el proceso de migración, lo cual se traduce en tiempos muertos en los nodos más veloces.

La alternativa que se eligió para minimizar esta desventaja es la de determinar de forma dinámica el período migratorio para cada sub-población en dependencia de su tiempo de ejecución. Esta forma es una variación de lo que se denomina un algoritmo genético asíncrono (Alba and Toya 2001), en el sentido de que permite no solo reducir la cantidad de generaciones para los nodos más lentos, sino de incrementar aquellos que son más veloces. Las figuras 1 y 2 muestran el algoritmo propuesto.

```
Crear P sub-poblaciones
para i = 1 hasta P
  enviar parámetros a la sub-población i
para i = 1 hasta GMAX/MP
  para j = 1 hasta MR
    para k = 1 hasta P
      recibir MR individuos de la sub-población k
      recibir T[k](el tiempo de procesamiento de la sub-población k)
    calcular el tiempo promedio de ejecución TP
    para k = 1 hasta P
      enviar MR individuos de la sub-población k a la sub-población P-(k+1)
    MP = MP*(TP/T[k])
    si MP > MPMAX
      MP = MPMAX
    Enviar el nuevo MP a la población P-(k+1)
desplegar el individuo con mayo aptitud como la solución
```

Fig 1. Tarea maestra

```
recibir parámetros del maestro
generar la población inicial aleatoria
número de ciclos C = GMAX/MP
para k=1 hasta C
  para i=1 hasta MP
    selección
    cruce
    mutación
    g = g + 1
  enviar MR individuos al maestro
  recibir MR individuos del maestro
  reemplazar los MR peores individuos con los recibidos
  recibir el nuevo MP
```

Fig. 2: Tarea esclava

Con este nuevo enfoque, aquellas sub-poblaciones con un tiempo de ejecución por debajo del promedio, ejecutan más iteraciones del algoritmo antes de la siguiente migración, mientras aquellas con un tiempo de ejecución mayor al promedio ejecutan menos iteraciones. Esta estrategia progresivamente equilibra el tiempo de ejecución de todas las poblaciones, de tal suerte que los tiempos de espera disminuyen.

6. OPTIMIZACIÓN DE DOS FASES

El proceso de optimización fue dividido en dos etapas; la primera busca encontrar un conjunto de coeficientes para la parte consecuente de la base de reglas difusas que minimicen el error total calculado como:

$$\sum_{s=1}^{n} \left| xc_s - xm_s \right| \quad \textbf{(8)}$$

Donde n es el número total de muestras de entrenamiento (en nuestro caso se contó con 490 muestras), xc es la concentración de biomasa calculada y *xm* la concentración de biomasa medida.

La segunda fase del proceso de optimización intenta encontrar un conjunto de parámetros para las funciones de membresía que minimicen el error máximo dado por:

$$\max(\left| xc_1 - xm_1 \right|, \ldots, \left| xc_n - xm_n \right|) \quad \textbf{(9)}$$

Para llevar a cabo la primera fase de optimización, se eligen parámetros para las funciones de campana generalizada que crean conjuntos difusos que dividen uniformemente el espacio de las variables de entrada:

$$\begin{aligned} &\alpha_j = 2 \quad \alpha_k = 2 \\ &\beta_j = 4 \quad \beta_k = 4 \\ &\gamma_j = x_{\min} + \frac{(x_{\max} - x_{\min})(2j-1)}{6} \\ &\gamma_k = s_{\min} + \frac{(s_{\max} - s_{\min})(2k-1)}{6} \end{aligned} \quad \textbf{(10)}$$

Dichos parámetros fueron utilizados para encontrar un conjunto de 27 coeficientes de funciones de salida que minimicen el error total de salida del modelo resultante calculado mediante la expresión (8).

Para la segunda parte del proceso de optimización, el mejor conjunto de coeficientes es dejado fijo y se trata de encontrar un conjunto de parámetros para las funciones de membresía que minimicen el error de predicción máximo dado por (9).

7. PARÁMETROS DEL ALGORITMO GENÉTICO

El cromosoma usa representación real en una matriz de 3 por 9 que corresponde a cada conjunto de 3 coeficientes que las 9 funciones de salida requieren.

Los parámetros elegidos para la primera fase de optimización fueron:

- Tamaño de población MU=600.
- Número máximo de generaciones GMAX=160.
- Selección por torneo, con tamaño de torneo Z=2.
- Probabilidad de cruce PC=0.7
- Parámetro de cruce λ generado con una distribución uniforme en (0,1).
- Mutación uniforme, con probabilidad de mutación PM=0.2
- Rangos de búsqueda: [-7,7] para a_{i0}, [-4,4] para a_{i1} y [-2,2] para a_{i2}.
- Período migratorio MP=20 generaciones.
- Máximo período migratorio MPMAX=30.
- Razón de migración MR=2 individuos

Los parámetros elegidos para la segunda fase son:

- Tamaño de la población IMU=300.
- Número máximo de generaciones IGMAX=60.
- Selección por torneo, con tamaño de IZ=2.
- Probabilidad de cruce IPC=0.7
- Parámetro de cruce λ generado con una distribución uniforme en (0,1).
- Mutación uniforme, con una probabilidad de mutación IPM=0.2.
- Rangos de búsqueda:

$$\alpha_j = 2 \pm 1 \quad \alpha_k = 2 \pm 1$$
$$\beta_j = 4 \pm 1 \quad \beta_k = 4 \pm 1$$
$$\gamma_j = x_{\min} + \frac{(x_{\max} - x_{\min})(2j-1)}{6} \pm \frac{(x_{\max} - x_{\min})}{6}$$
$$\gamma_k = s_{\min} + \frac{(s_{\max} - s_{\min})(2k-1)}{6} \pm \frac{(s_{\max} - s_{\min})}{6} \quad \textbf{(11)}$$

- Período migratorio IMP=20.
- Máximo período migratorio IMPMAX=30.
- Razón de migración IMR=2.

8. RESULTADOS

Se llevaron a cabo varias pruebas para medir la calidad de los modelos obtenidos, así como, el comportamiento del tiempo de ejecución bajo diferentes condiciones. El tamaño de muestra para cada juego de datos fue de 200 corridas que se llevaron a cabo en un cluster heterogéneo con un switch fast-ethernet y dos tipos de nodos:

(a) Pentium IV@2.8GHz, 1024 KB de caché y 512 MB DDR2 RAM@400 MHz.
(b) Pentium IV@2.0GHz, 512 KB de caché y 512 MB DDR2 RAM@266 MHz.

8.1. Calidad de la solución

La tabla 1 muestra el comportamiento del error cuando se varía el número de sub-poblaciones.

Tabla 1: Comportamiento del error

SP	ETP	EP	EMP
4	17.1492789	0.03499853	0.13239199
8	14.6136543	0.02982378	0.11224275
12	13.7827334	0.02812803	0.10273310
16	13.5702353	0.02769436	0.10423106
18	12.9202716	0.02636790	0.09917504

Donde SP es el número de sub-poblaciones, ETP el error total promedio, EP el error promedio por muestra, y EMP el error máximo promedio.

Es claro que al aumentar la cantidad de sub-poblaciones, mejora la calidad de la solución si se mantienen constantes el resto de los parámetros. La figura 3 compara la salida predicha por el modelo con la real, así como el error de predicción de un modelo típico generado con 8 sub-poblaciones. La figura 4 compara las salidas predichas y medida, así como el error de predicción con un modelo promedio generado con 18 poblaciones.

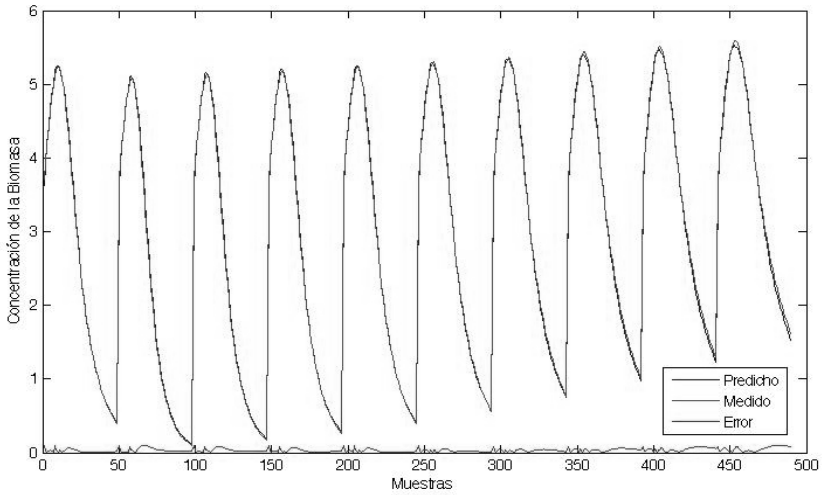

Fig. 3: Calidad de la predicción (4 sub-poblaciones)

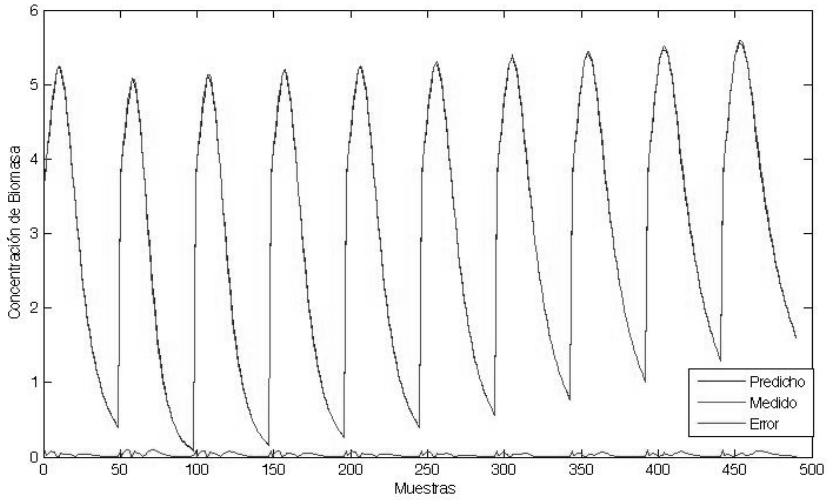

Fig. 4: Calidad de la predicción (18 sub-poblaciones)

8.2. Velocidad de ejecución

Para conseguir mejores tiempos de ejecución, manteniendo soluciones razonablemente buenas, los siguientes parámetros fueron modificados:

- Tamaño de población MU=400.
- Número máximo de generaciones GMAX=120.
- Periodo migratorio MP=15.
- Máximo período migratorio MPMAX=20.
- Tamaño de población IMU=200.
- Número máximo de generaciones IGMAX=30.
- Período migratorio IMP=15.
- Máximo período migratorio IMPMAX=20.

Con 18 sub-poblaciones, se obtuvo un error promedio total de 15.2724839, que representa un error promedio por muestra de 0.03116833; el error máximo promedio fue

de 0.132167275. Los siguientes tiempos de ejecución se obtuvieron al ejecutar el algoritmo con 9 nodos del tipo (a):

Tabla 2: Tiempos de ejecución, cluster homogéneo

Nodos	Segundos	Aceleración
1	879	1.00
2	453	1.94
3	340	2.59
6	177	4.97
9	109	8.70

La figura 5 muestra la aceleración obtenida, comparada contra una aceleración lineal.

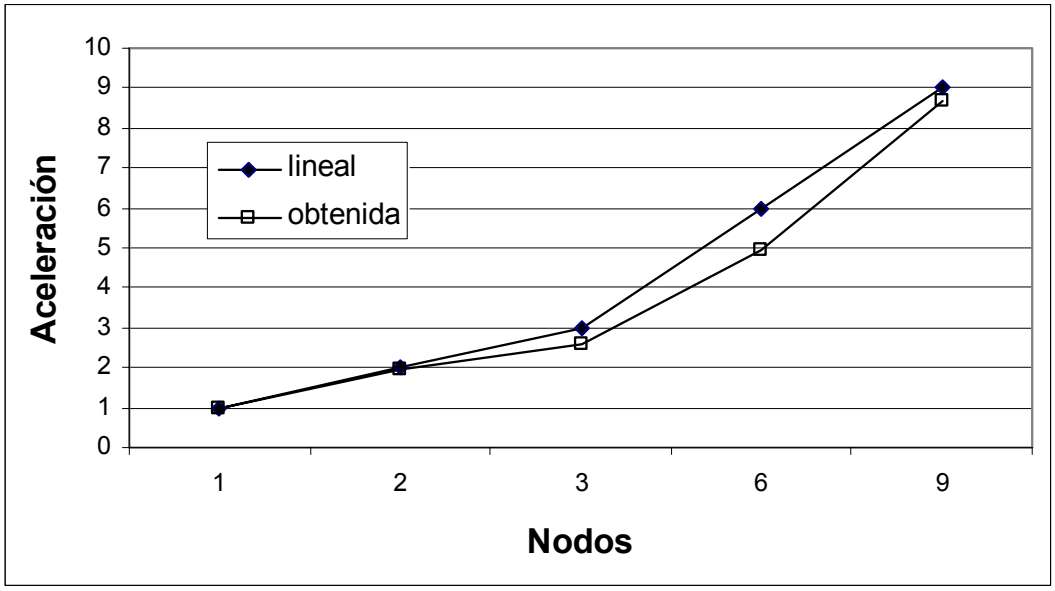

Fig. 5: Aceleración cluster homogéneo.

Con un cluster heterogéneo de 10 nodos tipo (a) y 8 nodos tipo (b) se obtuvieron los siguientes tiempos de ejecución:

Tabla 3: Cluster heterogéneo, período migratorio fijo

Nodos		Segundos	Aceleración
(a)	(b)		
	1	2448	1.00
	2	1252	1.96
	3	846	2.91
	6	422	5.80
1	8	279	8.77
10	8	141	17.36

Tabla 4: Cluster heterogéneo, período migratorio dinámico

Nodos		Segundos	Aceleración
(a)	(b)		
	1	2454	1.00
	2	1253	1.96
	3	852	2.88
	6	404	6.07
1	8	243	10.10
10	8	100	24.54

Los tiempos de ejecución muestran claramente la mayor velocidad de los nodos de tipo (a). La tabla 3 deja en evidencia la imposibilidad del algoritmo distribuido con periodo migratorio constante de aprovechar el mayor poder computacional de los nuevos nodos agregados al cluster. De hecho puede notarse que aún los 18 nodos del cluster heterogéneo son más lentos que los 9 de la tabla 2. Por otra parte la versión con período migratorio dinámico es capaz de hacer uso del mayo poder computacional de los nuevos nodos agregados produciendo una aceleración súper-lineal.La figura 6 muestra la aceleración de ambos enfoques comparada contra una aceleración lineal.

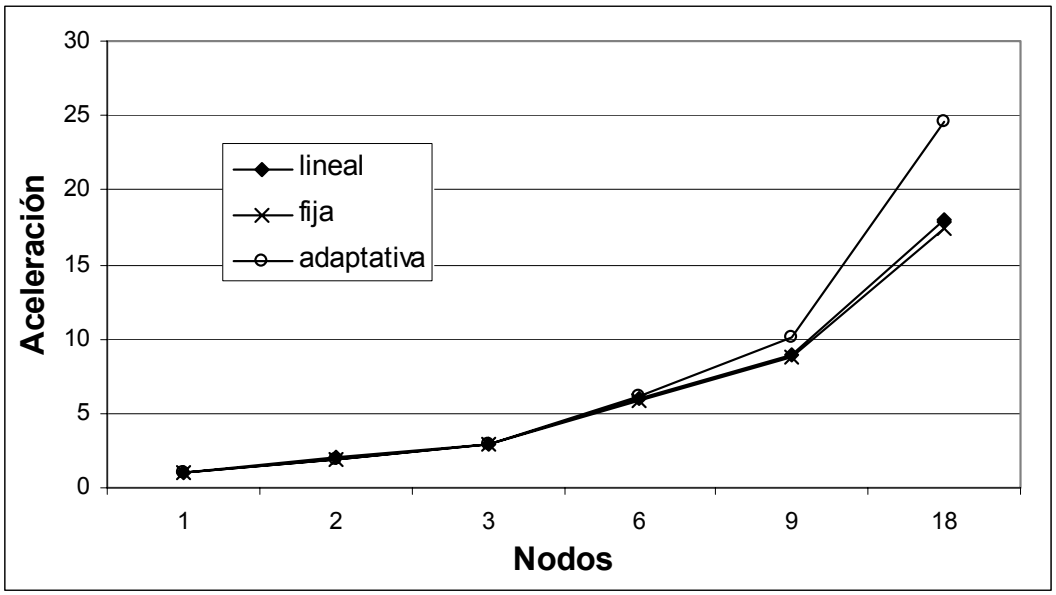

Fig. 6: Aceleración, cluster heterogéneo

9. CONCLUSIONES

El algoritmo propuesto es capaz de generar soluciones adecuadas para el problema de identificación.

Los resultados que se presentan muestran que la calidad de la solución mejora al aumentar la cantidad de sub-poblaciones, también se hace evidente que se pueden mantener tiempos de ejecución bajos al incrementar la cantidad de sub-poblaciones cuando agregamos nodos al cluster.

Las pruebas de aceleración para un cluster de 18 computadoras muestran que, al menos para esta cantidad de nodos, el algoritmo puede ser escalado sin una pérdida considerable de desempeño.

Se puede observar que mediante la determinación dinámica del período migratorio, es posible conseguir aceleraciones súper-lineales, cuando se agregan nodos más veloces al cluster.

El código fuente de la aplicación escrito en C para GNU/Linux y PVM 3.4 esta disponible en http://sistemas.itlp.edu.mx/castroga/biomasa-full.tgz.

REFERENCIAS

Alba, E. y M. Tomassini (2002). *Parallelism and Evolutionary Algorithms*. IEEE Transactions on Evolutionary Computation **6**(5): 443-462.

Alba, E. y J. M. Toya (2001). *Analyzing Synchronous and Asyncrhonous Parallel Distributed Genetic Algorithms*. Future Generation Computer Systems **17**(4): 451-465.

Cordón, O., F. Herrera, et al. (2001). *Genetic fuzzy systems : evolutionary tuning and learning of fuzzy knowledge bases*. Singapore, World Scientific.

Herrera Fernández, F., B. Martinez Jiménez, et al. (2003). *Aplicación de las Técnicas de la Inteligencia Artificial en un Proceso Biotecnológico de Reproducción Celular (Embriogénesis Somática)*. Santa Clara, Libre, Universidad Central "Marta Abreu" de Las Villas: 38.

Holland, J. H. (2001). *Adaptation in Natural and Artificial Systems*. Michigan, MIT Press.

Nowostawski, M. y P. Ricardo (1999). *Parallel Genetic Algorithm Taxonomy.* Third International Conference of Knowledge-Based Intelligent Information Engineering Systems.

Zadeh, L. A. (2001). *Foreword. Genetic fuzzy systems: evolutionary tuning and learning of fuzzy knowledge bases*. O. Cordon, F. Herrera, F. Hoffman y L. Magdalena.