Algoritmos PSO y DE aplicados al problema de inestabilidad en sistemas multiagentes nómadas

Alejandro Sosa, Víctor Zamudio, Rosario Baltazar, Carlos Lino, Miguel Angel Casillas y Marco Sotelo

División de Estudios de Posgrado e Investigación, Instituto Tecnológico de León, Av. Tecnológico S/N, 37290 Guanajuato, México
alejandro_sosa@ieee.org, vic.zamudio@ieee.org, r.baltazar@ieee.org, carloslino@itleon.edu.mx, miguel.casillas@ieee.org, masotelof@ieee.org
http://posgrado.itleon.edu.mx/
Paper received on 22/09/12, Accepted on 21/10/12.

Resumen El problema de inestabilidad cíclica en un sistema multi-agente nómada basado en reglas, se origina cuando un agente externo se incorpora al sistema, provocando así ajustes en las topologías de interconexión. Dichos ajustes pueden ocasionar comportamientos oscilatorios en el sistema, lo cual no es deseable. Se ha demostrado que dichas oscilaciones puede ser minimizadas mediante el bloqueo de uno o varios agentes. Los algoritmos evolutivos, como el PSO y el DE, pueden ser utilizados para la optimización numérica siempre y cuando se tenga una función objetivo. En el presente artículo se muestran los resultados de aplicar estos algoritmos al problema de inestabilidad cíclica, tratando de minimizar la función definida como el promedio de cambios en el sistema. Para dichas pruebas se generaron diversas instancias a las cuales se les aplicaron los algoritmos evolutivos antes mencionados. Los resultados obtenidos por dichos algoritmos se contrastaron mediante la prueba de Wilcoxon para poder determinar cual algoritmo tuvo mejor desempeño.

Palabras claves: PSO, DE, Inestabilidad Cíclica, Agentes Nomadas,Topologías de Interconexión

1. Introducción

Los ambientes inteligentes[1] se basan en el concepto de la computación ubicua, es decir, la integración de la informática en nuestro entorno de forma que los ordenadores y otros sistemas de procesamiento no sean elementos diferenciables, sino que formen parte natural de esos entornos. Basándose en ello, tenemos los entornos inteligentes, que son entornos digitales formados por redes de dispositivos inteligentes, que detectan la presencia y acciones del usuario y actuar en consecuencia (es decir, de manera reactiva), e incluso se pueden anticipar y adaptar a las necesidades y preferencias del usuario. Unos de los retos de los entornos inteligentes es minimizar o eliminar la inestabilidad que se pudiera generar en ellos, además de ser un problema poco abordado en

Mireya García-Vázquez, Grigori Sidorov (Eds.).
Advances in Computing Science and Control.
Research in Computing Science 59, 2012, pp. 171- 182

la literatura[2].
Una de las opciones propuestas es el algoritmo INPRES, basado en el uso de locking (que consiste en no permitir a un agente cambiar de estado) y la detencion de ciclos en la Interaction Network asociada. Con INPRES, se pretende obtener un sistema estable, con la restricción de que la cantidad de agentes bloqueados sea el mínimo para evitar que este bloqueo no sea prioridad dentro del sistema sino una herramienta para tener un control del sistema; este sistema es llamado c-INPRES[3] es la nueva versión del algoritmo antes mencionado donde se pretende que los agentes bloqueados sea el mínimo posible dentro del sistema. Uno de los algoritmos que se ha probado de manera exitosa usando el locking en ambientes estáticos es la optimización mediante cúmulo de partículas (Particle Swarm Optmization, PSO)[4] pues ha permitido reducir al mínimo las oscilaciones del sistema. Otro de los algoritmos usados en ambientes estáticos es el de DE[5] [6] el cual mantiene una población de soluciones candidatas, las cuales se recombinan y mutan para producir nuevos individuos los cuales serán elegidos de acuerdo al valor de su función de desempeño. Debido a lo anterior, en este trabajo representamos los resultados de haber aplicado los algoritmos PSO y DE a sistemas multiagentes dinámicos; es decir, donde los agentes presentan un comportamiento nómada.

2. Algoritmo de Evolución Diferencial

La evolución diferencial(Differential Evolution, DE)[5] [6] es una rama de la computación evolutiva desarrollada por Rainer Storn y Kenneth Price para optimización en espacios continuos, aplicado en la resolución de problemas complejos. Al igual que otros algoritmos de esta categoría, la DE mantiene una población de soluciones candidatas, las cuales se recombinan y mutan para producir nuevos individuos los cuales serán elegidos de acuerdo al valor de su función de desempeño. Lo que caracteriza a la DE es el uso de vectores de prueba, los cuales compiten con los individuos de la población actual a fin de sobrevivir. La DE es un método de búsqueda que utiliza N vectores:

$$x_{i,G}; i = 0, 1, 2, ..., N - 1 \tag{1}$$

como la población de cada generación (G). El valor de N no cambia durante el proceso de optimización. La población inicial se elige de manera aleatoria si no se conoce nada acerca del problema. Como regla, se asume una distribución uniforme para las decisiones aleatorias a tomar. La idea principal detrás de la DE es un nuevo esquema para generar vectores. La DE genera estos nuevos vectores cuando se suma la diferencia de pesos entre dos vectores miembros de la población a un tercer vector miembro. Si la aptitud del vector resultante es menor que el miembro de la población elegido entonces el nuevo vector reemplaza al vector con el cual fue comparado. Este vector a comparar puede ser (aunque no necesariamente lo es) parte del proceso de generación arriba mencionado. Además, el mejor vector $x_{mejor,G}$ se evalúa en cada generación G para no perder de vista el progreso durante el cual se hace la minimización.[7]

El algoritmo asume que las variables del problema a optimizar están codificadas como un vector de números reales. La longitud de estos vectores (n) es igual al número de variables del problema, y la población esta compuesta de np (número de padres)

vectores. Se define un vector x_p^g, en donde p es el índice del individuo en la población $(p = 1...np)$ y g es la generación correspondiente. Cada vector está a su vez compuesto de las variables del problema $x_{p,m}^g$, en donde m es el índice de la variable en el individuo $(m = 1...n)$. Se asume que el dominio de las variables del problema está restringido entre valores mínimos y máximos x_m^{min} y x_m^{max}, respectivamente. DE se compone básicamente de 4 pasos:

- Inicialización: Se genera una población inicial aleatoria con una distribución uniforme[8].
- Mutación: Se selecciona aleaoriamente tres vectores diferentes entre sí, se restan dos de ellos y a la diferencia se aplica un peso dado a ellos por un factor y por último se suma la diferencia la diferencia al tercer vector[9].
- Recombinación: Se realiza la recombinación, tomando a cada uno de los individuos de la población como el padre principal y otros 3 padres se seleccionan aleatoriamente generando un hijo. Si el hijo generado tiene un mejor valor de la función objetivo que el padre principal, entonces lo reemplaza[8].
- Selección: Todos los vectores son seleccionados una sola vez como padre principal sin depender de la función objetivo, se comprueba si el padre seleccionado resulta mejor que el hijo generado este conserva su valor de lo contrario se sustituye por el hijo[9].

Estos 4 procesos se realizan hasta que se cumpla el criterio de paró definido por el usuario, este proceso se muestra en el Algoritmo 1.

Algoritmo 1. Evolución diferencial.

```
f función objetivo a minimizar.
límites valores máximos y mínimos que pueden tomar las variables.
|P| número de individuos en la población  mayor o igual a 4.
N número de llamadas a función.
n dimensión.
F parámetro de entrada entre 0 y 2.
CR parámetro de entrada entre 0 y 1.
Return Pbest tal que f(Pbest) es el mejor fitness.
P = Generar una población aleatoria.
f(P) Se calcula el fitness de cada individuo de la población.
While {no se cumpla al numero máximo de llamadas a función}
    tempXi = Xj + F(Xk - Xl)
    for cada coordenada j del individuo
        r = numero aleatorio uniformemente distribuido _
             entre 0 y 1.
        if r <= CR
            tempXi[j]=  temXi[j] si r <= CR; en otro caso Xi[j]
        endif
    endfor
    if f(tempXi) mejor f(Xi)
        Xi =  tempXi
        f(Xi) = f(tempXi).
    endif
endWhile
```

3. Algoritmo de Optimización mediante Cúmulo de Partículas

El algoritmo de Optimización mediante Cúmulo de Partículas (Particle Swarm Optimization, PSO)[4], originalmente propuesto por Kennedy and Eberhart, en 1995 es un algoritmo bioinspirado, que se basa en una analogía con el comportamiento social del vuelo de las aves o del cúmulo de peces. Se fundamenta en el enfoque conocido como la "metáfora social", la cual puede resumirse de la siguiente forma: los individuos que conviven en una sociedad tienen una opinión que es parte de un "conjunto de creencias" compartido por todos los posibles individuos. Cada individuo, puede modificar su propia opinión basándose en tres factores:

- Conocimiento sobre el entorno (desempeño).
- Conocimiento histórico sobre experiencias personales anteriores (componente cognitivo).
- Conocimiento histórico sobre experiencias anteriores de los individuos situados en su vecindario (componente social).

Este algoritmo usa dos ecuaciones: la ecuación (2) es usada para encontrar la velocidad, describe la magnitud y la dirección del paso que será tomado por las partículas y está basado en el comportamiento adquirido hasta este momento.

$$v_{id} = wv_{id} + c_1(lBest_{id} - x_{id}) + c_2(gBest_{id} - x_{id}) \tag{2}$$

Donde:
v_{id}: es la velocidad de la i-ésima partícula.
$d = 1, 2, ..., N$, siendo N el tamaño de la dimensión.
$i = 1, 2, ..., N$, siendo N el tamaño de la población.
w es el coeficiente de adaptación al entorno.
c_i: es el coeficiente de memoria del vecindario.
c_2: es el coeficiente de memoria.
$lBest$ es el mejor partícula local encontrada por la i-ésima partícula.
$gBest$ es la mejor partícula encontrada hasta ese momento por las partículas.
x_i representa la i-ésima partícula.

La ecuación (3) actualiza la posición actual de la i-ésima partícula a la nueva posición utilizando el resultado obtenido del calculo de la velocidad.

$$x_{id} = x_{id} + v_{id} \tag{3}$$

PSO-Binario

El PSO Binario[10] fue diseñado para trabajar en espacios binarios. El PSO Binario selecciona las partículas lBest y gBest de la misma forma que el PSO. La principal diferencia con el PSO radica en la ecuación utilizada para actualizar la velocidad y la posición de la partícula. La ecuación utilizada para actualizar la velocidad se basa en probabilidades en el rango [0,1]. Para ello establece un mapeo para todos los valores

reales de la velocidad a el rango [0,1]. La ecuación 3 permite convertir el valor en binario 0 o 1.

$$v_{ij}(t) = sig(v_{ij}(t)) = \frac{1}{1 + e^{-v_{ij}(t)}} \tag{4}$$

Donde:
v_{ij} es la velocidad de la i-ésima partícula
$i = 1, 2, ...N$, siendo N el tamaño de la población
$j = 1, 2, ...D$, siendo D el numero de dimensiones

Mientras tanto la ecuación (5) es usada para actualizar la posición de la partícula a la nueva posición.

$$x_{ij}(t+1) = \begin{cases} 1 & \text{si } r_{ij} < sig(v_{ij}(t+1)) \\ 0 & \text{en otro caso} \end{cases} \tag{5}$$

Donde:
r_{ij} es un número aleatorio uniforme en el rango [0,1]
$i = 1, 2, ...N$, siendo N el tamaño de la población
$j = 1, 2, ...D$, siendo D el número de dimensiones

A continuación se muestra el Algoritmo 2, donde se observa a detalle la optimización por cúmulos de partículas.

Algoritmo 2. Optimización por Cúmulo de Partículas.

```
n: El número de partículas de la población.
LI: Límite Inferior.
LS: Límite Superior.
C1: Coeficiente de la memoria.
c2: Coeficiente de la memoria del vecindario.
w: Coeficiente de adaptación al entorno.
for cada partícula i
   Inicializar la población y buscar el mejor local
   de la población inicial.
   X= x1, x2, x3,...,xd |xi que pertenece [LI,LS]
   V= v2, v2, v3,...,vd |vi que pertenece [0,1]
   LBest =X
endfor
Buscar el mejor Global
if LBestx es mejor que GBestx
   GBest = LBest
endif
while se cumpla la condición de paro
   for cada partícula i
      Vi = wVi y c1(GBest - Xi) + c2(LBest - Xi)
      Xi = Xi + Vi
   if f(Xi) es menor que f(LBest)
      f(LBest) = f(Xi)
   endif
   endif
Buscar el mejor Global
for cada partícula i
```

```
   if LBest es mejor que GBest
      GBest = LBest
   endif
endfor
endwhile
Return GBest
```

4. Cálculo de Inestabilidad Dinámica

Uno de los retos que presenta los ambientes inteligentes es la inestabilidad ciclica.Un ejemplo es cuando un dispositivo A esta esperando una acción de un dispositivo B, sin la cual no puede continuar trabajando sin embargo el dispositivo B, requiere de información o ejecución de una tarea dada por A, generando un ciclo entre ambos dispositivos provocando que ninguno termine la tarea que se les ha asignado. Para entender mejor el fenómeno debemos retomar el hecho que dentro de los Ambientes Inteligentes existen múltiples agentes interconectados entre si, dichos agentes son gobernados por diversas reglas que afectan su comportamiento y pueden llevar a los agentes a cambiar de estado múltiples veces en un periodo de tiempo dado. De lo anterior suelen suceder múltiples casos ya documentados en donde se presenta interferencia o comportamiento no deseado entre los dispositivos[11],[12]. Al efectuar una evaluación del sistema todos los dispositivos y/o agentes que se encuentren en conflicto provocan que en general el sistema se vuelva inestable, obteniendo a su vez comportamientos erráticos o simplemente que los dispositivos no lleven a cabo tarea para la que fueron programados.

Para medir la oscilación de un sistema se han propuesto diferentes medidas de esta. En primera instancia se tiene la Oscilación Acumulativa Promedio[14] la cual trata de medir la relación de cambio entre el estado actual del sistema y el estado siguiente. En este trabajo se utiliza el Promedio de Cambios en el Sistema, ésta medida evalúa la estabilidad en base a la cantidad de cambios en los estados del sistema. [13]

Cálculo del Promedio de Cambios en el Sistema

El valor considerado para medir la inestabilidad es el Promedio de Cambios en el Sistema[14]. Para calcular este valor se utiliza la ecuación 5. En dicha ecuación se considera solo los cambios entre el estado actual del sistema con respecto al estado siguiente (sin considerar la magnitud del cambio):

$$O = \frac{\Sigma_{i=1}^{n-1} x_i}{n-1} \begin{cases} 1 \text{ si } S_i \neq S_{i+1} \\ 0 \text{ en otro caso} \end{cases} \tag{6}$$

Donde:
O: Promedio de cambios en el sistema.
n: Generaciones del ciclo de vida con S_i siendo el estado del sistema en el tiempo *i* y S_{i+1} siendo el estado del sistema en el tiempo *i+1*.

5. Minimizando la Inestabilidad en Ambientes Inteligentes

Los experimentos se realizaron bajo diferentes instancias de prueba donde cada instancia tuvo las siguientes características: se inicio con una matriz de 30x2 agentes, posteriormente se fue incrementando el tamaño y la dimensión, terminando con una matriz de 30x30 agentes. Los agentes entrantes se fueron agregando en diferentes periodos, usando la misma topología(Figura 1). La función objetivo es el cálculo del promedio de cambios en el sistema(Ecuación 6) donde se evaluan y conseervan los estados de los agentes que presentan menos oscilaciones. Con ello se pretende que el sistema no oscile, que afecten lo menos posible al sistema, para asegurar que conserve su propia integridad y funcionalidad para el usuario final. En la topología usada en el sistema se muestra el siguiente comportamiento: se está iniciando con un número definido de nodos o agentes, que en este caso son 2, con dependencias del anterior estado para su interacción. Dado que el sistema se va incrementando, los nodos se van insertando de forma sucesiva; de igual manera se pueden formar diferentes topologías de conexión, es decir, de un nodo pueden depender uno o más nodos con lo que se puede formar diferentes topologías con el mismo número de nodos, como se muestra en la Figura 1:

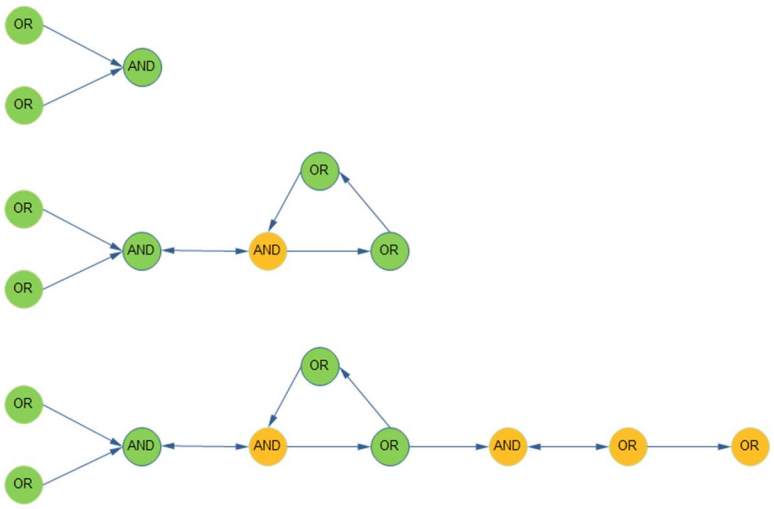

Figura 1. Topología usada en los experimentos realizados

La Figura 1 muestra el crecimiento del sistema iniciando con 2 agentes y terminando con 9, para este caso se llegará a una dimensión mayor.Las instancias de pruebas implementadas en los algoritmos fueron diseñados para incrementar el tamaño de manera uniforme iniciando un periodo del ciclo establecido, como por ejemplo en la primera instancia inicio con 2 agente y fue incrementando en diferentes periodos fijos del algoritmo terminando con 30 agentes en total,donde cada agente entrante posee sus propia

regla de interacción con los agentes ya establecidos en el sistema y con lo cual se pretende emular la entrada de agentes en un ambiente inteligente. Un escenario donde se ha dejado evolucionar al sistema se muestra en al Figura 2, utilizando la misma topología mencionada en la Figura 1 y sin aplicar ninguna técnica que permita minimizar las oscilaciones en el sistema.

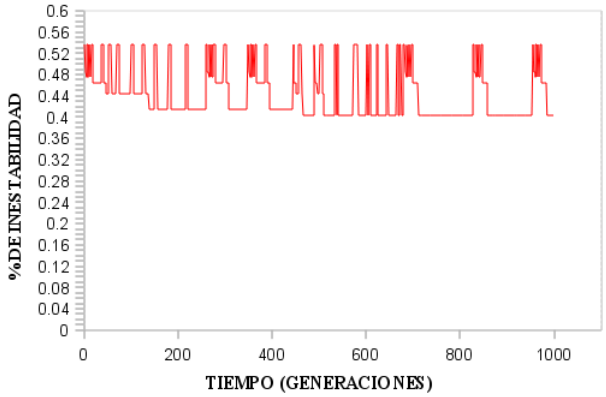

Figura 2. Gráfica de Inestabilidad

En la Figura 1 se muestra la topología de interconexión que tendría el sistema. El estado de cada agente es representado por un 0 o 1 en una cadena binaria, la cual representa su estado en un determinado tiempo y se utilizan las compuertas lógicas AND y OR como reglas de interacción respectivamente[15], formando una cadena de AND y OR representado todas las reglas en el sistema. Los parámetros utilizados para los algoritmos DE y PSO se muestran en la Tabla 1.

Tabla 1. Parámetros del algoritmo ED y PSO.

Parámetros	Valores PSO	Valores ED
Individuos	30	30
Dimensión inicial	2	2
Dimensión final	30	30
Llamadas a función	1000	1000
% de agentes bloqueados	0.2	0.2
W	1	//
C1	0.4	//
C2	0.6	//
F	//	0.9
CR	//	0.8

6. Resultados

Los resultados son mostrados en la Tabla 1 y Figura 3 y 4, los cuales permiten decir que el ambiente inteligente dinámico puede ser estabilizado exitosamente. En los experimentos realizados, se inició el sistema con 2 agentes, incrementandose hasta llegar a 30. Los algoritmos PSO y DE implementados permitiran encontrar combinaciones de agentes bloqueados, que minimizan las oscilaciones del sistema. En la Tabla 2 se muestran los resultados obtenidos en varias pruebas realizadas aplicadas a los algoritmos antes mencionados.

Tabla 2. Resultados.

Escenario	DE	PSO
Instancias 1	0.3831	0.3941
Instancias 2	0.3840	0.3985
Instancias 3	0.3737	0.4040
Instancias 4	0.3636	0.3989
Instancias 5	0.3787	0.4040
Instancias 6	0.3777	0.4144
Instancias 7	0.3643	0.4033

Los anteriores resultados fueron sometidos a la Prueba No Paramétrica de Rangos con Signo de Wilcoxon[16] para analizar la identidad estadística con respecto a los resultados del Cálculo del Promedio de Cambios en el Sistema de los algoritmos, de donde se obtuvieron T+ = 0,T− = 28, con una T0 = 4 para una muestra de 7 instancias de prueba con un nivel de significancia del 0.10. Los resultados indican que hay suficiente evidencia estadística para establecer que el algoritmo muestra un mejor desempeño, por lo cual podemos decir que el algoritmo de Evolución Diferencial se encuentra más a la izquierda obteniendo mejores resultados.

7. Conclusiones y Trabajos Futuros

En el presente trabajo se estudio el problema de inestabilidad cíclica en sistemas multiagentes. Se aplicaron y compararon los algoritmos PSO y DE bajo diferentes instancias de prueba. Con los resultados obtenidos se concluye que la inestabilidad que se genera en un escenario dinámico puede ser controlada usando alguno de estos algoritmos. Al aplicar la prueba de Wilconxon (debido a que existe suficiente evidencia estadística) se concluye que el algoritmo de Evolución Diferencial tuvo mejores resultados que el algoritmo de Optimización mediante Cúmulos de Partículas. Como trabajo futuro se plantea implementar otras técnicas que permitan mejorar los resultado obtenidos.

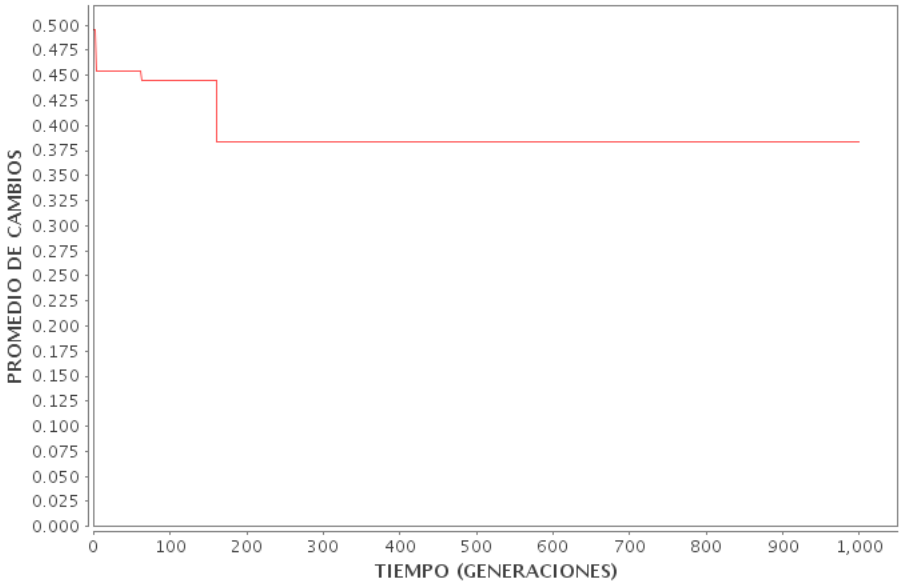

Figura 3. Promedio de Cambios en el Sistema al aplicar DE

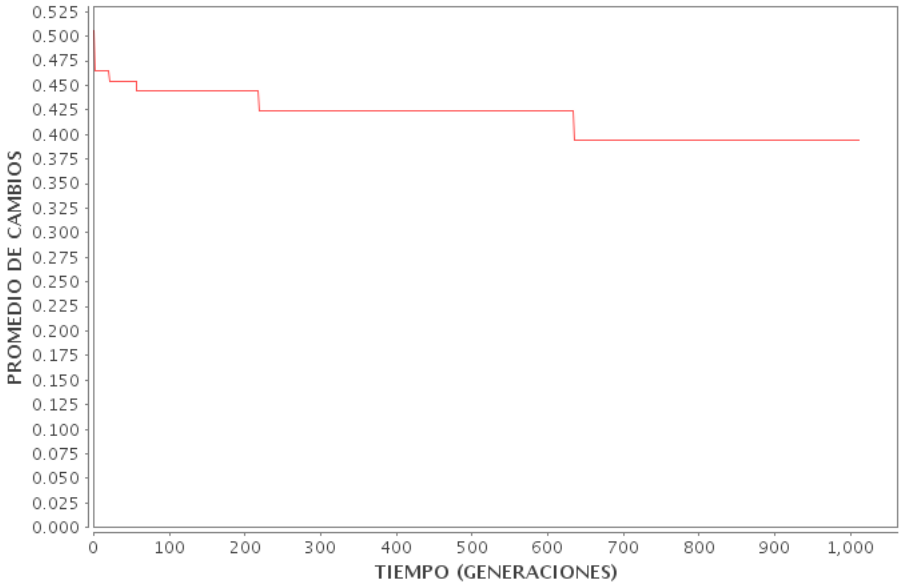

Figura 4. Promedio de Cambios en el Sistema al aplicar PSO

Agradecimientos

Alejandro Sosa Sales agradece el apoyo del Consejo Nacional de Ciencia y Tecnología (CONACyT), por el apoyo recibido para la realización de este trabajo de investigación y a la DGEST por el apoyo del proyecto 4311.11P

Referencias

1. Stuart Russell & Peter Norvig; Inteligencia Artificial, un enfoque moderno, Prentice Hall
2. Victor Zamudio, Vic Callaghan and Jeannette Chin "A Multi-dimensional Model for Task Representation and Allocation in Intelligent Environments", (2006), Volume: 3823, Publisher: Springer Berlin Heidelberg, Pages: 345-354
3. Victor Zamudio, Rosario Baltazar, Miguel Angel Casillas,Vic Callaghan "c-INPRES: Coupling Analysis Towards Locking Optimization in Ambient Intelligence". The 6th International Conference on Intelligent Environments IE10. 19-21 Julio 2010, Monash University (Sunway campus), Kuala Lumpur, Malaysia
4. Kennedy, J. and Eberhart, R. (1995). Particle swarm Optimization. In Proceedings of IEEE International Conference on Neural Networks, 1995., volume 4, pages 19421948 vol.4.
5. Rainer Storn and Kenneth Price. Differential evolution -a simple and efficient adaptative scheme for global optimization over continuous spaces. Technical Report TR-95- 12, International Computer Science, Berkeley, California, March 1995.
6. Rainer Storn and Kenneth Price. Differential evolution - a fast and efficient heuristic for global optimization over continuous spaces. Journal of Global Optimization, (11):341–359, 1997.
7. Dr. Carlos A. Coello Coello, Luis Vicente Santana Quintero. Un Algoritmo Basado en Evolución Diferencial para Resolver Problemas Multiobjetivo,Tesis Maestría, México DF, 2004.
8. Luis Vicente Santana Quintero y Carlos A. Coello Coello. Un Algoritmo Basado en Evolución Diferencial para Resolver Problemas Multiobjetivo.Tesis, CINVESTAV-IPN(Grupo de Computación Evolutiva).
9. Rubí del Carmen Gómez Ramón. Estudio empírico de variantes de Evolución Diferencial en optimización con restricciones. Tesis, Laboratorio Nacional de Informática Avanzada.
10. J. Kennedy, R. Eberhart, and Y. Shi. Swarm Intelligence. San Francisco: Morgan Kaufmann Publishers, 2001.
11. Victor Zamudio and V. Callaghan. Facilitating the ambient intelligent vision: A theorem, representation and solution for instability in rule-based multi-agent systems. Special Section on Agent Based System Challenges for Ubiquitous and Pervasive Computing. International Transactions on Systems Science and Applications., 4(2):108–121, May 2008.
12. Víctor Manuel Zamudio and Vic Callaghan. Understanding and Avoiding Interaction Based Instability in pervasive Computing Environmentes. Journal: International Journal of Pervasive Computing and Communications , vol. 5, no. 2, pp. 163-186, 2009
13. Leoncio Alberto Romero, Victor Zamudio, Rosario Baltazar, Aplicacion de Locking por Medio de Tecnicas de Inteligencia Artificial en Ambientes de Computo Pervasivo con Alta Inestabilidad, Tesis, Instituo Tecnologico de León 2012.
14. Leoncio Alberto Romero, Victor Zamudio, Rosario Baltazar and Marco Sotelo, "A Comparison Between PSO and MIMIC as Strategies for Minimizing Cyclic Instabilities in Ambient Intelligent", in the 5th International Symposium on Ubiquitous Computing and Ambient Intelligence (UCAmI´11), Rivera Maya, México, December 5-9, 2011.
15. V. Zamudio, V. Callaghan, "Preventing Instability in Rule-Based Multi-agent Systems; A Challenge to the Ambient Intelligence Vision". In workshop on Multi-agent Systems Challenges for Ubiquitous and Pervasive Computing MASUPC07 held at First International Conference on New Technologies, Mobility and Security (NTMS'2007), Telecom Paris, France, 2 to 4 May, 2007.

16. Wilcoxon, F.: Individual comparisons by ranking methods. Biometrics Bulletin 1(6) (1945) 80–83