Recibido: 7 de septiembre del 2012 Aceptado: 12 de enero del 2013
Publicado en línea: 25 de junio del 2013

Algoritmos PESO y DE Aplicados a la Minimización de la Inestabilidad Cíclica de Sistemas con Agentes 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, vic.zamudio, r.baltazar, miguel.casillas, masotelof}@ieee.org, carloslino@itleon.edu.mx

Resumen. En el presente trabajo se aborda el problema de inestabilidad en ambiente dinámicos, la cual se genera cuando un agente entra o sale del ambiente en un periodo determinado; con lo cual se pretende minimizarlo o en dado caso eliminarla, por tanto, existen algoritmos de optimización que nos permiten realizar esta función como: el algoritmos de Evolución Diferencial (Differential Evolution, DE) y el Algoritmo de Optimización mediante Evolución de Cúmulos de Partículas (Particle Evolutionary Swarm Optmization, PESO), los cuales fueron aplicados a las instancias de prueba en ambientes dinámicos y mediante la prueba de Wilcoxon podremos discernir que algoritmo obtuvo mejores resultados, en base a la menor inestabilidad calculada en el sistema.

Palabras clave: PESO, DE, Inestabilidad Cíclica, Agentes Nómadas, Topologías de Interconexión

1. Introducción

Los ambientes inteligentes integran la visión futura de las telecomunicaciones, electrónica de consumo e informática. En un mundo con ambientes inteligentes, los dispositivos van a tener como objetivo asistir a la gente en las tareas que debe hacer cada día, costumbres, actividades personales, etc., todo ello de una manera inteligente y no invasiva (oculto para el usuario). Para formar que todos los dispositivos estén conectados entre sí, creando de esta manera lo que se puede llamar: el internet de las cosas. Con un solo objetivo, que los dispositivos sean cada vez más inteligentes y más pequeños de forma que al final estén completamente integrados en el entorno, de manera que desaparezcan para nuestros ojos, dejando solo a nuestra percepción una interfaz del usuario. Uno de los retos en los entornos dinámicos inteligentes es controlar la inestabilidad que se desarrolla entre los agentes y evitar un comportamiento inestable en un período determinado, así como procesos que se ejecutan de forma indefinida, cuando se tienen esquemas de interacciones dinámicos, ya sea relativo a reglas dependientes del tiempo o agentes nómadas. Una de las primeras opciones propuestas es el algoritmo INPRES, basado en el uso de locking, que consiste en no permitir a un agente cambiar de estado, contradiciendo las reglas lógicas de programación para su comportamiento. Con estas estrategias, 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[1] 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 a implementar es el PESO[2] algoritmo para resolver un solo problema de optimización con restricciones objetivas. El algoritmo PESO propone dos nuevos operadores de perturbación: C-Perturbación y la M-Perturbación. El objetivo de estos operadores es luchar contra la convergencia prematura y los problemas observados en la diversidad del PSO puestas en la práctica. El Manejo de restricciones se basa en reglas simples de factibilidad. Por otra parte se encuentra lo que es algoritmo de Evolución Diferencial(DE)[3] [4] 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 la función de desempeño. En el caso de minimización, un individuo es mejor que otro cuando el primero tiene un menor valor en la función objetivo con respecto al segundo; de tal manera que se pueda reducir al mínimo las oscilaciones del sistema.

2. Algoritmo de Optimización mediante Evolución de Cúmulos de Partículas (PESO)

El algoritmo de Optimización mediante Evolución de Cúmulos de Partículas (Particle Evolutionary Swarm Optmization o PESO)[2], que se basa en el algoritmo enjambre propuesto originalmente en 1995 por Kennedy y Eberhart: Partículas Swarm Optimization (PSO)[5]. Este enfoque incluye el tratamiento de la restricción y el mecanismo de selección basado en las reglas de viabilidad, una organización de topología de anillo que lleva un registro de los mejores locales, y dos operadores de perturbación dirigidos a mantener la diversidad y la exploración. En este algoritmo se ha tomado nota de la compensación de velocidad y la diversidad de los algoritmos PSO [5]. Además, muchos señalaron la necesidad de mantener la diversidad de la población cuando el problema de diseño incluye limitaciones. Aunque los enfoques que se han propuesto varían en cuanto al trato de restricciones, los cuales se han mejorado con estos mecanismos para mantener la diversidad de la población y una mejor exploración del espacio de búsqueda. Sin embargo, una conclusión importante alcanzada recientemente en [6], es que el mecanismo de control de restricción que debe estar ligado al mecanismo de búsqueda, aún más, ligado a la forma en que los operadores de exploración del espacio de búsqueda de una posible solución. Este algoritmo se muestra en Algoritmo 1.

Algoritmo 1. Optmización mediante Evolución de Cúmulos de Partículas.

n: El número de particulas de la poblacion.
LI: Límite inferior.
LS: Límite superior.
c1: Coeficiente de la memoria.
c2: Coeficiente de la memoria del vecindario.
w: Coeficiente de adaptacion del vecindario.

```
For cada particula (i en n) do
   Inicializar la poblacion y buscar el mejor
local de la poblacion inicial.
   X = x1,x2,x3,...,xd |xi pertenece uniformemente distribuido entre [LI,LS]
   V = v1,v2,v3,...,vd |vi pertenece uniformemente distribuido entre [0,1]
   Lbest=X
endFor
Buscar el mejor Global
if Lbestx es mejor que Gbestx Then
   Gbest = Lbest
endif
while se cumpla la condicion de paro then
   for Cada particula i do
      Vi = wVi +c1(Gbest -Xi) + c2(Lbest- Xi)
      Xi= Xi + Vi
 if f(Xi) es menor que f(Lbest) then
         f(Lbest) = f(Xi)
      endif
endFor
Buscar el mejor Global
For cada particula i do
   if Lbest es mejor que Gbest then
      Gbest = Lbest
   endif
endfor
Temp = C-Perturbacion (Xi)
FTemp = Fitness (Temp)
for cada particula i do
   if FTemp es mejor que Gbest then
      Gbest = FTemp
   endif
endfor
Temp = M-Perturbacion (Xi)
```

```
FTemp = Fitness (Temp)
for Cada particula i do
   if FTemp es mejor que Gbest then
      Gbest = FTemp
   endif
endfor
end While
Return Gbest
```

Algoritmo 2. C-Perturbacion.

```
for cada particula i do
   r = U (1,0)
   p1 = Ramdon (n)
   P2 = Ramdon (n)
   p3 = Ramdon (n)
   Temp[i,j] = Pi+1[p1,j] + r(Pi+1[p2,j] -
Pi+1[p3,j])
endfor
```

En el Algoritmo 2, muestra una de las perturbaciones aplicadas en el algoritmo PESO, la cual es muy similar al operador de reproducción encontrado en al algoritmo de evolución diferencial; esta perturbación es aplicada a todo el conjunto de partículas denominada Temp y en el caso donde la partícula Temp resulte tener un mejor valor que la partícula mejor posicionada esta se sustituye para la misma.

Algoritmo 3. M-Perturbacion.

```
for cada particula i do
   r = U (1,0)
   if r<= 1/d then
      Temp[i,j] = Ramdon(LI,LS)
   else
      Temp [i,j] = Pi+1[i,j]
   endif
endfor
```

En el Algoritmo 3 la perturbación se realiza con una cierta probabilidad en cada dimensión del conjunto de partículas, y puede ser explicado como la adición de valores aleatorios a cada componente de las partículas. Esta perturbación es llamada M-pertubación y es aplicada a todo el conjunto de partículas, formando un conjunto de partículas Temp. Una vez más, como en la C-Perturbación, las partículas Temp son comparados y se conserva la de mejor valor. La perturbación se realiza con probabilidad p = 1 / d, donde d es la dimensión de la partícula.

3. 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:

xi,G; i = 0, 1, 2, ...,N – 1 (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 está 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 aleatoriamente 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 número 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
```

4. Cálculo de Inestabilidad Dinámica

Dentro de los errores de los ambientes inteligentes se encuentra la inestabilidad que se genera entre ellos, en la cual varios investigadores en ambientes inteligentes que se han encontrado con dicha inestabilidad; Entre los que se puede mencionar esta el investigador Víctor Zamudio[8], el cual nos explica varias propuestas en donde se topa con este comportamiento y explica varios casos en donde se producen comportamientos no deseados por los diseñadores de los ambientes inteligentes, de igual forma se ha visto que en otros campos en donde se tienen conectados varios dispositivos (agentes) interconectados entre sí pueden llegar a tener un comportamiento no deseado. Un ejemplo de esto es cuando un dispositivo A, está esperando una acción de un dispositivo B, sin la cual no puede continuar trabajando sin embargo el dispositivo B, generando el resultado de la acción que debe ejecutar el dispositivo A, generando un ciclo entre ambos dispositivos provocando que ninguno termine la tarea de 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[9],[10]. 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 la tarea para la que fueron programados.

Entonces para medir la oscilación de un sistema se proponen diferentes medidas de esta. En primera instancia se propone y utiliza la Oscilación Acumulativa Promedio (Ecuación 1) la cual trata de medir la rela-

ción de cambio entre el estado actual del sistema y el estado siguiente. Por otro lado se propone y 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. [11]

4.1 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):

$$O = \frac{\sum_{i=1}^{n-1} x_i}{n-1} \begin{cases} 1 \; si \; S_i \neq S_{i+1} \\ 0 \; en \; otro \; caso \end{cases} \qquad (2)$$

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 el Problema de Inestabilidad en Ambiente Inteligentes

Las pruebas que se realizaron ajo diferentes instancias de prueba donde cada instancia tuvo las siguientes características: se iniciaron con una matriz de 30x2, posteriormente se fue incrementando el tamaño y la dimensión, terminando con una matriz de 30x30, los agentes entrantes fueron en diferentes periodos, usando la misma topología; donde la función objetivo es el cálculo del promedio de cambios en el sistema donde se evalúan y se conserva quien tenga menos cambios en el sistema, con lo que se pretende que el sistema no oscile tanto o que no afecte tanto al sistema, para asegurar que el sistema conserve su propia integridad. En la topología usada en el sistema se muestra el siguiente comportamiento; donde se está iniciando con un número definido de nodos, que en este caso son 2, con dependencias del anterior estado para su interacción; dado que el sistema va incrementando, los nodos se van insertando de forma sucesiva; de igual manera se pueden formar diferentes topologías de conexión y no es necesario de forma lineal o sucesiva, 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 por ejemplo:

En diferentes casos las cadenas de los estados de los nodos no siempre van a ser 1's, en algunos de estos casos los nodos se van a incluir en estados inactivos para su posterior activación, esperando la interacción con otro nodo de la red.

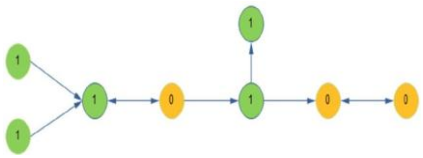

Fig. 1. Topología Usada en Ambientes Inteligentes

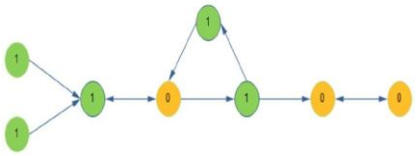

Fig. 2. Topología Usada en Ambientes Inteligentes con cambios de estados

La Figura 2 muestra el crecimiento iniciando con 2 agentes y terminando con 9, para este caso se llegara 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 para la comunicación con los ya establecidos en el sistema y con lo cual se pretende emular la entrada de agentes en un ambiente inteligente. En el gráfico anterior muestra la topología de intercomunicación que tendría el sistema, es decir 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, es decir, cuando el estado del agente es 1 la compuerta es AND y cuando es 0 la compuerta es OR[13], formando una cadena de AND y OR en el sistema y también formando los bucles dentro de él lo que genera oscilación, para el estudio la cantidad de bucles fueron 3. Los parámetros utilizados para los algoritmos DE y PESO se muestran en la Tabla 1 y 2.

Tabla 1. Parámetro del algoritmo PESO.

Parámetros	Valores
Individuos inicial	30
Dimensión inicial	2
Dimensión final	30
Llamadas a función	1000
% de agentes bloqueados	0.2
W	1
C1	0.4
C2	0.6

Tabla 2. Parámetro del algoritmo ED.

Parámetros	Valores
Individuos inicial	30
Dimensión inicial	2
Dimensión final	30
Llamadas a función	1000
% de agentes bloqueados	0.2
F	0.9
CR	0.8

6. Resultados

Los resultados son mostrados en las siguiente tabla y gráficos(Figura 3 y 4), los cuales permiten decir que el ambiente inteligente dinámico puede ser estabilizado en un periodo determinado con la condición de que los agentes puedan entrar en cierto tiempo al ambiente, iniciando con 2 agentes y terminando con 30 agentes en el ambiente y al evaluarlos nos permite encontrar una combinación de agentes que permita que el ambiente pueda quedar estabilizado o que la oscilación no afecte tanto al ambiente. En la Tabla 3 se muestran los resultados obtenidos en varias pruebas realiza-

das aplicadas a los algoritmos antes mencionados.

Tabla 3. Resultados.

Instancias	DE	PESO
Instancia 1	0.3831	0.3939
Instancia 2	0.3840	0.3985
Instancia 3	0.3737	0.4040
Instancia 4	0.3636	0.4041
Instancia 5	0.3787	0.4040
Instancia 6	0.3777	0.4044
Instancia 7	0.3643	0.3933

Los anteriores resultados fueron sometidos a la Prueba No Paramétrica de Rangos con Signo de Wilcoxon[14] 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 par 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 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.

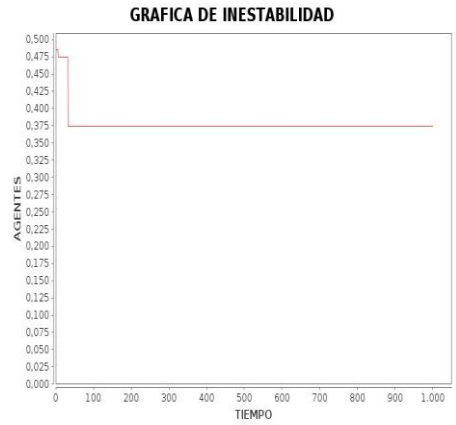

Fig. 3. Grafica DE

Fig. 4. Grafica PESO

7. Conclusiones y Trabajos Futuros

Dadas las pruebas obtenidas mediante los algoritmos DE y PESO a las instancias y con los resultados obtenidos se concluye que la inestabilidad que se genera en un escenario estático o dinámico puede ser controlada usando alguno de estos algoritmos. También de acuerdo a los resultados obtenidos no es posible determinar que algoritmo representa un mejor desempeño, debido a que no existe la suficiente evidencia estadística para comprobarlo. Teniendo en cuenta que los algoritmos evolutivos mejoran sus resultados de acuerdo a su parámetros de entrada. Sin embargo en que en base a los resultados mostrados anteriormente se puede afirmar que los algoritmos evolutivos representan una forma viable para resolver este tipo de problemas con las instancias anteriormente probadas.

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. V. Zamudio, R Baltazar y M Casillas, “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).
2. Angel E. Muñoz Zavala, Arturo Hernández Aguirre, Enrrique R. Villa Diharce. Particle Evolutionary Swarm Optmization Algorithm (PESO) In GECCO ’05: Proceedings of the 2005 conference on Genetic and evolutionary computation (2005), pp. 209-216.
3. 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.
4. 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.
5. 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
6. E. Mezura. Üso de la técnica Multiobjetivo NPGA para el Manejo de restricciones en Algoritmos Geneticos". PhD Thesis. Universidad Veracruzana, Xalapa, Ver 2001.
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. Víctor Manuel Zamudio. Understanding and Preventing Periodic Behavior in Ambient Intelligence. PhD thesis, University of Essex, October 2009.
9. V. 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.
10. Víctor Manuel Zamudio. Understanding and Preventing Periodic Behavior in Ambient Intelligence. PhD thesis, University of Essex, October 2009.
11. Leoncio Alberto Romero, Victor Zamudio, Rosario Baltazar, Aplica-

cion de Locking por Medio de Tecnicas de Inteligencia Artificial en Ambientes de Computo Pervasivo con Alta Inestabilidad, Tesis, Instituo Tecnologico de León 2012.

12. 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
13. V. Zamudio, V. Callaghan, "Preventing Instability in Rule-Based Multi-agent Systems; A Challenge to the Ambient Intelligence Vision". In workshop on Multiagent 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.
14. Wilcoxon, F.: Individual comparisons by ranking methods. Biometrics Bulletin 1(6) (1945) 80–83

Alejandro Sosa sales estudiante de la Maestría en Ciencias en Ciencias de la Computación en el Instituto Tecnológico de León, Ingeniero en Sistemas Computacionales por el Instituto Tecnológico de Lázaro Cárdenas. Áreas de interés son en el área de Ambientes Inteligentes.

Víctor Manuel Zamudio Rodríguez *profesor Investigador* de la División de Estudios de Posgrado e Investigación (DEPI) del *Instituto Tecnológico de León*, en el programa de Maestría en Ciencias en Ciencias de la Computación desde Octubre del 2009. Doctor en Ciencias de la Computación (Ph.D.) por la *Universidad de Essex*, Reino Unido (2009). Maestro en Ciencias de la Computación por el *ITESM* (1998). Físico Matemático por la *Facultad de Ciencias* de la UASLP. Es miembro del *Sistema Nacional de Investigadores* (nivel candidato) desde Enero del 2011, miembro del comité editorial del *International Journal of Ubiquitous Computing*. Ha sido Regional Area Advisory Chair (2010) del *International Conference on Intelligent Environments* IE’10, Workshop co-Chair (2011) y General Chair (2012) del mismo evento. Es miembro del *Institute of Electrical and Electronics Engineers* (IEEE) y de la *Association for Computing Machinery* (ACM). Ha sido becario del *Instituto de Investigaciones Eléctricas* (IIE) para realizar tesis de licenciatura (1994-1995), becario de la *Academia Mexicana de Ciencias* en el programa de Veranos de la Ciencia (1991, 1992) y Residencia de la Investigación (1998), becario del *CONACYT* para realizar estudios de maestría y doctorado. Seleccionado para aparecer en *Who is Who in the World*, 2010 Edition. Recibió la medalla *Los Mejores Estudiantes de México* (1995). Es facilitador acreditado de la

técnica didáctica *Proyect Oriented Learning*, habiendo sido capacitado en la Facultad de Ciencias e Ingeniería de la Universidad de Aalborg, en Dinamarca (2002). Ha presentado su trabajo de investigación en diversos congresos internacionales en *Alemania*, *Malasia*, *Reino Unido*, *Francia*, *Estados Unidos* y *México*. Sus áreas de investigación *son Inteligencia de Ambiente, Sistemas Multiagentes* y *Sistemas Complejos*. Para más detalles de su trayectoria y publicaciones favor de consultar www.vic-zamudio.org

María del Rosario Baltazar Flores *profesor Investigador* de la División de Estudios de Posgrado e Investigación (DEPI) del *Instituto Tecnológico de León*, en el programa de Maestría en Ciencias en Ciencias de la Computación desde Octubre del 2003. Doctor en Ciencias de la Computación (Ph.D.) por el Centro de Investigaciones en Óptica (2003). Ingeniera en Comunicaciones y Electrónica por la *Facultad de Ingeniería de la Universidad Autónoma de Zacatecas (1995)*. Es miembro del *Sistema Nacional de Investigadores* (nivel 1) a partir de Enero del 2012, es presidente del Consejo de Posgrado de la MCCC en el ITL. Es miembro de la academia de Sistemas Computacionales. Es miembro del comité organizador del *International Conference on Intelligent Environments* IE'12. Ha sido becario del *CONACYT* para realizar estudios de doctorado. Ha sido reconocida con la beca de profesor con perfil deseable de PROMEP.Fue miembro del Cuerpo Académico de Metrología y Sistemas Inteligentes y actualmente es líder del Cuerpo Académico Robótico y Ambientes Inteligentes. Ha publicado su trabajo de investigación en revistas internacionales indexadas de Holanda y Estados Unidos y en congresos en *Malasia*, *Reino Unido* y *México*. Sus áreas de investigación *son Inteligencia de Ambiente, Computo Móvil* y Comunicaciones en Robótica. Para más detalles de su trayectoria y publicaciones favor de consultar http://computomovil.wordpress.com/

Carlos Lino Ramírez *profesor Investigador* de la División de Estudios de Posgrado e Investigación (DEPI) del *Instituto Tecnológico de León*, en el programa de Maestría en Ciencias en Ciencias de la Computación desde Noviembre del 2010. Doctorado en Arquitectura y Tecnología de los Sistemas Informáticos por la *Universidad de Valencia*, España (2012). Maestro en Ciencias en Ciencias Computacionales por el *Instituto Tecnológico de León* (1999). Ingeniero en Sistemas Computacionales por el *Instituto Tecnológico de León (1996)*. Ha sido subdirector del *Instituto Tecnológico de León (2006-2007), jefe del departamento de Sistemas y Computación del Instituto Tecnológico de León (2004-2006), jefe del área de redes en el Sistema Avanzado de Bachillerato y Educación Superior (1999-2001).* Ha presentado sus trabajos de investigación en diversos congresos internacionales en *España*, *Italia*, *Alemania* y *México*. Sus áreas de

investigación *son Inteligencia de Ambiente, Algoritmos de Encaminamiento* y *Redes de Sensores Inalámbricas*.

Miguel Ángel Casillas Araiza profesor del Instituto Tecnológico de León, Ingeniero en Electrónica y Comunicaciones por la Universidad Iberoamericana y Maestro en Ciencias por el Centro de Investigaciones en Óptica. Su experiencia profesional incluye las áreas de Ingeniera Electrónica, Mecatrónica, Aprendizaje máquina, Programación de Dispositivos Embebidos y en el área de Robótica. Es presidente de la Academia Electrónica-Mecatrónica, del Instituto Tecnológico de León, es Secretario del Colegio de Ingenieros en Electrónica, Telecomunicaciones y Mecatrónica de León. Counselor de la rama de la IEEE campus II.

Marco Aurelio Sotelo Figueroa estudiante de Doctorado en Ciencias de Computación, Maestro en Ciencias en Ciencias de la Computación por el Instituto Tecnológico de León (2010). Primer lugar en el Concurso Nacional de Tesis de Posgrado (2011). Cuenta con más de 10 publicaciones de corte internacional, publicaciones en revistas de alto impacto y libros. Dentro de sus áreas de interés están: cómputo inteligente, optimización heurística, metaheurística e híper-heurística.