**Aplicación de algoritmos cGA y 𝜇PSO para reducir inestabilidades cíclicas en un entorno emulado**

Antonio Pachecoa, V. Zamudioa, M. R. Baltazara, C. Linoa, M. A. Casillasa

aInstituto Tecnológico de León, León, Gto., jpacheco@aiidia.com, vic.zamudio@ieee.org, charobalmx@yahoo.com.mx, carloslino@itleon.edu.mx, miguel.casillas@gmail.com

ResumEn:

En la actualidad gran parte de la población tiene a su alcance dispositivos electrónicos de gran capacidad de cómputo, bajo costo y en mayor cantidad. La interacción entre estos dispositivos y sus reglas de comunicación, pueden generar disturbios en la estabilidad del sistema, en el presente trabajo se aborda en específico una problemática que surge de dicha interacción, las inestabilidades cíclicas. Se han generado técnicas para reducir o intentar eliminar estas inestabilidades, una de estas técnicas es el Locking que en conjunto con algoritmos específicos de Inteligencia Artificial (Compact Genetic Algoritmh, micro-Particle Swarm Optimization) han logrado en simulaciones computacionales la reducción y en algunos casos, la desaparición de inestabilidades cíclicas, dejando al ambiente inteligente estable. Dentro del presente trabajo, se acercan más estos conceptos a la realidad cotidiana, pasando de simulaciones a emulaciones dentro de un micro ambiente inteligente, replicando los resultados previamente obtenidos.

1. IntroducciÓn

Uno de los principales retos que surgen de la alta intercomunicación de dispositivos en un ambiente inteligente es la inestabilidad cíclica, en donde un número de dispositivos o agentes muestran comportamientos periódicos o cíclicos, cambiando su estado periódicamente [1]. Una estrategia que ha sido desarrollada para reducir estas inestabilidades en AmI es el Locking [1]. Esta estrategia en conjunto con algoritmos bio-inspirados como el micro Particle Swarm Optimisation [2] y Genetic Algoritmh [3], los cuales han sido probados de manera exitosa para minimizar estos comportamientos cíclicos. Los microalgoritmos [4] son versiones que se acoplan a plataformas de recursos limitados ya que permiten trabajar con poblaciones de datos pequeñas lo cual los hace ideales para ser implementados en sistemas embebidos.

En esta tesitura, la inestabilidad cíclica es un reto que necesita soluciones y propuestas prácticas. En el análisis del estado del arte dentro de esta área, se encontró que las soluciones expuestas usan simulaciones computacionales, sin embargo, existe la necesidad de la implementación en sistemas embebidos. Uno de los temas principales de esta investigación, es la de explorar que tan factible es la implementación de estas soluciones usando hardware embebido en donde se ejecuten algoritmos bio-inspirados de optimización.

Ahora, la implementación de un ambiente inteligente, implicaría cablear todo un campus o equipar con sensores y actuadores los edificios, seria costoso en términos monetarios y de tiempo. Ejemplos como habitaciones inteligentes o laboratorios de ambientes inteligentes se han desarrollado en las universidades o centros de investigación, sin embargo, en [5] se presenta un nuevo enfoque para subsanar esta limitante, las cajas inteligentes (Smart Boxes), que son la unidad mínima para la construcción jerarquizada de un ambiente más extenso, por ejemplo,

un conjunto de cajas inteligentes nos proporcionan un edificio y un conjunto de estos últimos, una ciudad. La función de la caja inteligente depende de las sub-funciones provistas para la caja. La unidad minima, una sola caja inteligente, puede ser usada como una habitación inteligente, en donde los estudiantes e investigadores implementes sus conocimientos en un entorno sobre su escritorio.

2. Inestabilidad CICLica

La inestabilidad cíclica surge de las interacciones no deseadas entre agentes o dispositivos basados en reglas, sin embargo, su aparición no solo depende de esas interacciones, también es importante el estado inicial del sistema. De la teoría de sistemas complejos, se ha encontrado que no es posible predecir si un sistema multiagente basado en reglas tendrá inestabilidades cíclicas a lo largo de su ejecución, sin embargo, es posible prevenirlas [6]. Una de las estrategias que han sido probadas para prevenir estas inestabilidades, es la de Locking, que consiste en bloquear uno o más agentes, no permitiendo cambiar de su estado, en otras palabras, ignoran sus reglas internas de comportamiento en aras de mejorar la estabilidad general del sistema.

En [7], que es un enfoque basado en la optimización, se ofrecen nuevas herramientas para trabajar con este tipo de problemáticas en los ambientes inteligentes en donde la cantidad e interconexiones de los agentes se mantiene inalterada en el transcurso del análisis del ambiente inteligente, el autor define un par de funciones objetivo: Oscilaciones Acumulativas Promedio (1) en donde es el la representación logarítmica del vector de estado del sistema en el tiempo si se considera a como un numero en base numérica 10 representado de forma binaria y corresponde a la cantidad de ciclos de vida durante el cual se dejara evolucionar el sistema, y Promedio de Cambios en el Sistema (2) en donde es el estado del sistema en el tiempo y corresponde a la cantidad de ciclos de vida durante el cual se dejara evolucionar el sistema y toma el valor de “1” o “0” dependiendo de si el sistema vario respecto a su estado anterior o no.

|  |  |
| --- | --- |
|  | (1) |
|  |  |
|  | (2) |

Estas funciones fueron usadas junto con una serie de algoritmos de optimización metaheurística en donde se reporta que los algoritmos con mejores desempeños, en función al número de agentes bloqueados y minimización en las oscilaciones, fueron el 𝜇PSO y PSO.

En [8], Sosa et al. se investigó el rendimiento de algoritmos bio-inspirados en entornos dinámicos, es decir, en donde se tienen agentes nómadas. Las conexiones entre los agentes nómadas varían conforme evoluciona el sistema. Los agentes nómadas presentan estado, reglas y conexiones con otros agentes de manera aleatoria, en consecuencia, pueden contribuir a la formación de ciclos e incrementar la probabilidad de que se presente una inestabilidad cíclica. En la investigación referenciada, el algoritmo PSO y PESO fueron probados de manera satisfactoria para prevenir comportamientos oscilatorios.

3. BUzz box

La herramienta Buzz Box (Fig 1) forma parte de un sistema de prototipado rápido, basado en un conjunto de módulos electrónicos que permite a los estudiantes e investigadores crear de manera rápida un entorno inteligente. Las necesidades de los roles antes mencionados podría implicar que cada actor cuente con su propio sistema inteligente para realizar sus labores de aprendizaje o investigación, sin embargo, la implementación de ambientes reales para estos propósitos, sin contemplar a cantidad que se requiere de ellos, implican altos costos en infraestructura, tomando en cuenta estos requerimientos, la Buzz Box representa una solución factible mediante la deconstrucción de grandes entornos para la creación de un ambiente de escritorio. Los componentes clave en la Buzz Box son los Buzz-Panels, que se interconectan entre ellos mediante los conectores que no solo sirven para la comunicación eléctrica de las partes, sino que también son el soporte mecánico de la caja, de esta manera, se puede dar forma a los ambientes necesarios. Cada Buzz-Panel puede contener sensores y actuadores de acuerdo a las necesidades del trabajo que se esté desarrollando [5].



Fig 1. Buzz Box.

*Imagen tomada del sitio http://fortito.mx*

4. Implementación

Los algoritmos que se usaron para el presente trabajo, fueron el 𝜇PSO, que ya había sido probado en trabajos previos y el algoritmo Compact Genetic Algorithm [9], que es un algoritmo de estimado de probabilidad. La representación de las soluciones de estos algoritmos se da en forma vectorial, en donde cada componente del vector de solución, corresponde a un agente en el ambiente inteligente. Los valores que toman las componentes del vector solución, son “0” (cero) que significa que el agente debe comportarse de acuerdo a sus reglas, y “1” (uno) que indica que el agente correspondiente está bloqueado y en consecuencia mantendrá su estado sin importar lo que le dicten sus reglas. Los estados de los agentes solo pueden tomar dos valores, “encendido” y “apagado”.

En el presente trabajo se hizo uso de una Buzz Box con 5 Buzz-Panels: El panel superior cuenta con 8 leds tricolor y 8 botones de acción, el panel derecho es la base para el procesador, el panel izquierdo tiene dos juegos de leds con dos leds cada juego que pueden variar la intensidad de iluminación, el panel trasero sostiene un ventilador y un dispositivo de calefacción y por último, el panel inferior, que solo se usa para darle rigidez a la estructura.

El sistema embebido usa una Raspberry Pi Model B con un procesador de 700MHz de arquitectura ARM, 512MB de RAM, tiene 26 puertos GPIO para la comunicación con la Buzz Box además de dos puertos USB 2.0, un puerto de video HDMI y un puerto Ethernet 10/100, cuenta también con un slot para la tarjeta SD que servirá de almacenamiento en el sistema.

El vector de bloqueo, que es el resultado de la ejecución de los algoritmos, es un vector binario, el cual se mapeo con los agentes que se usaron dentro de la Buzz Box, es decir, se asignó un cada agente dentro del ambiente emulado a una componente del vector de bloqueo, de esta manera, al mantener el estado de la componente en “1”, el agente se bloqueará dejando su estado como el ultimo antes del bloqueo e ignorando las reglas programadas para el agente.

Las instancias usadas fueron creadas de manera aleatoria con 20 agentes cada una. Cada agente tiene está configurado como una compuerta AND o una compuerta OR. Se comprobó que cada una de estas instancias tuviera comportamientos cíclicos por medio de inspección del vector de estado del ambiente inteligente.

5. Resultados

Para el presente trabajo se implementaron dos algoritmos distintos para estabilizar un sistema inteligente multiagente.

|  |  |
| --- | --- |
| Parámetro | Valor |
| Partículas | 6 |
| *w* | 1 |
| c1 | 0.3 |
| c2 | 0.7 |
| Generación de reemplazo | 100 |
| Numero de partículas reiniciadas | 2 |
| Tasa de muta | 0.1 |

Tabla 1. Configuración del algoritmo 𝜇PSO

En la Tabla 1 se puede ver la configuración para el algoritmo 𝜇PSO, se usaron los parámetros reportados en el estado del arte.

|  |  |
| --- | --- |
| Parámetro | Valor |
| Población | 40 |
| Error permitido | 0.001 |

Tabla 2. Configuración del algoritmo cGA

En la Tabla 2, se muestra la configuración para el algoritmo cGA, la cantidad de individuos y el error permitido se seleccionaron de manera arbitraria.

|  |  |  |
| --- | --- | --- |
| Algoritmo | Agentes bloqueados promedio | Fitness promedio |
| 𝜇PSO | 75% | 10.01 |
| cGA | 70% | 10 |

Tabla 3. Resultados obtenidos.

En la Tabla 3 se muestran los resultados obtenidos. Para todos los experimentos se limitó la cantidad de llamadas a función a un máximo de 3000 y se penalizó la función objetivo cuando los resultados bloquean más del 20% del total de agentes. El comportamiento de los algoritmos con cada instancia es similar en función de la cantidad de agentes bloqueados, se tomo el promedio de todas las instancias. En todos los casos, la inestabildiad del sistema fue eliminada. Los experimentos consistieron en 100 corridas independientes de cada algoritmo con cada una de las instancias.

4. CONCLUSIONES

En el presente estudio, se mostró la factibilidad de la ejecución de algoritmos de optimización para el tratamiento de las inestabilidades cíclicas dentro de un ambiente inteligente, usando una herramienta de alta tecnología e innovadora la cual mostro ser versátil y adecuada para la experimentación. Los resultados obtenidos garantizan el control de la inestabilidad del sistema de manera efectiva. Se está trabajando para mejorar la estrategia, en particular minimizando la cantidad de agentes bloqueados.

4. agradecimientos

Agradecimientos al Consejo Nacional de Ciencia y Tecnología (CONACYT)  por el apoyo brindado a esta investigación y a la División de Estudios de Posgrado e Investigación del Instituto Tecnológico de León.

**BIBLIOGRAFÍA**

1. V. Zamudio, “Understanding and Preventing Periodic Behavior in Ambient Intelligence”, Ph. D. thesis, University of Essex. 2009
2. J. Kennedy and R. Eberhart, “Particle swarm optimization”, Proceedings of IEEE International Conference on Neural Networks, Perth, Australia, 1995, pg. 1942-1948.
3. D. E. Goldberg  Genetic Algorithms in Search, Optimization, and Machine Learning,  1989 :Addison-Wesley
4. A. Sosa, “Stabilization strategies in dynamic intelligent environments using artificial intelligence techniques”. 2013, MSc. thesis, Instituto Tecnológico de León, León, Gto, 2013
5. V. Callaghan “Putting the Buzz Back into Computer Science Education”, Workshop Proceedings of the 9th International Conference on Intelligent Environments, doi:10.3233/978-1-61499-286-8-454
6. V. Callaghan and V. Zamudio, “Faciliting the ambient intelligence vision: A theorem, representation and solution for instability in rule-based multi-agent systems. 2008, Special Section on Agent Based System Challenges for Ubiquitous and Pervasive Computing. International Transactions on System Science and Applications, May 2008, vol. 2 no. 4 pp 108-121
7. L. Romero, V. Zamudio, M. Sotelo, M. Baltazar and E. Mezura, “A comparation between mataheuristics as strategies for minimizing cyclic instability in ambient intelligence”, 2012, Sensor, vol. 12
8. A. Sosa, V. Zamudio, R. Baltazar, C. Lino, M. Casillas and M. Sotelo, “PSO and PESO algorithms applied to the problem of dynamic instability in multiagent nomads systems”, 2012, ROPEC International
9. G. Harik, F. Lobo and D. Goldberg, “The Compact Genetic Algorithm”, IEEE Transactions on Evolutionary Computation, vol. 3, no 4, pp 287-297. Nov. 1999