Presented at the Cloud-of-Things 2013, Athens, Greece, 16-17th July 2013

Genetic Algorithms and Differential Evolution Algorithms Applied to Cyclic Instability Problem in Intelligent Environments with Nomadics Agents

Alejandro Sosa [a,1], Víctor Zamudio, [a] Rosario Baltazar, [a] Vic Callaghan, [b] and Efren Mezura [c]

[a] *División de Estudios de Posgrado e Investigación, Instituto Tecnológico de León, Av. Tecnológico S/N, CP 37290 Guanajuato, México*

[b] *School of Computer Scienceand Electronic Engineering, University of Essex, Wivenhoe Park CO4 3SQ, UK*

[c] *Departamento de Inteligencia Artificial, Universidad Veracruzana, Sebastián Camacho #5, Centro, Xalapa, Veracruz, 91000, México.*

Abstract. In this paper the problem of cyclic instability in dynamic environments is presented. This cyclic instability is generated when binary rule-based nomadic agents (agents entering or leaving the environment) interact in complex ways, generating undesirable outputs for the final user. Our strategy is focused on minimizing this cyclic behaviour, using optimization algorithms, in particular Genetic and Differential Evolution Algorithms. These algorithms are applied to the Average Change Function. Different test instances were used to evaluate the performance of these algorithms. Additionally, statistical tests were applied to measure their performance.

Keywords. Nomadic Agents, Cyclic Instability, Genetic Algorithms, Differential Evolution

Introduction

Smart environments integrate the future vision of telecommunications, consumer electronics and computing. In a world of smart environments, the environment will be intended to assist people and provide customized services for monitoring, education, health, leisure, energy optimization, in a non-invasive way. Devices are getting smarter, smaller and cheaper, and they will be fully integrated to the environment, being able to communicate their states, and following the rules (learned or programmed) of the user. Cyclical instability is a fundamental problem characterized by the presence of unexpected oscillations caused by the interaction of the rules governing the agents involved [1] [2] [3] [4]. As mentioned before one of the challenges faced in intelligent environ-

[1] Alejandro Sosa, Víctor Zamudio; E-mail:alejandro_sosa@ieee.org, vic.zamudio@ieee.org

ments is to prevent this cyclic instability; in our case we are considering the case of nomadic agent, in particular new agents joining the environment. New agents join the environment randomly, connecting and interacting with the agents already present in the system.
An Instability Prevention System (INPRES) has been successfully applied to the problem of cyclic instability [1] [2] [3] This strategy is based on analyzing the interaction network associated (a digraph, where the vertex are the agents, and the edges represent the dependencies of the rules of the agents), finding the cycles and locking an agent for each cycle. One of the main disadvantages of this strategy is the computational cost, as in big systems with high interconnectivity finding the cycles or loops can be computationally expensive. In the case of dynamic scenarios (where the topology of the network is changing, growing with time), INPRES is not the best option, as for each new change in the topology (in our case, a new agent joining the system) the interaction network should be analysed. Due to these problems, in this paper we are applying an optimization approach, comparing the performance of Differential Evolution and Genetic Algorithm. These algorithms aims to minimize the average change to the system required to prevent instability by locking a set of agents. By using this approach, we avoid finding the loops in the interaction network of the system. [2] [3] [4] [5].

1. Internet/Cloud of things

The Internet of Things (IoT) is an idea based on which a layer of digital connectivity to existing things, where ”things” refers to all kinds of everyday objects, and even their components. This idea is expected to bring benefits in the short term, with application such as: mobile phones that open doors, sensors detecting leaks in pipes, billboards changing their ads according to the consumer profile of people passing through the street, small sensors measuring the temperature of a room or the traffic on the streets, and security cameras watching over the safety, and subway panels indicating the time remaining until the arrival of the next train[6].

Internet of Things is the kind of ubiquitous society where all people and all objects will be connected, and they will be identified and will be found. Everything will be connected to each other and exchange of information between objects and devices will become reality. In this world all objects and parts would be recorded, making it pracically impossible for an object to be lost[7] [8].
One of the problems facing these interactions are the instabilities that may be generated between the devices, generating undesirable behavior in the intelligent environment. In this paper we analyze this problem (considering nomdic agents leaving the environment) and propose a solution based on optimization algorithms (GA and DE).

2. Problem in Cases Real

In rcent years it has been found a numbers of cases showing unwanted cyclic behaviour, for example Robotics and manufacture, operating systems, telephony and emails [1] that have unwanted behavior as is the case of the Toyota Prius 2010 car which had faults

in the braking system [9], however, by the complexity of the system and the number of agents involved is extremely difficult to locate the agents that cause unwanted behavior.

Another problem caused by instability cyclic intelligent environments are, for example, in telephony, since it was not possible to have many devices connected together [1], another reported instability occurs with the software agent in this case by sending an email list because some are forming loops and causing instabilities in the system. Since it is very difficult to find when an occur oscillation strategies have been proposed to find the probability of occurrence of this instability, the probability of finding such instability can be calculated with:

$$P_{oscillations} = \frac{G(c, S_0)}{\prod_{i=1}^{N} \frac{2^{k_i} N!}{(n-k_i)!}} \tag{1}$$

where N is the number of nodes and k_i is the connectividad of node n. The number of sistem with oscillations, denoted by the function G depends on the cycles, denoted by c and the initial conditions denoted by S_0. The demonstration can be seen in [1].

3. Differential Evolution

Differential Evolution (DE) [10] [11]is an algorithm developed by Rainer Storn and Kenneth Price for continuous space optimization, applied to solving complex problems. Differential Evolution has a population of candidate solutions, which recombine and mutate to produce new individuals to be elected according to the value of the function performance. Differential Evolution is a parallel direct search method which utilizes NP D-dimensional parameter vectors

$$x_{i,G}, i = 1, 2, ..., NP \tag{2}$$

as a population for each generation G. NP does not change during the minimization process. The initial vector population is chosen randomly and should cover the entire parameter space. As a rule, we will assume a uniform distribution of probability for all random decisions unless otherwise stated. In case that a preliminary solution is available, the initial population might be generated by adding normally distributed random deviations to the nominal solution $x_{nom,0}$.DE generates new parameter vectors by adding the weighted difference between two population vectors to a third vector. Let this operation be called mutation. The mutated vectors parameters are then mixed with the parameters of another predetermined vector, the target vector, to yield the so-called trial vector. If the trial vector yields a lower cost function value than the target vector, the trial vector replaces the target vector in the following generation. This last operation is called selection. Each population vector has to serve once as the target vector so that NP competitions take place in one generation [12].More specifically DEs basic strategy can be described as follows:

- Initialization: an initial population is generated randomly with a distribution uniform [12].

- Mutation: randomly select three vectors that are different, subtract two of them and the differences are applied weight given to them by a factor and finally add the difference to the third vector difference [13].
- Recombination: Recombination is performed, taking each of the individuals in the population as the primary parent and other parents are randomly selected three generating a son. If the child has generated a value of the objective function better that the primary parent, then replaces it [12].
- Selection: All vectors are selected once as primary parent without depending on the objective function, checks whether the selected parent is better that their child preserved generated this otherwise its value is replaced by the child [13].

3.1. Binary Differential Evolution

The binary DE (binDE)[14] borrows concepts from the binary particle swarm optimizer (binPSO), developed by Keneedy and Eberhart, in similar way, the binDE uses the floating-point DE individuals to determine a probability for each component. These probabilities are he used to generate a bitstring solution form the floating-point vector. This bitstring is used by the fitness function to determine its qulity. The resulting fitness is then associated with the floating point representation of the individual. Let xi(t) represent a DE indivua, with aech xij(t)(j=1,,nx, where nx is the dimension of the binary-valued problem) a floating-point number. Then, the corresponding bitstring solution, yi(t), is calculate using:

$$y_{ij}(t) = \begin{cases} 1 & \text{if } U(0,1) < f(x_{ij}(t)) \\ 0 & \text{otherwise} \end{cases} \tag{3}$$

Where f is the sigmoid function

$$f(x) = \frac{1}{1+e^{-x}} \tag{4}$$

The fitness of the individual $x_i(t)$ is the simply the fitness obtained using the binary representation $y_i(t)$.

Algorithm 1 Binary Differential Evolution Algorithm.

```
Initialize a population and set control parameter values:
Repeat
Select parent x_i(t)
Select individuals for reproduction.
Produce one offspring x'(t)
y_i(t)=generated bitsting form x_i(t).
y'_i(t)=generates bitsting form x'_i(t).
if f(y'_i(t)) is better than f(y_i(t)) then
    Replace parent x_i(t) with offspring x'_i(t)
else
    Retain parent
end if
Until a convergence criterion is satisfied
```

4. Genetic Algorithm

Genetic Algorithm (GA) [15] is a search technique proposed by John Holland based on the theory of evolution by Darwin [15] [16][17]. This technique is based on the selection mechanisms that nature uses, according to which the fittest individuals in a population are those who survive, to adapt more easily to changes in their environment.
A fairly comprehensive definition of a Genetic Algorithm is proposed by John Koza [18]: It is a highly parallel mathematical algorithm that transforms a set of individual mathematical objects with respect to time using operations patterned according to the Darwinian principle of reproduction and survival of the fittest and after naturally have arisen from a series of genetic operations from which highlights the sexual recombination. Each of these mathematical objects is usually a string of characters (letters or numbers) of fixed length that fits the model of chains of chromosomes and is associated with a certain mathematical function that reflects their ability.

The GA seeks solutions in the space of a function through simple evolution. In general, the individual fitness of a population tends to reproduce and survive to the next generation, thus improving the next generation. Either way, inferior individuals can, with a certain probability, survive and reproduce. In Algorithm 3, a genetic algorithm is presented in a summary form [16].

Algorithm 2 Algorithm Genetic

```
Data: t (population size), G (maximum allowed function evaluations).
Result: Best Individual (Best Individual of last population).
P← Initialize-population(t) Generate (randomly) an initial population
Evaluate(P) Calculate the fitness of each individual

for g = 1 to G do
    P ← Select(P) Choose the best individuals in the population and pass them to the next generation
    P' ← Select(P) Choose the best individuals in the population and pass them to the next generation
    P'← Mutation(P") Mutate one individual of population randomly chosen
    Evaluate (P') Calculate the fitness of each individual of new population
    P← (P') Replace the old population with new population
end for
```

5. Using Optimization Algorithms to Solve the Problem of Cyclic Instability

In order to solve the problem of cyclic instability using optimization algorithms we need to minimize the amplitude of the oscillations. In the best-case scenario this would result in a stable system. Additionally we are interested on affecting the fewest number of agents (agents locked).
In order to measure the oscillatory behaviour of the system, two functions have been reported in the literature: Average Cumulative Oscillation (ACO)[16] and Average Change of the System (ACS)[19].
In this paper we use the Average Change of the System (ACS) function, which has been reported to be more accuracy to measure cyclic instability [19].

$$O = \frac{\Sigma_{i=1}^{n-1} x_i}{n-1} \begin{cases} 1 \text{ si } S_i \neq S_{i+1} \\ 0 \text{ en otro caso} \end{cases} \tag{5}$$

O: average change in system.
n : number of generations of scenario to test (total time of test).
with $S(t)$ being the state of the system in time t and $S(t+1)$ being the state of system in time $t+1$.
In the case of a stable system, this equation will show a flat line. Due to the previous, it is possible to use them as objective functions in a minimization algorithm.

6. Test Instance

Experiments were performed using test instance with following characteristics: they start with a 2x2 matrix of agents which was subsequently increased in size, ending with a 30x30 matrix of agents. In Figure 2 we can see the incidence matrix, increasing its size (as an example) from 2x2 to 4x4. Incoming agents were added at different times to the existing system, increasing the interaction network of the system.
The objective function is the weighted average of changes in the system which evaluates and retains the states of the agents that have fewer oscillations. It is intended that the system does not oscillate, affecting the system as little as possible, to ensure that it keep its own integrity and functionality to the end user.
A scenario where the system is no longer evolving shown in Figure 1, using the same topology and without applying any technique to minimize oscillations in the system.

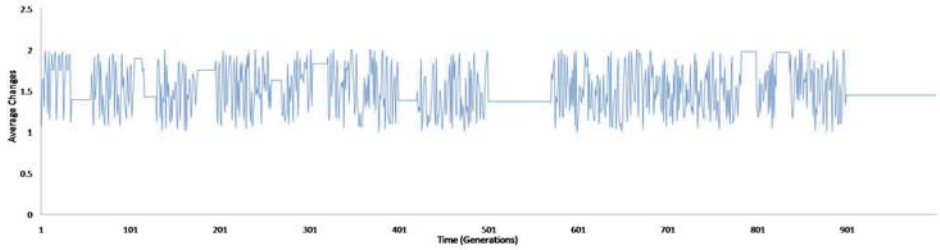

Figure 1. Graph Instability

0	0
1	0

0	0	1
1	0	0
1	1	0

0	0	1	0
1	0	0	0
1	1	0	1
1	0	0	0

Figure 2. Matrix of agents

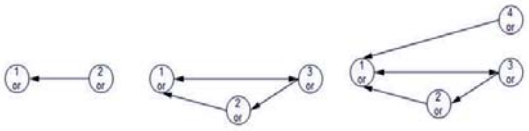

Figure 3. Graph of Agents associated to Figure 2.

In our case we will use the incidence matrix to represent of the digraph and manage the list of incidence, with the restriction that the main diagonal contains only 0's, in order to prevents the formation of loops on the same node.

As we mention before, we begin with a system with 2 agents (ie a 2x2 matrix), which will be increased randomly (as shown in Figure 2 and 3), ending with 30 agents, ie a 30x30 matrix (see Figure 4).

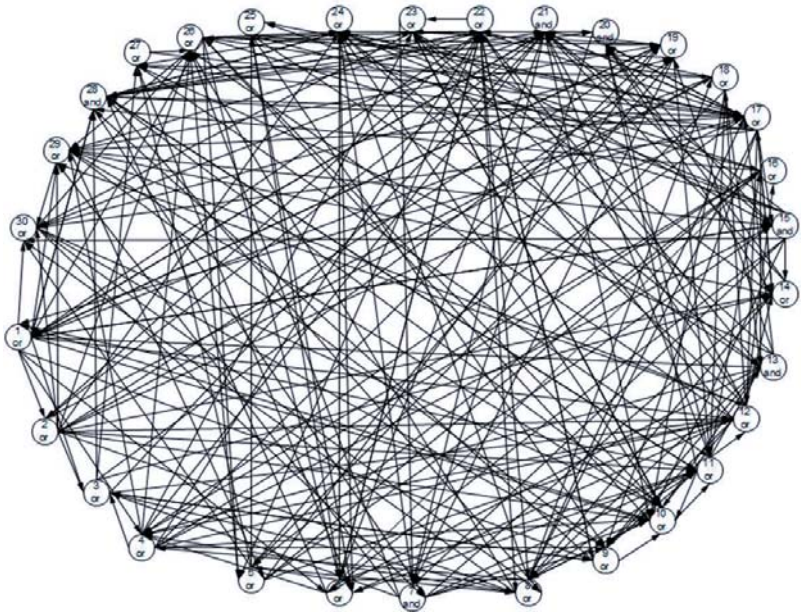

Figure 4. Digraph with 30 agents, associated to a 30x30 matrix.

To generate the test instance we used the following parameters

Parameters	Values
% of Connectivity	30
% of AND gate	40
% of Agents Locking	30
Dimension	30
Lifecycle	100

Table 1. of parametrs

7. Experimental Result.

For the test performed with DE and AG for test instances we used the parameters shown in Tables 2 and 3.

Parameters	Values
# Particles	30
W	1
C1	0.4
C2	0.6
Function Call	1000

Table 2. Parameters of PSO algorithm

Parameters	Values
# Particles	30
% of Elitism	55
% of Cross	35
% of Mute	29
Function Call	1000

Table 3. Parameters of GA

The results are shown in Table 4 and Figure 5 which allows say the dynamic intelligent ambient can be stabilized over a given period provided. From these results it can be seen that cyclic instability arising from dynamic environments (where the interaction network is randomly increased with time) can be stabilized successfully using the GA and DE algorithms.

	Median	Best
AG	1.01011339	1.01011339
DE	**1.00250249**	**1.00250249**

Table 4. Result

The above results were subjected to nonparametric signed ranks Wilcoxon test [20] to analyze the identity statistics regarding the results of the calculation of the average change in the system of algorithms, from which were obtained T + = 437, T = 27, with T0 = 109 for a sampling of 30 test with a significance level of 0.01. The results indicate that there is enough statistical evidence to establish which algorithm shows better performance; in our case Differential Evolution has better results.

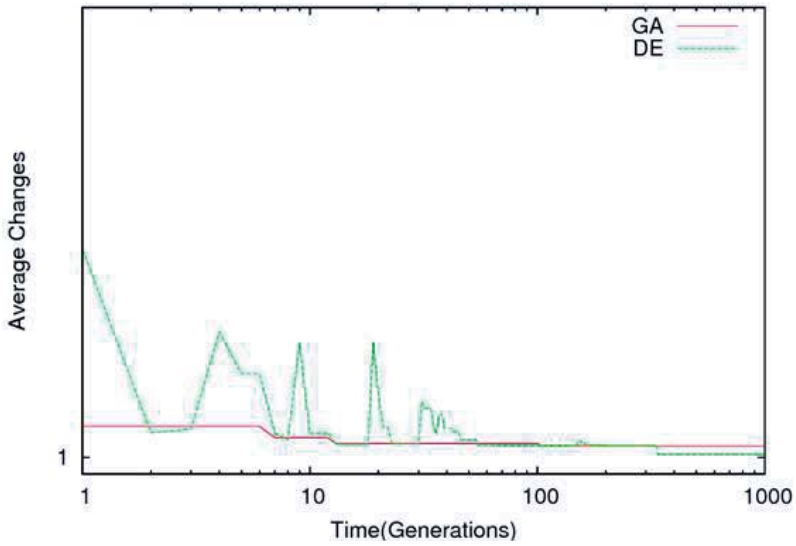

Figure 5. Graph Stability

8. Conclusions

In this paper we analysed the problem of cyclic instability with nomadic agents. In this particular case we are considering nomadic agents joining the environment, and interacting with those already present in the system. Two algorithms were considered: Differential Evolution and Genetic Algorithms. These algorithms were applied to the Average Change Function, which measured the number of changes in the state of the system in a given unit of time. The instability generated in these dynamic scenarios was successfully controlled using these algorithms. Using the Wilcoxon test (and due the fact there was enough statistical evidence) it was found that Differential Evolution algorithm had a better performance controlling these oscillations. These results are preliminary, but show very promising results. At the moment we are working on more complex test instances, including agents leaving the system (ie matrixes growing and shrinking), and a monitoring system to optimize the use of our strategy. We hope to report our result in future conferences.

References

[1] V. M. Zamudio, *Understanding and Preventing Periodic Behavior in Ambient Intelligence*. PhD thesis, University of Essex, Oct. 2009.

[2] V. Callaghan and V. Zamudio, ""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 Ubiquitoues and Pervasive Computing. International Transactions on System Science and Applications*, vol. 4, May 2008.

[3] V. Callaghan and V. Zamudio, "Understanding and Avoiding Interaction Based Instability in Pervasive Computing Environments," *International Journal of pervasive Computing and Communications*, vol. 5, pp. 163–186, 2009.

[4] V. Zamudio, R. Baltarzar, and M. Casillas, "c-INPRES: Coupling Analysis Towards Locking Optimization in Ambient Intelligence," *The 6th International Conference on Intelligent Environments IE10*, July 2010.

[5] A. Egerton, V. Zamudio, V. Callaghan, and G. Clarke, "Instability and Irrationality: Destructive and Constructive Services within Intelligent Environments," *Essex University: Southend-on-Sea, UK*, 2009.

[6] D. Ucklemann, M. Harrison, and F. Michehelles, eds., *Architecting the Internet of Things*. springer-verlag berlin heidelberg ed., 2011.

[7] F. Mattern and C. Floerkemeier, "From active data management to event-based systems and more," ch. From the internet of computers to the internet of things, pp. 242–259, Berlin, Heidelberg: Springer-Verlag, 2010.

[8] J. Gubbi, R. Buyya, S. Marusic, and M. Palaniswami, "Internet of Things (IoT): A Vision, Architectural Elements, and Future Directions," *CoRR*, pp. –1–1, 2012.

[9] "oyota. Prius, 2010. URL http://www.toyota.com/espanol/recall/abs.html.,"

[10] R. Storn and K. Price, "Differential evolution - a fast and efficient heuristic for global optimization over continuous spaces," *Journal of Global Optimization*, 1997.

[11] S. Rainer and K. Price, "Differential evolution -a simple and efficient adaptative scheme for global optimization over continuous space," *International Computer Science*, Mar. 1995.

[12] L. V. Santana Quintero and C. A. Coello Coello, "Un Algoritmo Basado en Evolución Diferencial para Resolver Problemas Multiobjetivo," Master's thesis, IPN, 204.

[13] R. d. C. Gomez Ramon, "Estudio empírico de variantes de Evolución Diferencial en optimización con restricciones," Master's thesis, Laboratorio Nacional de Informática Avanzada., 2001.

[14] A. Engelbrecht and G. Pampara, "Binary differential evolution strategies," pp. 1942–1947, 2007.

[15] J. Holland, *Adaptation in Natural and Artificial Systems*. University of Michigan Press, Ann Arbor., 1975.

[16] C. Houck, J. Joines, and M. Kay, "A Genetic Algorithm for Function Optimization: A Matlab Implementation," *Technical Report NCSU-IE-TR-95-09*, 1995.

[17] C. Coello Coello, "Introducción a la Computación Evolutiva," *Available online: http://delta.cs.cinvestav.mx/ ccoello/genetic.html*, 2012.

[18] J. Koza, "Genetic Programming: On the Programming of Computers by Means of Natural Selection," *MIT Press: Cambridge, MA, USA*, 1992.

[19] L. A. Romero, V. Zamudio, M. Sotelo, R. Baltarzar, and E. Mezura, "A Comparison between Metaheuristics as Strategies for Minimizing Cyclic Instability in Ambient Intelligence," *Sensors 2012*, 2012.

[20] F. Wilcoxon, "Individual comparisons by ranking methods ," *Biometrics Bulletin*, vol. 1(6) (1945) 80–83.