Workshop Proceedings of the 8th International Conference on Intelligent Environments
J.A. Botía et al. (Eds.)
IOS Press, 2012
© 2012 The authors and IOS Press. All rights reserved.
doi:10.3233/978-1-61499-080-2-130

A Comparative Study of Intelligent Bio-inspired Algorithms Applied to Minimizing Cyclic Instability in Intelligent Environments

Leoncio ROMERO [a,1], Victor ZAMUDIO [a,2], Rosario BALTAZAR [a], Marco SOTELO [a], Carlos LINO [a], Efren MEZURA [b] and Vic CALLAGHAN [c]

[a] *Division of Research and Postgraduate Studies, Leon Institute of Technology, Leon, Guanajuato, Mexico*
[b] *Laboratorio Nacional de Informatica Avanzada, Xalapa, Veracruz, Mexico*
[c] *School of Computer Science and Electronic Engineering, University of Essex, Wivenhoe Park, United Kingdom*

Abstract. Cyclic instability is a problem that, despite being shown to affect intelligent environments and in general any rule-based system, the strategies available to prevent it are still limited and mostly focused on centralized approaches. These approaches are based on topological properties of the ***Interaction Network*** (IN) associated, and locking a set of agents. In this paper we present a comparative study of the performance of different optimization techniques when solving the problem of cyclic instability in synthetic scenarios. Instead of using the Interaction Network of the System (which can be computationally expensive, specially in very dense systems). We introduced the concept of ***Average Change of the System*** (ACS) in order to measure the oscillatory behavior of the system and an optimization strategy. In particular Particle Swarm Optimization (PSO), Micro-Particle Swarm Optimization (μ-PSO), Bee Swarm Optimization (BSO), Artificial Immune Systems (AIS) and Genetic Algorithms (GA) were considered. The results found are very promising, as they can successfully prevent unwanted oscillations. Additionally, some of these strategies could be implemented using parallel and distributed processing, in order to be used in real-time scenarios.

Keywords. Cyclic Instability, Ambient Intelligence, Locking

Introduction

Intelligent Environments are systems that are affected by errors, as any other computer system. Among the problems affecting rule-based multiagent ambient intelligence sce-

[1]Corresponding Author: Leoncio Romero, Avenida Tecnologico S/N Fracc. Industrial Julian de Obregon, Leon, Gto., Mexico; E-mail: leoncior@acm.org

[2]Corresponding Author: Victor Zamudio, Avenida Tecnologico S/N Fracc. Industrial Julian de Obregon, Leon, Gto., Mexico; E-mail: vic.zamudio@ieee.org

narios we find cyclical instability [1]. This behavior is characterized by the presence of unexpected fluctuations caused by the interaction of the rules governing the different agents [1].

In the literature there are several approaches to the problem of cyclic instability [2], [1], showing good results. However, these approaches require a large number of calculations, therefore the possibility to be applied to real-time scenarios are very limited. Bio-inspired algorithms have been successfully applied to the problem of cyclic instability, using the ***Game of Life as a scenario*** [3].

In this paper we present two functions to measure the oscillatory behavior of the system. Several optimization techniques were applied to these functions, in order to compare their performance preventing oscillatory behavior. Among the optimization techniques used we can mention *Particle Swarm Optimization* (PSO), *Micro Particle Swarm Optimization* (μ-PSO), *Bee Swarm Optimization* (BSO), *Immune Artificial Systems* (AIS), and *Genetic Algorithms* (GA). These algorithms were applied to well known scenarios where the strategy INPRES has successfully prevented oscillatory behavior [1]. In our experiments the number of locked agents was used to measure the performance of the algorithms applied. The strategy with the minimum number of agents locked was considered the best.

1. Cyclic Instability in Intelligent Environments

Rule-based systems play an important role in Ambient Intelligence. In particular, multi-agent systems can provide different functionalities to the final user, generating complex interactions between a large number of connections in the system. Under some conditions (in particular the presence of feedback in the rules), unwanted oscillations of the agents can affect the expected performance of the system. These changes over time can even cause interference with other devices or unwanted behavior [1,4–6].

The state of the system $s(t)$ (see eq. 1 is defined as the logarithm base 10 of the decimal representation of the binary state of the agents involved. An agent can have two states on (1) and off (0).

$$s(t) = \log(s) \tag{1}$$

where:
$s(t)$ is the state of the system at time t.
s is base-10 representation of the binary vector of the agents.

2. Measuring the cyclical instability

In order to measure the oscillatory behaviour of the system, two functions are used: Average Cumulative Oscillation (ACO) and Average Change of the System (ACS).

Average Cumulative Oscillation (ACO) [3] measure the difference between $s(t)$ and $s(t+1)$ on the assumption that if a system is unstable difference will always exist among the states which will cause the value of the ACO will increase as increases the number of generations of the system, however if the system is stable this value will tend to 0.

$$o = \frac{\sum_{t=1}^{n-1} |S(t) - S(t+1)|}{n-1} \tag{2}$$

where:
o: average cumulative oscillation
n : number of generations (total time of testing)
$S(t)$: state of system at time t
$S(t+1)$: tate of system at time $t+1$

In this paper we introduce another function, called Average Change of the System (ACS). This equation means that an unstable system will remain constantly changing ie. the state $s(t)$) is always different from the state $s(t+1)$ this implies that if a system is unstable the value of ACS obtained is close to 1, while if the system is stable this value is approximated to 0.

$$p = \frac{\sum_{i=1}^{n-1} x_i}{n-1} \tag{3}$$

where:
p: average change in system
n : number of generations of scenario to test (total time of test)
$x_i = \begin{cases} 1 \; si \; S(t) \neq S(t+1) \\ \quad 0 \; in \; other \; case \end{cases}$

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, these two equations will show a flat line. Due to the previous, it is possible to use them as objective functions in a minimization algorithm.

3. Optimization Algorithms

3.1. Particle Swarm Optimization

Particle Swarm Optimization (PSO) [7, 8] algorithm was proposed by Kennedy and Everhart. It is based on the choreography of a flock of birds [7–12]. The basic PSO algorithm [8] uses two equations. The first one 4 finds the velocity, describes the size and direction of the step that will be taken by the particles and is based on the knowledge achieved until that moment.

$$v_i = wv_i + c_1 r_1 (lBest_i - x_i) + c_2 r_2 (gBest - x_i) \tag{4}$$

where:
v_i is the velocity of the i-th particle.

$i = 1, 2 \ldots, N$ and N is the number of the population.
w is the environment adjustment factor
c_1 is the memory factor of neighborhood
c_2 is memory factor
r_1 and r_2 are random numbers in range $[0, 1]$
$lBest$ is the best local particle founded for the i-th particle
$gBest$ is the best general particle founded until that moment for all particles
The equation 5 updates the current position of the particle to the new position using the result of the velocity equation.

$$x_i = x_i + v_i \tag{5}$$

where x_i is the position of the i-th particle.

3.2. Binary PSO

Binary PSO [12, 13] was design to work in binary spaces. Binary PSO select the $lBest$ and $gBest$ particles in the same way as PSO. The main difference between binary PSO and normal PSO are the equations that are used to update the particle velocity and position. The equation for updating the velocity is based on probabilities in the range $[0, 1]$. For that mapping is established for all real values of velocity to the range $[0, 1]$. The normalization function 6 used is a sigmoid funcion.

$$v_{ij}(t) = sigmoid\left(v_{ij}(t)\right) = \frac{1}{1 + e^{-v_{ij}(t)}} \tag{6}$$

and equation 7 is used to update the new particle position.

$$x_{ij}(t+1) = \begin{cases} 1\, if\; r_{ij} < sigmoid\left(v_{ij}(t+1)\right) \\ 0\; in\; other\; case \end{cases} \tag{7}$$

where r_{ij} is a random vector with uniform values in the range $[0, 1]$.

3.3. Micro PSO

Micro-PSO (μ-PSO) algorithm [14, 15] is a modification made to the original PSO algorithm in order to work with small populations. PSO and μ-PSO are very similar, but μ-PSO has more exploratory power. In order to avoid local optimum (exploring the configuration space), μ-PSO includes the concepts operators of replacement and mutation [15, 16].

3.4. Bee Swarm Optimization

This algorithm is based on PSO and Bee algorithm [13] and uses a local search step to to improve the performance. This algorithm was proposed by Sotelo [13]. The local search is made around $gBest$ in each iteration of the algorithm.

Another variant of the algorithms consists in applying the local search around $lBest$ after comparing it with a bee.

In the future we will refer to BSO algorithm with the $gBest$ enchancer as BSO1 while BSO algorithm with the $lBest$ enhancer will be known as BSO2.

3.5. Artificial Immune System

The Artificial Immune System (AIS) [17] is a metaheuristic based on the Immune System behavior of living things [13], particularly of mammals [17].

One of the main functions of the immune system is to keep the body healthy. A variety of microorganisms (called pathogens) could invade the body, which could be harmful. Antigens are molecules that are expressed on the surface of pathogens that can be recognized by the immune system and are also able to initiate the immune response to eliminate them [17].

Artificial immune systems have various types of models, in this work we use the one that implements the clonal selection algorithm that emulates the process by which the immune system in the presence of a specific antigen, stimulates only those lymphocytes that are more similar, then they are cloned and mutated [17].

3.6. Genetic Algorithm

Genetic algorithms (GAs) [18] proposed by John Hollan based on the theory of evolution by Darwin [18–20]. 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 [21]:

"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.

3.6.1. Clones and Scouts

In order to increase the performance of the GAs the concept of clones and explorers is considered [22]. A clone is an individual whose fitness is equal to the best individual fitness. When it reaches a certain percentage of clones a percentage of the worst individuals in the population is then mutated. Mutated individuals are named scouts. The application of clones and explorers is in addition to the mutation carried out by the GA generation.

4. Experimental Results

In order to test the performance of the previous algorithms 5 diferent well known topologies were used: 1) iDorm [1], 2) Strong coupling [6], 3) Week coupling [6] 4) non-coupled cycles [1], and 5) cycles coupled in two points [1]. In iDorm topology the maximum allowed percentage of locked agents was 30% while in other was 20%.

If a solution generated by any of the algorithms was good with respect to the metric used but the percentage of locked agents exceeded the maximum allowable then the solution is penalized by increasing the value of the metric obtained by a constant value.

In our experiments we set as a parameter, 3000 functions calls as a measure of success of the algorithms i.e. the system has 3000 opportunities to find a better solution. If after 3000 functions calls a better solution is not found, the system has failed.

The best solution not only minimizes the value of the metric used but also minimizes the number of locked agents. In the experiments the percentage of agents that can be blocked is set also as a parameter. This is important because if this percentage grows the system can be disabled.

Interaction Networks test instances [1] are showed on Table 1.

Table 1. Benchmark

Instance	# of Agents	ACO	ACS
1 (iDorm)	4	0.07183234371149662	1.0
2 (Strong coupling)	7	0.26938367763858284	1.0
3 (Week coupling)	7	0.1362409906735542	1.0
4 (Non-coupled)	64	0.35847554949972144	1.0
5 (Coupled in 2 points)	64	1.1840427653926249	1.0

A summary of the parameters used in our experiments is shown in Table 2.

Based on the previous parameters Tables 3 and 4 shows the results obtained in the case of the function ACO, for each algorithm.

Table 5 shows the results obtained when using the function ACS.

All algorithms were tested using well know topologies and rules, and where the strategy INPRES had been previously successfully applied [1, 6]. In order to have a fair comparison, we considered only the number of agents locked (as is the only parameter considered by INPRES).

Table 6 shows the number of locked agents, when the ACO is applied (see Tables 3 and 4.

Table 7 shows the number of locked agents corresponding to the results obtained with the ACS (shown in Table 5).

Table 2. Algorithms Parameters

Algorithm	Parameter	Value
PSO BSO	Particles	45
	w	1
	c_1	0.3
	c_2	0.7
μ-PSO	Particles	6
	w	1
	c_1	0.3
	c_2	0.7
	Replacement generation	100
	Number of restart particles	2
	Mutation Rate	0.1
AIS	Antibodies	45
	Antibodies to select	20
	New Antibodies	20
	Beta Factor	2
GA	Chromosomes	30
	Mutation percentage	0.15
	Elitism	0.2
	Clones percentage	0.3
	Scouts percentage	0.8

Table 3. ACO Results (A)

Instance	Average Cumulative Oscillation		
	PSO	BSO1	BSO2
1 (iDorm)	8.628730269198046E-4	0.0	0.0
2 (Strong coupling)	0.0032242818868545857	0.0	0.0
3 (Week coupling)	0.003153206791536531	0.0	0.0
4 (Non-coupled)	0.002508031766729257	3.8389318144437694E-4	7.7867850781205E-9
5 (Coupled in 2 points)	3.022378356262937E-4	5.27630744376312E-15	0.0

In order to show how the oscillations are succesfully removed, in Figures 1 to 5 the evolution of the system is shown. In figure 1a the oscillatory behavior of instance 1 (iDorm) is showed and in figure 1b the instabilities are successfully removed. For the instance 2 (Strong coupling) the evolution of the system is shown in figure 2. For the instance 3 (Weak coupling) the behavior is whown Figure 3. In figure 4a the oscillatory behavior of the instance 4 (Non-coupled) is showed, and behavior without oscillation is showed in figure 4b. The oscillatory behavior of the instance 5 (Coupled in 2 points) is showed in figure 5a, and behavior without oscillation is showed in figure 5b.

Table 8 shows the results obtained by the algorithms considered in this paper. In order to determine whether an algorithm outperforms another one, the Wilcoxon test was applied for both the number of agents locked and the oscillatory functions ACO and ACS.

Table 4. ACO Results (B)

Instancia	Average Cumulative Oscillation		
	AIS	**GA**	**μPSO**
1 (iDorm)	0.0	0.0173	0.0
2 (Strong coupling)	0.0	0.0	0.0
3 (Weak coupling)	0.0	0.00253	0.0
4 (Non-coupled)	7.63419134025501E-6	0.0	0.008354842172152571
5 (Coupled in 2 points)	0.0	0.0	3.2806731364235696E-4

Table 5. ACS Results

Instance	Average Change of the System					
	PSO	**BSO1**	**BSO2**	**AIS**	**GA**	**μ-PSO**
1 (iDorm)	0.0	0.0	0.0	0.0	0.0	0.0
2 (Strong coupling)	0.02	0.0	0.0	0.0	0.0	0.0
3 (Weak coupling)	0.03	0.0	0.0	0.0	0.0	0.0
4 (Non-coupled)	0.03	0.0	0.0	0.03	0.0	0.03
5 (Coupled in 2 points)	0.07	0.0	0.0	0.0	0.0	0.07

Table 6. ACO Locked Agents

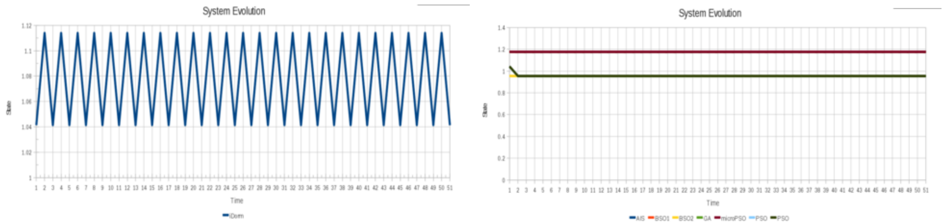
Instance	Number of Locked Agents							
	Permited	**INPRESS**	**PSO**	**BSO1**	**BSO2**	**AIS**	**GA**	**μ-PSO**
1 (iDorm)	1	1	1	1	1	1	1	1
2 (Strong coupling)	1	3	1	1	1	1	1	1
3 (Weak coupling)	1	1	1	1	1	1	1	1
4 (Non-coupled)	12	16	6	9	12	6	5	7
5 (Coupled in 2 points)	12	28	7	12	9	9	5	5

Table 7. ACS Locked Agents

Instance	Number of locked agents							
	Permited	**INPRESS**	**PSO**	**BSO1**	**BSO2**	**AIS**	**GA**	**μ-PSO**
1 (iDorm)	1	1	1	1	1	1	1	1
2 (Strong coupling)	1	3	1	1	1	1	1	1
3 (Weak coupling)	1	1	1	1	1	1	1	1
4 (Non-coupled)	12	16	9	12	12	9	3	12
5 (Coupled in 2 points)	12	28	5	9	7	6	2	6

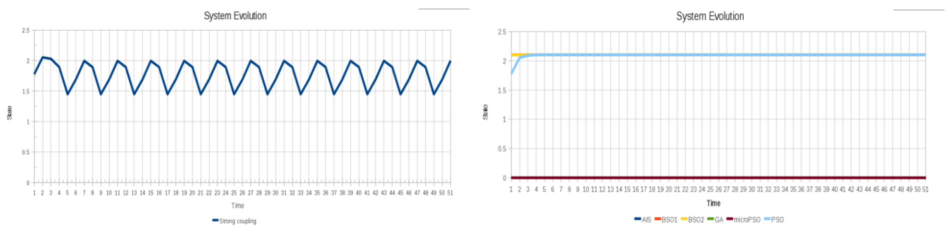
From Table 8 it was found that GAs were able to obtain smaller for ACO and ACS but if we take into account the number of locked agents the algorithms PSO and μ-PSO were the best.

All the algorithms used were able to prevent cyclic behaviour and showed low values (good performance) in the different parameters involved. However the most important parameter is the number of locked agents. In this case, μ-PSO showed the best performance in the experiments performed.



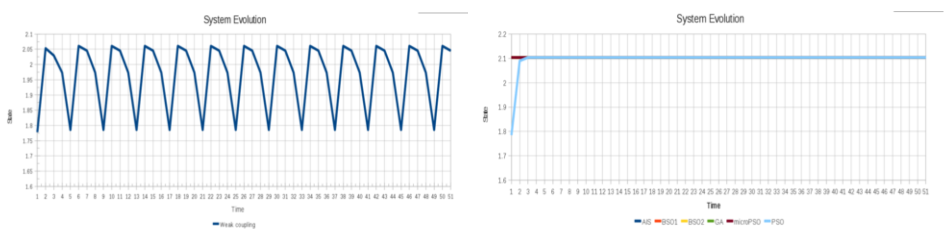
(a) Oscillatory behavior of the system (b) Instabilities are successfully removed

Figure 1. Evolution of the system for the case of 1 (iDorm)



(a) Oscillatory behavior of the system (b) Instabilities are successfully removed

Figure 2. Evolution of the system for the case of 2 (Strong coupling)

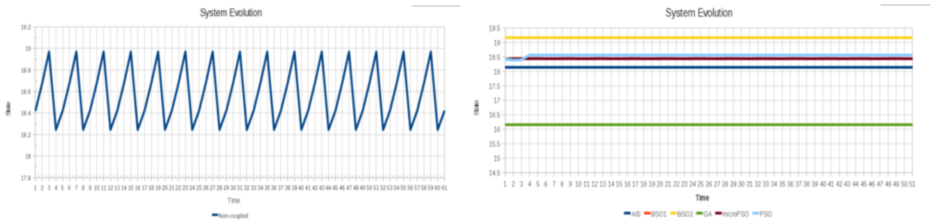


(a) Oscillatory behavior of the system (b) Instabilities are successfully removed

Figure 3. Evolution of the system for the case of 3 (Weak coupling)

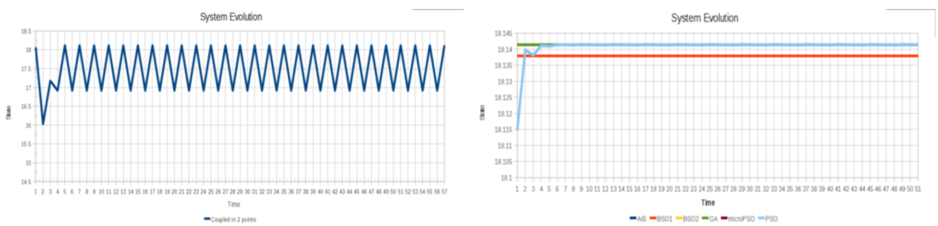
5. Conclusions and Future Work

From our experiments we found that bio-inspired algorithms can successfully minimize the oscillations when applied to different well-known test instances. These results are very encouraging, as bio-inspired algorithms could be applied to real-time scenarios where rules of interactions could be changing on time, and where the agents could be nomadic (with the corresponding changes of the rules). Additionally, this approach prevents having a large number of agents locked, disabling the system. The metrics used in our experiments *Average Cumulative Oscillations* ACO and *Average Change of the System* ACS showed good results. However it was found that for ACO when the relation of change between states is very small the system can have oscillation values close to 0 or significantly below than the original oscillation value and cyclic instability was still



(a) Oscillatory behavior of the system (b) Instabilities are successfully removed

Figure 4. Evolution of the system for the case of 4 (Non-coupled)



(a) Oscillatory behavior of the system (b) Instabilities are successfully removed

Figure 5. Evolution of the system for the case of 5 (Coupled in 2 points)

Table 8. Algorithms Compared Using the Wilcoxon Test

Algorithm	Number of Algorithms (Metric AAO)				Number of Algorithms (Metric ACS)			
	by Value		by # of Agents		by Value		by # of Agents	
	Overcome	Not Overcome	Overcome	Not Overcome	Overcome	Not Overcome	Overcome	Not Overcome
PSO	0	5	4	1	1	4	5	0
BSO (1)	1	4	2	3	3	2	1	4
BSO (2)	2	3	2	3	4	1	2	3
μPSO	4	1	5	0	0	5	4	1
AIS	4	1	3	2	2	3	3	2
GA	5	0	0	5	5	0	0	5

present. In the case of ACS we found that if you have oscillations close to 1 the instability is present, but if the oscillations are equal or close to 0 the system is stable.

From the experiments performed it was found that PSO and μ-PSO are the most likely to be implemented in real-time scenarios. Additionally, a parallel implementation of these algorithms could improve the results obtained in these experiments. More research is needed in these directions, in particular more complex scenarios and dynamic conditions (rules and nomadic agents), and additional optimization techniques. We hope to report our results in future publications.

Acknowledgments

The authors want to thank Jorge Soria for their comments and suggestions to his work. Leoncio Romero acknowledges the support of the National Council for Science and Technology CONACyT. Additionally, Efren Mezura acknowledges the support from CONACyT through project No. 79809.

References

[1] Victor Manuel Zamudio. *Understanding and Preventing Periodic Behavior in Ambient Intelligence*. PhD thesis, University of Essex, October 2008.

[2] Jaafar Gaber. Action selection algorithms for autonomous system in pervasive environment: A computational approach. *ACM Transactions on Autonomous and Adaptative Systems*, 6, February 2011.

[3] Leoncio Alberto Romero, Victor Zamudio, Rosario Baltazar, Marco Sotelo, and Vic Callaghan. A comparison between pso and mimic as strategies for minimizing cyclic instabilities in ambient intelligence. In *5th. International Symposium on Ubiquitous Computing and Ambient Intelligence UCAmI*, 2011.

[4] 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.

[5] Victor Zamudio and Vic Callaghan. Understanding and avoiding interaction based instability in pervasive computing environments. *International Journal of Pervasive Computing and Communications*, 5:163–186, 2009.

[6] V. Zamudio, R. Baltazar, and M. Casillas. c-INPRES: Coupling analysis towards locking optimization in ambient intelligence. In *The 6th International Conference on Intelligent Environments IE10*, Monash University (Sunway campus), Kuala Lumpur, Malaysia., July 2010.

[7] Anthony Carlise and Gerry Dozier. An off-the-shelf pso. 2001.

[8] Russel C. Eberhart and Yuhui Shi. Particle swarm optimization: Developments, applications and resources. *IEEE*, pages 82–86, 2001.

[9] Carlos A. Coello and Maximo Salazar. Mopso: A proposal for multiple objetive particle swarm optimization. *Evolutionary Computation, IEEE*, pages 1051–1056, 2002.

[10] Swagatam Das, Amit Konar, and Uday K. Chakraborty. Improving particle swarm optimization with differentially perturbed velocity. *GECCO*, pages 177–184, 2005.

[11] K.E. Parsopoulos and M. N. Vrahatis. Initializing the particle swarm optimizer using the nonlinear simplex method. *GECCO*, 2005.

[12] Tamer M. Khali, Hosam K. M. Youssef, and M. M. Abdel Aziz. A binary particle swarm optimization for optimal placement and sising of capacitor banks in radial distribution feeders with distored substation voltages. *AIML 06 International Conference*, Jun 2006.

[13] Marco A. Sotelo. Aplicacion de metaheuristicas en el knapsack problem. Master's thesis, Leon Institute of Technology, Guanajuato, 2010.

[14] Juan C. Fuentes Cabrera and Carlos A. Coello Coello. Handling constraints in particle swarm optimization using a small population size. In *6th. Mexican International Conference on Artificial Intelligence*, Aguascalientes, Mexico, November 2007.

[15] Francisco Viveros Jiménez, Efrén Mezura Montes, and Alexander Gelbukh. Empirical analysis of a micro-evolutionary algorithm for numerical optimization. 2011.

[16] Efrén Mezura Montes, Omar Cetina Domínguez, and Betania Hernández Ocaña. Nuevas heurísticas inspiradas en la naturaleza para optimización numérica. 2010.

[17] Nareli Cruz Cortés. Sistema inmune articial para solucionar problemas de optimización. Octuber 2004.

[18] Jhon Holland. Adaptation in natural and artificial systems. *MIT Press*, 1992.

[19] Christorpher R. Houck, Jeffery A. Joines, and Michael G. Kay. A genetic algorithm for function optimization: A matlab implementation. 1995.

[20] Carlos A. Coello Coello. Introducción a la computación evolutiva. January 2004.

[21] Jhon R. Koza. Genetic programming: On the programming of computers by means of natural selection. *MIT Press*, 1992.

[22] Jorge A. Soria-Alcaraz, J. Martin Carpio-Valadez, and Hugo Terashima-Marin. Academic timetabling desing using hyper-heuristics. *Soft Computing for Intell. Control and Mob. Robot.*, 318:43–56, 2010.