In the 5th International Symposium on Ubiquitous Computing and Ambient Intelligence (UCAmI'11), Hotels Grand Palladium Resort & Spa, Riviera Maya, México, December 5-9, 2011

A comparison between PSO and MIMIC as strategies for minimizing cyclic instabilities in ambient intelligence

Leoncio Alberto Romero
Victor Zamudio
Rosario Baltazar
Marco Sotelo
Victor Callaghan

Abstract In this paper we present a comparison between two novel approaches to the fundamental problem of cyclic instability in ambient intelligence. These approaches are based on two optimization algorithms, Particle Swarm Optimization (PSO) and Mutual Information Maximization for Input Clustering (MIMIC). In order to be able to use these algorithms, we introduced the concept of *average accumulative oscillation*, which enabled us to measure the average oscillatory behaviour of the system. PSO and MIMIC have the advantage that they do not need to analyze the topological properties of the system, in particular the loops. In order to test these algorithms we used the well-known discrete system called the game of life for 9, 25, 49 and 289 agents. It was found that PSO performed better than MIMIC in terms of the number of agents blocked. These results were confirmed using the Wilcoxon test. This novel and successful approach is very promising, and can be used to remove instabilities in real scenarios with a large number of agents (including nomadic agents) and complex interactions and dependencies between them.

1 Introduction

Any computer system can have errors and ambient intelligence is not exempt from them. Cyclical instability is a fundamental problem characterized by the presence of unexpected oscillations caused by the interaction of the rules governing the agents involved [Zamudio 2008, Zamudio 2008b, Zamudio 2008c, Zamudio 2010].

The problem of cyclical instability in ambient intelligence is a problem that has received little attention by the designers of intelligent environments [Zamudio 2008b, Egerton 2009-1]. However in order to achieve the vision of AmI this problem must be solved.

The strategies reported in the literature (in particular the Instability Prevention System INPRES) are based on analyzing the topological properties of the Interaction Network (digraph associated to the system, capturing the dependencies of the rules between agents), finding the loops, and locking a subset of agents making up a loop, preventing them to change their state [Zamudio 2008, Zamudio 2008b, Zamudio 2008c, Zamudio 2010]. INPRES has been tested successfully in systems with low density of interconnections, and static rules (nomadic devices and time variant rules are not allowed). However when the number of agents involved in the system increases (with high dependencies between them) or when the agents are nomadic, INPRES is not practical.

In this paper we compare the results of Particle Swarm Optimization PSO and Mutual

Information Maximization for Input Clustering MIMIC when applied to the problem of cyclic instability. These algorithms find the best set of agents to be locked, in order to minimize the oscillatory behaviour of the system. In order to do so, the concept of average oscillatory behaviour was introduced. This approach has the advantage that there is not need to analyze the dependencies of the rules of the agents (as in the case of INPRES). We used the game of life (reference) to test this approach.

2 Particle Swarm Optimization

The Particle Swarm Optimization (PSO) algorithm was proposed by Kennedy and Everhart. It is based on choreography of a flock of birds. [Antony 2001, Coello 2002, Das 2005, Parposo 2005, Russel 2001, Tammer 2006].

The algorithm uses two equations. The first one is used to find the velocity, describes the size and direction of the step that will be taken by the particles and is based on the best particles found until that moment.

$$v_{id} = wv_{id} + c_1 r_1 \left(lBest_{id} - x_{id}\right) + c_2 r_2 \left(gBest_{id} - x_{id}\right) \quad (1)$$

where

d=1, 2, ..., D and D is the number of dimensions.
I=1, 2, …, N and N is the number of the population.
w, c_1, c_2, are parameters given by user.
r_1 and r_2 are random numbers in range [0, 1]
lBest is the best local particle
gBest is the best general particle

The second equation updates the current position of the particle to the new position using the result of the velocity equation.

$$x_{id} = x_{id} + v_{id}$$

The basic PSO algorithm is shown next [Eberhart 2001]:

1. Initialize a population (array) of particles with random positions and velocities on d dimensions in the problem space
2. For each particle, evaluate the desired optimization fitness function in d variables
3. Compare particle's fitness evaluation with particle's lBest. If current value is better than lBest, then set lBest value equal to the current value, ante the lBest location equal to the current location in d-dimensional space
4. Compare fitness evaluation with the population's gBest overall previous best. If current value is better than gBest, then rest gBest to the current particle's array index and value.
5. Change de velocity and position of the particle according to equation to update that

values respectively

6. Loop to step 2 until a criterion is met, usually a sufficiently good fitness or a maximum number of iterations (generations).

2.1 Binary PSO

The binary PSO was design to work in binary spaces. Binary PSO selects the lBest and gBest particles in the same way. 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 update velocity is based on probabilities but these probabilities must be in the range [0, 1]. For that a mapping is established for all real values of velocity to the range [0, 1].
The equation used for normalization is:

$$v_{ij}(t) = sig(v_{ij}(t)) = \frac{1}{1+e^{-vv_{ij}(t)}} \tag{2}$$

and the equation used to update the new particle position is

$$x_{ij}(t+1) = \begin{cases} 1\_if\_r_{ij} < sig(v_{ij}(t+1)) \\ 0\_in\_other\_case \end{cases} \tag{3}$$

Where r_{ij} is a random number in range [0, 1]

3 Mutual Information Maximization for Input Clustering MIMIC

The Mutual Information Maximization for Input Clustering (MIMIC) [Bonet 1997, Bosman 1999, Larrañaga 2003, Sotelo 2010] is part of the algorithms known These algorithms aim to get the probabilistic distribution of a population based on a set of samples, searching for the permutation associated to the lowest value of the Kullback-Leiberg divergence. This value is used to calculate the similarity between two different sets of samples:

$$H_l^{\pi} = h_l(x_{in}) + \sum_{j=1}^{n-1} h_l(x_{ij} \mid x_{ij+1}) \tag{4}$$

where:

$h(x) = -\sum_x p(X=x)\log p(X=x)$ is Shannon's entropy of X variable

$h(X \mid Y) = -\sum_x p(X \mid Y)\log p(Y=y)$, where
$h(X \mid Y=y) = -\sum_x p(X=x \mid Y=y)\log p(X=x \mid Y=y)$ is the X entropy given Y.

This algorithm suppose that the different variables have a bivariate dependency described by:

$$p_l^{\pi}(x) = p_l(x_{i1} \mid x_{i2}) \cdot p_l(x_{i2} \mid x_{i3}) \cdot \ldots \cdot p_l(x_{in-1} \mid x_{in}) \cdot p_l(x_{in}) \quad (5)$$

where $\pi = (i1, i2, \ldots in)$ is a index permutation

The steps of the algorithm are as follows:

1. Initialize a population (array) of individuals with random values on d dimensions in the problem space
2. Select a subpopulation throw a selection method
3. Calculated Shannon's entropy for each variable.
4. Generate a permutation $p_l^{\pi}(x)$

   Chose variable with lowest entropy.
   For the next variables chose $i_k = \arg\min_j h_l(X_j \mid X_{lk+1})$
   Where $j \neq i_{k+1}, \ldots, i_n$.

5. Sample the new population using the generated permutation $p_l^{\pi}(x)$
6. Loop until a criterion is met.

4 Using PSO and MIMIC to solve the problem of cyclic instability

In order to solve the problem of cyclic instability using PSO and MIMIC we need to minimize the amplitude of the oscillations. In the ideal case this would result in a stable system. Additionally we are interested in affecting the fewest number of agents (agents locked).

In order to test these approaches we used the game of life with open boundary conditions. The open boundary condition in our case is considered cold (in terms of heat entropy) and all cells outside the grid are considered dead. We enriched the game of life with additional conditions: a list of agents that are allowed to be locked. PSO and MIMIC can lock them according to their results. This is because priority agents (such as alarms, security systems) shouldn't be disabled.

Each solution vector represents the list of blocked agents where the aim is to minimize the average oscillation of the system in a given period of time. The average oscillation is calculated using the following equation

$$o = \frac{\sum_{i=1}^{n-1} |S_i - S_{i+1}|}{n-1} \quad (6)$$

Where:

o: average oscillation

n: game of life generations

S_i state of the system at the time i

S_{i+1} state of the system at the time i+1

The best solution should not only minimize the amplitude of oscillation but also the number of agents locked. In these experiments the percentage of agents that can be locked is included as a parameter. This is important because, as this percentage grows, the systems becomes more disabled.

In these experiments we consider systems whose adjacency matrix are of 3x3, 5x5, 7x7, and 17x17. In all the cases the percentage of maximum locked agents was of 20%.

If PSO or MIMIC can not find a better solution in terms of the amplitude of the oscillations no agents will be locked.

In our experiments we set a parameter, 3000 functions calls, as the 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 is deemed to have failed.

5 Experimental Results

The first test was performed using the oscillator known as blinker inside in a grid of 3x3. Parameters and results for this test are showed in table 1. In this case PSO locked 1 agent, and MIMIC locked 2 agents. PSO was able to minimize the oscillations, with an average accumulative oscillation of 0. 0349637605, while MIMIC's value was 100.0349637605.

Parameter	***PSO***	***MIMIC***
Matrix	3x3	3x3
Agents	9	9
Individuals	20	20
Function Calls	3000	3000
Maximum % Locked Agents	0.2	0.2
Number of agents locked	1	2
Game of life Generations	50	50
Average accumulative oscillation without agents locked	0.4161648288	0.4161648288
Average accumulative oscillation with agents locked	0.0349637605	100.0349637605

Table 1. Summary of the results comparing PSO and MIMIC for the case of 3x3

In the second test we used a configuration without oscillation, and both algorithms found a configuration to decrease the average accumulative oscillation. Parameters and results for this test are showed in table 2. In this case a system of 7x7 was used (49 agents) and PSO was able to find a better solution locking only 7 agents.

Parameter	***PSO***	***MIMIC***
Matrix	7x7	7x7
Agents	49	49
Individuals	30	30
Function Calls	3000	3000
Maximum % Locked Agents	0.2	0.2
Number of agents locked	7	21
Game of life Generations	50	50
Average oscillation without agents locked	0.2802692471	0.2802692471
Average oscillation with agents locked	0.0062323279	100.0000595946

Table 2. Summary of the results comparing PSO and MIMIC for the case of 7x7.

In third test we used the oscillator called blinker with a 5x5 grid size. Parameters and results for this test are showed in table 3. In this case PSO locked only 1 agent, minimizing the average oscillations from 0.9748569646 to 0.0725226996.

Parameter	***PSO***	***MIMIC***
Matrix	5x5	5x5
Agents	25	25
Individuals	20	20
Function Calls	3000	3000
Maximum % Locked Agents	0.2	0.2
Number of agents locked	1	6
Game of life Generations	50	50
Average oscillation without agents locked	0.9748569646	0.9748569646
Average oscillation with agents locked	0.0725226996	100.0725226996

Table 3. Summary of the results comparing PSO and MIMIC for the case of 5x5

The fourth experiment considered the oscillator known as pulsar. Pulsar has 3 different modes of oscillation. For this test we used a 17x17 grid size. Additional parameters are shown in table 4.

Parameter	***PSO***	***MIMIC***
Matrix	17x17	17x17
Agents	289	289
Individuals	100	100
Function Calls	3000	3000
Maximum % Locked Agents	0.2	0.2
Number of agents locked	20	131
Game of life Generations	50	50
Average oscillation without agents locked	2.2132849513	2.2132849513
Average oscillation with agents locked	0.0120411998	100.0095424251

Table 4. Summary of the results comparing PSO and MIMIC for the case of 17x17, using the oscillator known as pulsar.

In the last test we used a 10 cell row as initial state inside a 17x17 grid size. From that state the system oscillates, showing 14 different patterns of oscillation. The parameters and results for this test are shown in table 5. In this experiment involving 289 agents, PSO locked 17 agents with an average oscillation of 0.015771896 and MIMIC locked 145, with an average oscillation of 100.1666385973.

Parameter	PSO	MIMIC
Matrix	17x17	17x17
Agents	289	289
Individuals	100	100
Function Calls	3000	3000
Maximum % Locked Agents	0.2	0.2
Number of agents locked	17	145
Game of life Generations	50	50
Average oscillation without agents locked	0.135629376	0.135629376
Average oscillation with agents locked	0.015771896	100.1666385973

Table 5. Summary of the results comparing PSO and MIMIC for the case of 17x17

In figure 1 and 2 we show the behaviour of the system for experiment 4 (pulsar), with and

without locking respectively.

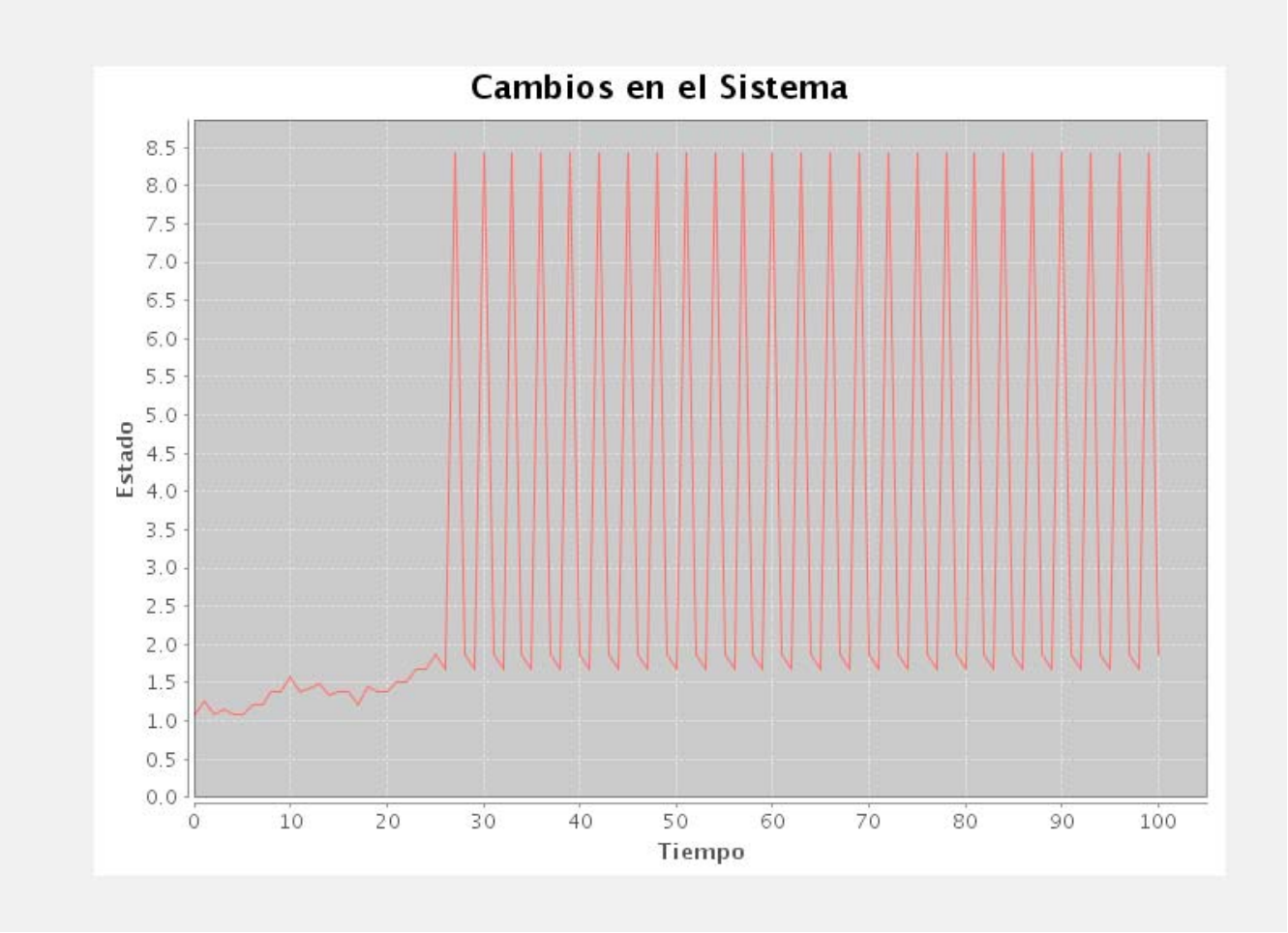

Fig 1. Oscillatory behaviour of the system for the case of 17x17 (pulsar)

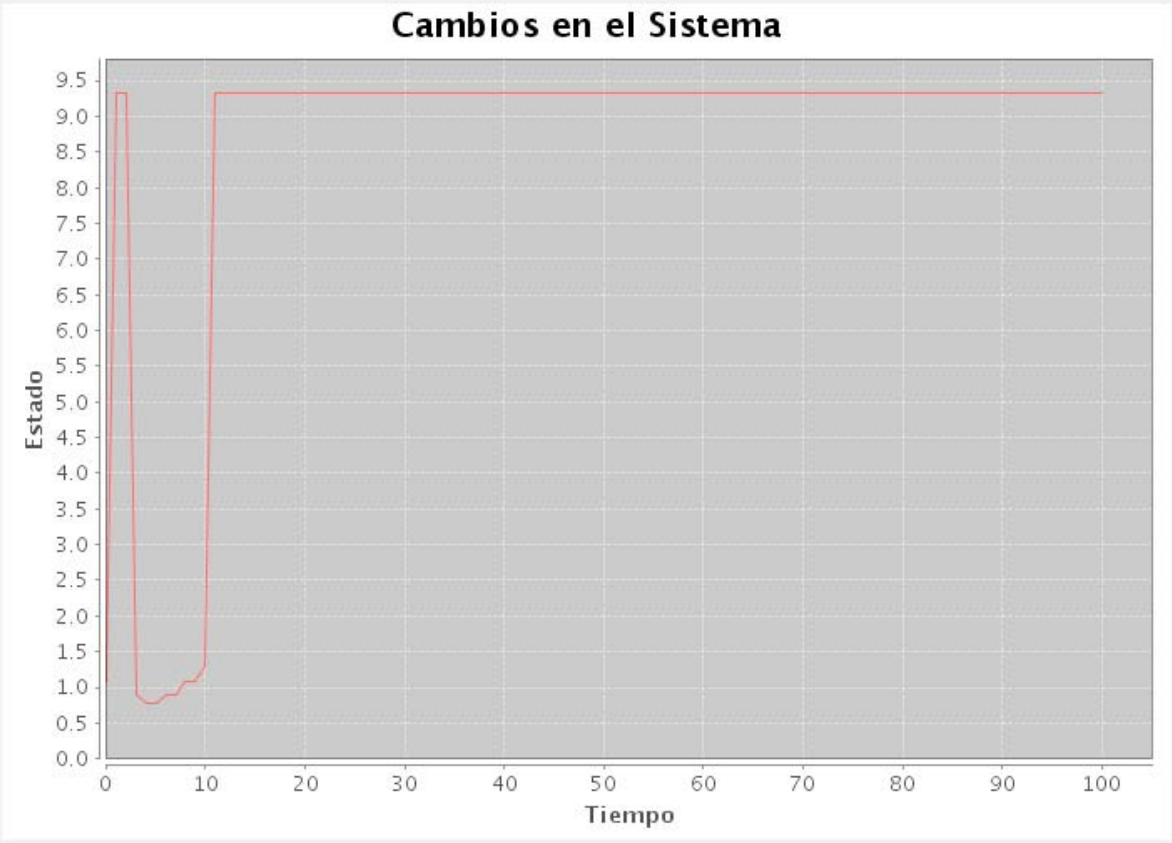

Fig 2. Instabilities are successfully removed to the 17x17 (pulsar) system using PSO.

6 Conclusions and Future Work

From our experiments we found that PSO was able to find a vector of locked agents that prevent the system from oscillating. Additionally Wilcoxon test showed that PSO has better results in terms of time and number of agents locked. MIMIC consistently violated the restriction of maximum percentage of agents locked permitted. MIMIC is based on estimating the distribution of data and for that reason needs a larger amount of data (in our case the amount of list of agents locked), and that is the main reason the time spent for the algorithm to find a solution increases significantly.

The parameters used internally by PSO were found experimentally. Once these parameters where set conveniently, PSO perform better than MIMIC, which is consistent with Wilcoxon test.

This new approach open the possibility for other algorithms to be applied for the problem of cyclic instability, such as genetic algorithms, simulated annealing, mimetic Algorithms, Bio-inspired algorithms or those based on a social metaphor and, in general, algorithms for discrete optimization. In particular we are interested in testing this approach with the case of nomadic and weighted agents using different percentages of locked agents. Additionally it is possible to improve the estimation of the average oscillation in order to be able of discriminate

between stable systems with abrupt changes and systems with small oscillations. We hope to report this results in future conferences.

Acknowledgements

The authors want to thank Jorge Soria for their comments and suggestions to his work. Leoncio Romero acknowledge the support of the National Council for Science and Technology CONACyT.

References

A. Egerton, V. Zamudio, V. Callaghan and G. Clarke, "Instability and Irrationality: Destructive and Constructive Services within Intelligent Environments". Essex University, 2009.

M. Weiser, "The Computer for the 21st Century". Scientific American, pp. 94-10, September 1991

M. Satyanarayanan, "Pervasive Computing: Vision and Challenges". IEEE Personal Communicatios, Vol:8 Issue: 4, August 2001, Pages, 10-17

Victor Manuel Zamudio Rodriguez, "Understanding and Preventing Periodic Behavior in Ambient Intelligence". University of Essex, October 2008

V. Zamudio and V. Callaghan, "Facilitating the Ambient Intelligent Vision: A Theorem, Representation and Solution for Instability in Rule-Based Multi-Agent Systems". University of Essex, 2008.

Victor Zamudio and Vic Callaghan, "Understanding and Avoiding Interaction Based Instability in Pervasive Computing Environments". University of Essex, Octuber 2008

Referencias V. Zamudio, R. Baltazar, M. Casillas, "c-INPRES: Coupling Analysis Towards Locking Optimization in Ambient Intelligence". Instito Tecnológico de Leon, 2010.

Jeremy S. De Bonet, Charles L. Isbell y Jr., Paul Viola, "MIMIC: Finding Optima by Estimating Probability Densities". Advances in Neural Proessing Systems MIT Press, Cambridge, MA. 1997.

Pedro Larrañaga, Jose A. Lozano y Heinz Mü hlenbein, "Algoritmos de Estimación de Distribuciones en o Problemas de Optimización Combinatoria". Inteligencia Artificial, Revista Iberoamericana de Inteligencia Artificial. No. 19, 2003.

Peter A. N. Bosman y Dirk Thierens, "Linkage Information Processing In Distribution Estimation Algorithms". Departament of Computer Science, Utrecht University. 1999.

Russel C. Eberhart e Yuhui Shi, "Particle Swarm Optimization: Developments, Applications and Resources". IEEE, 2001. Paginas 82-86.

Carlos A. Coello y Maximo Salazar, "MOPSO: A proposal for multiple Objetive Particle Swarm Optimization". Evolutionary Computation, IEEE, 2002. Paginas 1051-1056

Swagatam Das, Amit Konar y Uday K. Chakraborty, "Improving Particle Swarm Optimization with Differentially Perturbed Velocity". GECCO, 2005. Paginas 177-184

Anthony Carlise y Gerry Dozier, "An Off-The-Shelf PSO". Purde Scholl of Engieneering and Technology, Indianapolis, IN, 2001

K.E. Parsopoulos y M. N. Vrahatis, "Initializing The Particle Swarm Optimizer Using The Nonlinear Simplex Method". GECCO, 2005.

Tamer M. Khali, Hosam K. M. Youssef y 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, Junio 2006

Marco A. Sotelo, "Aplicación de Metaheurísticas en el Knapsack Problem". Instituto Tecnológico de León, León, Gto. 2010.

Hans Kellerer, Ulrich Pferschy y David Pisinger, "Knapsack Problems". Springer's internal project number, Octuber, 2003.

Egon Balas y Eitan Zemel, "An Algorithm for Large Zero-One Knapsack Problems". Operations Research Society of America, Diciembre, 1997.

Sami Khuri, Tomas Bäck y Jörg Heitkötter, "The Zero/One MUltiple Knapsack Problem and Genetic Algotihms". ACM Symposium of Applied Computation, 1993.

Juan E. Nápoles Valdes, "El juego de la Vida: Geometría Dinámica". Universidad de la Cuenca del Plata, Argentina.

