Implementing a Model of Japanese Tree Frogs' Calling Behavior in Sensor Networks: A Study of Possible Improvements

Hugo Hernández
ALBCOM Research Group
Universitat Politècnica de Catalunya
Barcelona, Spain
hhernandez@lsi.upc.edu

Christian Blum*
ALBCOM Research Group
Universitat Politècnica de Catalunya
Barcelona, Spain
cblum@lsi.upc.edu

ABSTRACT

Male Japanese tree frogs exhibit a self-organized behavior for the desynchronization of their calls. This property has evolved because female frogs are not able to correctly localize the male frogs when their calls are too close in time. A model for this behavior has been proposed in the literature. However, its use in technical applications is, so far, quite limited.

In this paper we implement the originally proposed model in sensor networks, with the aim of desynchronizing neighboring nodes as much as possible. Moreover, we propose extensions of the original model. Experimental results show that the proposed extensions improve the desynchronization capabilities of the original model.

Categories and Subject Descriptors

I.2.11 [**Computing Methodologies**]: Artificial Intelligence—*Distributed Artificial Intelligence*

General Terms

Algorithms, performance

Keywords

Sensor networks, Japanese tree frogs, calling behavior

1. INTRODUCTION

Different studies (see, for example, [7]) have shown that male Japanese tree frogs use their calling to attract females. Apparently, females of this family of frogs can recognize the source of the calling in order to determine the current location of the corresponding male. A problem arises when two of these males are too close in space and communicate at the same time. In this case females are not able to properly recognize both calls independently and are, therefore, unable to detect where the calls came from. For this reason, males have evolved to desynchronize their sounds in time. They achieve to uniformly distribute the distance between each pair of calls, which allows the females to locate the males they can hear, and to choose one. In fact, this behavior is a prime example for self-organization in nature.

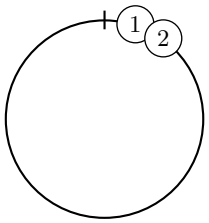
More recently, Aihara et al. [1] introduced a formal model based on a set of coupled oscillators each one simulating the phase change in the calling period of a single frog. As oscillators are associated to frogs, we will use both terms in the following with the same meaning. The basic way of working of this model is graphically illustrated in Figure 1. The circle represents—in all three graphics—the time frame between two calls of the same frog (2π), the calling period. The nodes marked by integer numbers 1 and 2 indicate the phase of the corresponding frogs, that is, the moment of time in which they call. Note that the oscillators are not able to reach perfect anti-phase in a single step. In general, an indefinite number of steps is needed before reaching the stable situation corresponding to perfect anti-phase. Moreover, the difficulty of reaching the optimal configuration tends to increase with an increasing number of frogs and also with an increasing degree of interaction between them (note that two frogs that can not hear each other do not influence each other).

Technically, the system introduced by Aihara et al. [1] works as follows. Each oscillator i has a phase $\theta_i \in [0, 2\pi]$ that changes over time with frequency ω_i (where 2π is the time interval between two calls of the same frog, the calling period). When the phase reaches 2π, the oscillator fires and returns to the baseline. In addition, oscillators may be coupled with other oscillators. In case an oscillator j is coupled to an oscillator i, when oscillator i fires, oscillator j receives a boost and changes the frequency of firing in the next round depending on the gap $\Delta_{ji} \in [0, 2\pi]$ (see below) between both oscillators. These changes do not happen instantly upon receiving the stimulus. The corresponding oscillator rather waits until it fires. The model can be summarized in the following equations. First, the behavior of an isolated oscillator i is modelled as follows:

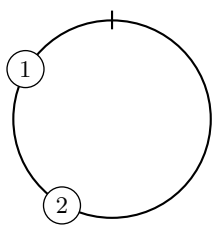
$$\frac{d\theta_i}{dt} = \omega_i \tag{1}$$

*Corresponding author.

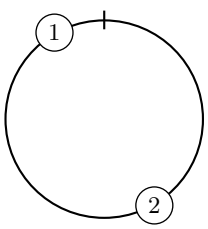
GECCO'11, July 12–16, 2011, Dublin, Ireland.



(a) Fictitious initial situation with two frogs calling close in time.



(b) The system after some iterations. The system has managed to increase the distance between the calls of the two frogs.



(c) Final situation. The two frogs call in perfect anti-phase.

Figure 1: Graphical illustration of the working of a system of two coupled oscillators. The circle in all three graphics represents the time frame between two calls of the same frog (2π), the calling period. The nodes marked by integer numbers 1 and 2 indicate the phase of the corresponding frogs, that is, the moment of time in which they call. (a) shows a fictitious initial situation. (b) shows the situation after some iterations. Clearly the system tries to put some distance between the calling of frogs 1 and 2. (c) shows an optimal final situation in which the frogs (or oscillators) are in perfect anti-phase, that is, their respective calls have the reached the maximum distance in time (half a circle).

Assuming that oscillators j and i are coupled, the gap between their (current) phases is defined as:

$$\Delta_{ji} = \theta_j - \theta_i \tag{2}$$

Now, the change in the behavior of oscillator j as influenced by oscillator i can be described as follows:

$$\frac{d\theta_j}{dt} = \omega_j + g(\Delta_{ji}) \quad , \tag{3}$$

where $g(\cdot)$ is the phase shift function which is responsible for changing the phase of the frogs that are influenced by other frogs. In [1], the authors suggest the use of the following phase shift function:

$$g(x) = -\alpha \sin(x) \tag{4}$$

We say that this system of oscillators is in a stable situation and in anti-phase when the following two conditions are satisfied:

$$\Delta_{ij} = \Delta_{ji} \quad , \tag{5}$$

$$g(\Delta_{ij}) = 0 \quad , \tag{6}$$

for all $i \neq j$. The system presented in [1] is able to successfully locate two coupled oscillators in perfect anti-phase, independent of the initial settings of θ_1 and θ_2. Unfortunately, several problems arise when the number of oscillators grows. Figure 2 shows two examples for such problems. Given an undirected graph $G = (V, E)$, henceforth we will assign one oscillator to each node in the graph. Therefore, in the following the terms *node* and *oscillator* will refer to the same. We consider that two oscillators are coupled if and only if their corresponding nodes are connected by an edge. Depending on the initial phases of the oscillators, for both topologies shown in Figures 2(a) and 2(d) it is possible to reach suboptimal desynchronizations (as shown in Figures 2(b) and 2(e)). The corresponding optimal desynchronizations are shown in Figures 2(c) and 2(f). In [1] the authors provide analytical results for using three oscillators and show that there is a high system sensitivity with respect to the initial phases, which means that only a small subset of the possible initial settings leads to an optimal distribution of the θ-values.

The initial model by Aihara et al. [1] was later extended by Mutazono et al. [6]. They used their extended model for anti-phase synchronization for the purpose of collision-free transmission scheduling in sensor networks. In order to make the system applicable to larger topologies (sensor networks may consists of hundreds of nodes), they introduced weights in order to regulate the coupling between each pair of oscillators. The resulting phase shift function as introduced in [6] can be described as follows:

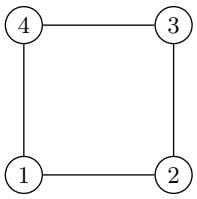
$$\delta(x) = \min\{x, 2\pi - x\} \quad , \tag{7}$$

$$g(x) = \alpha \sin(x) \cdot e^{-\delta(x)} \tag{8}$$

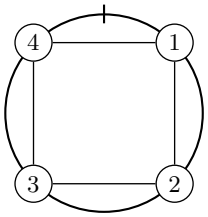
Thanks to these weights, the system reaches stable situations more easily, especially when rather small values of α are used. The authors experimented with topologies of up to 20 nodes and although the system still showed certain difficulties to reach stable solutions, the sensitivity to initial conditions decreased significantly.

Mutazono et al. [6] compared the results of their system to another mechanism for coupled oscillator desynchronization proposed in [3]. Note that the mechanism from [3] is not based on the calling behavior of Japanese tree frogs. The main difference to frog-inspired systems is the fact that the phase change of a node is made on the basis of only two other nodes. The phase values allow to order all the nodes sequentially from small to large phase values. The nodes whose phase values are used to change the phase value of a node are determined as the predecessor and the successor in this (cyclic) sequence. As shown in [6], both systems achieve similar results although no extensive experimentation is made on a broad-enough set of network topologies: mostly random geometric graphs and hand-made instances with at most eight nodes were used.

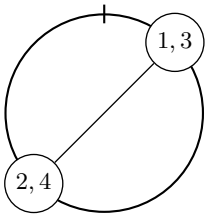
Another extension of the system by Aihara et al. [1] was introduced in [5]. The changes concern the use of different weights for the phase shift function and the introduction



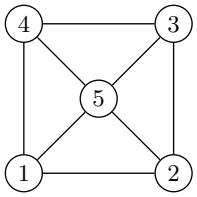
(a) Topology 1



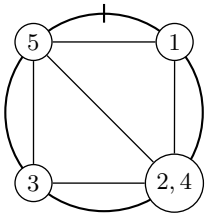
(b) Suboptimal desynchronization of topology 1 (with 4 different phases)



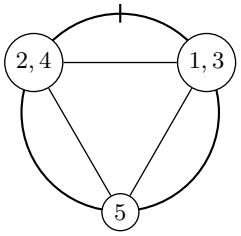
(c) Optimal desynchronization of topology 1 (with 2 different phases)



(d) Topology 2



(e) Suboptimal desynchronization of topology 2 (with 4 different phases)



(f) Optimal desynchronization of topology 2 (with three different phases)

Figure 2: Two examples for graph topologies (graphics (a) and (d)) that may cause problems for the desynchronization as performed by the model proposed in [1]. Graphics (b) and (e) show suboptimal desynchronizations (corresponding to stable attractors of the system) for both topologies. In contrast, graphics (c) and (f) show optimal desynchronizations.

of a so-called frustration parameter which reduces the coupling between each pair of nodes. The authors show that their system is able to obtain better solutions than the original model for many different topologies as, for example, k-partite graphs, grids or platonic solids. Moreover, the authors make some interesting observations: (1) the number of oscillators is not the key factor for achieving desynchronization. It is rather the topology which most determines the problem complexity. (2) the time distance between phases is not uniformly distributed around the whole period. The number of nodes firing at each phase strongly affects the amount of time between the phases.

The rest of the paper is organized as follows. In Section 2 we introduce our implementation of the original model by Aihara et al. in sensor networks. Moreover, we propose our extensions of this model. Next, in Section 3 we provide experimental results concerning the comparison of the original model with the extended model. Finally, in Section 4 we present conclusions and an outlook to future work.

2. IMPLEMENTATION OF THE ORIGINAL AND THE EXTENDED MODEL IN SENSOR NETWORKS

First we provide the implementation of the original model by Aihara et al. in sensor networks. In this implementation, communication rounds are the basic units of time. A communication round corresponds to the calling period (2π) as known from the models presented in the previous section. The only difference is that the length of a communication round is considered to be one time unit. Therefore, the numerical length of a communication round is denoted by 1, instead of 2π. Each sensor node executes exactly one sensor event in each communication round. The moment in time when a sensor node $i \in V$ executes its sensor event is denoted by $\theta_i \in [0, 1)$. Note that θ_i corresponds to the phase of an oscillator from the model(s) presented in the previous section. Moreover, a sensor event includes the sending of exactly one message. Therefore, each sensor node i maintains a message queue M_i for sensor event messages received from other sensor nodes since the last execution of its own sensor event. The pseudo-code of a sensor event is shown in Algorithm 1. In the following we give a rough description of the algorithm. A detailed technical explanation of the functions of Algorithm 1 will be provided later on.

Algorithm 1 Sensor event of node i

```
1: θ_i :=recalculateTheta()
2: sendMessage()
3: clearMessageQueue()
```

The working of the algorithm requires an a priori organization of the sensor network in form of a rooted tree. The root node of this tree (henceforth also called *master node*) will have some additional functionalities in comparison to the rest of the nodes. Once the tree is available, the master node runs a protocol to calculate the height of the tree, that is, the distance in hops (communication rounds) from the master node to the farthest node in the network. In order to produce a tree with a low height, the distributed method for generating spanning trees with minimum diameter as presented in [2] may be used. Next, the master node triggers the start of the main algorithm by means of a broadcast

message. In this message, the height of the tree is communicated to the rest of the nodes as well. Later on it will be described how this overlay tree structure is used to calculate the state of convergence which will be used to stop the algorithm.

As mentioned above, in each communication round, a node i executes its sensor event at time θ_i. First, node i will examine its message queue M_i. If M_i contains more than one message from the same sender node, all these messages apart from the last one are deleted. In general, a sensor event message $m \in M_i$ contains only one real number:

$$m =< \text{theta}_m > \quad , \tag{9}$$

where $\text{theta}_m \in [0.1)$ contains the θ-value of the emitter. Based on the messages in M_i, function recalculateTheta() recalculates a new value for θ_i:

$$\theta_i := \theta_i - \alpha_i \sum_{m \in M_i} \frac{\mathbf{sin}(2\pi \cdot (\theta_m - \theta_i))}{2\pi} \quad , \tag{10}$$

where $\alpha_i \in [0, 1]$ is a parameter used to control the convergence of the system. In general, the lower the value of α_i the smaller the change applied to θ_i. Note that the multiplication of $(\theta_m - \theta_i)$ with 2π and the division of the result of the sinus function by 2π is necessary for the transformation of the length of a calling period from $[0, 2\pi]$ to $[0, 1]$. Finally, node i sends the following message m (see function sendMessage()):

$$m =< \text{theta}_m := \theta_i > \tag{11}$$

To conclude a sensor event, node i deletes all messages from its queue M_i (see function clearMessageQueue()), that is, $M_i = \emptyset$.

It remains to provide a description of the algorithms' stopping mechanism. In this context recall the rooted tree that is generated before the execution of the main algorithm. The key characteristic of this tree is its height, henceforth referred to by h. It determines the maximum number of communication rounds necessary for the master node to broadcast a message to all other nodes. In turn, it also determines the maximum number of communication rounds necessary to pass information from all the nodes to the master node. The main goal of the tree is to efficiently alert the nodes about when and how to stop executing the sensor events for desynchronization. In the following we assume that the master node knows the size of the network. At each communication round, each node i must communicate the following information to its parent node in the tree: (1) a real number corresponding to the sum of the distances between the old theta values and the new ones concerning all nodes included in the subtree rooted at node i, (2) an integer number that indicates the corresponding communication round. In fact, these values can easily be added to the body of the sensor event messages used by our algorithm. In other words, no additional messages are required.

Note that, for example, in the first communication round only the leaves of the tree will report the differences between their old and new theta values to their parents. This is because the leaves are the only nodes without children. In the second communication round, the parents of the leaves will be able to add these values to their own distances between the old and new theta values from the first communication round and report the aggregated data to their respective parents. Given the height h of the tree, it takes h communication rounds until all the information regarding a specific communication round has reached the master node. This means that the sensor nodes must store the differences between their old and new theta values during h communication rounds. Once all the necessary information reaches the master node, it divides the value by the size of the network and obtains in this way the average change of the theta values in the corresponding communication round. In case this average change is below a certain threshold value (in our case we always used 0.001), the master node broadcasts a stopping message to all nodes, which terminates the algorithm.

2.1 Model Extensions

As in the original model, in each communication round a node i executes its sensor event at time θ_i. However, a message $m \in M_i$ has now the following format:

$$m =< \text{theta}_m, \text{relevance}_m > \quad , \tag{12}$$

where, as before, $\text{theta}_m \in [0.1)$ contains the θ-value of the emitter and relevance_m is a parameter that depends on the number of messages received by the emitter during the last communication round. This parameter controls the weight that is given by node i to the corresponding message m. In particular, less weight is given to messages that were emitted by nodes that are influenced by many other nodes. The intuition for this definition of the weights is that the θ-values of nodes that are little influenced by other nodes should converge first. This may facilitate the convergence of the θ-values of highly-influenced nodes, which in turn may facilitate that the system reaches a stable situation, a term which refers to a situation in which the θ-values do not change anymore.

Based on the messages in M_i, function recalculateTheta() recalculates a new value for θ_i:

$$\theta_i := \theta_i + \alpha_i \sum_{m \in M_i} \text{relevance}_m * inc[\theta_m - \theta_i] \quad , \tag{13}$$

where $\alpha_i \in [0, 1]$ is, again, a parameter used to control the convergence of the system. Moreover, inc$[\cdot]$ is a new function that replaces the phase shift function of Equation 4. This new function is defined as follows:

$$inc[x] = \begin{cases} x - 0.5 & \text{if } x \geq 0 \\ x + 0.5 & \text{if } x < 0 \end{cases} \tag{14}$$

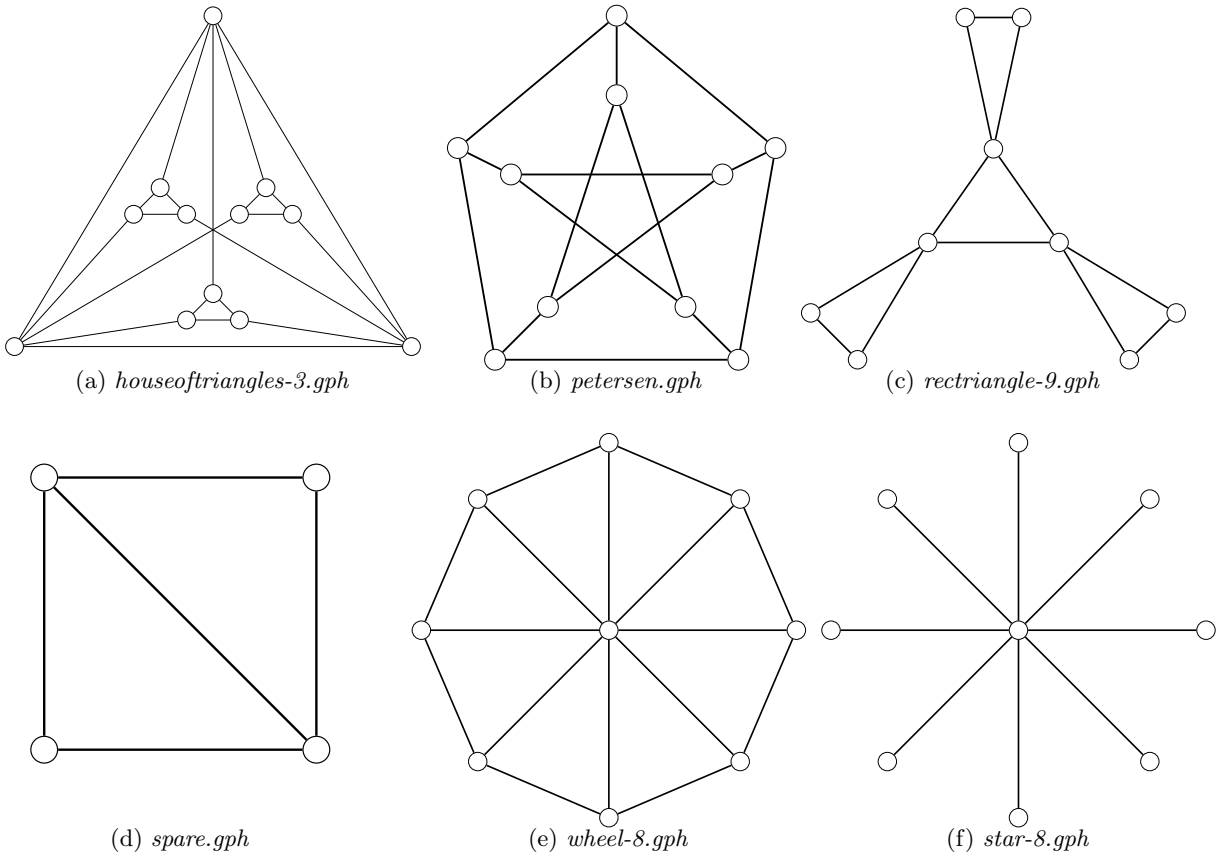
The hope is to achieve a similar, or even better, convergence behavior with this much simpler function. Finally, node i sends the following message m (see function sendMessage()):

$$m =< \text{theta}_m := \theta_i, \text{relevance}_m := \frac{1}{|M_i|} > \tag{15}$$

Note that when $M_i = \emptyset$, then relevance_m is, of course, set to 1. As described before, to conclude a sensor event, node i deletes all messages from its queue M_i (see function clearMessageQueue()), that is, $M_i = \emptyset$.

3. EXPERIMENTS

We applied four versions of the presented algorithm to 12 small sensor network topologies. The four algorithm versions are defined as follows: (1) The original model, (2) the original model with the relevance term, (3) the new model



(a) *houseoftriangles-3.gph* (b) *petersen.gph* (c) *rectriangle-9.gph*

(d) *spare.gph* (e) *wheel-8.gph* (f) *star-8.gph*

Figure 3: Sensor network topologies used in this work.

(that is, using the simplified phase shift function) without the relevance term, and (4) the new model with the relevance term. Each of these four algorithm versions was applied 100 times to each of the following 12 sensor network topologies:

- *line-2.gph*: two connected nodes
- *line-3.gph*: three connected nodes in a line
- *line-10.gph*: 10 connected nodes in a line
- *cycle-3.gph*: three fully connected nodes
- *cycle-4.gph*: four cyclicly connected nodes
- *cycle-10.gph*: 10 cyclicly connected nodes
- *houseoftriangles-3.gph*: see Figure 3(a)
- *petersen.gph*: see Figure 3(b)
- *rectriangle-9.gph*: see Figure 3(c)
- *spare.gph*: see Figure 3(d)
- *wheel-8.gph*: see Figure 3(e)
- *star-8.gph*: see Figure 3(f)

The last six of these topologies are shown in Figure 3. Note that all these topologies must be interpreted as follows. For each two nodes that are connected by a link we assume that the corresponding nodes receive the transmissions of each other. The optimal distribution of the theta values can be determined easily in each of the above described topologies. In this context, the theta values are considered to be optimally distributed if they are maximally separated from each other. For example, in the case of *line-2.gph* this value is 0.5. Another example concerns *spare.gph* where this value is $0.\overline{3}$.

Results are shown in Tables 1 and 2. For each of the four algorithm versions we indicate for each of the 12 sensor network topologies the number of times (out of 100 applications) in which an optimal distribution of the theta values was reached. Hereby, Table 1 provides these numbers concerning an error margin of 5%, whereas the numbers given in Table 2 are calculated on the basis of an error margin of 15%. For each algorithm version there are four columns of numbers. Each of these columns corresponds to a different value of α.

The results can be interpreted as follows. First, the relevance term seems, unfortunately, rather not useful. Both the behavior of the original model and the one of the model with the alternative phase shift function deteriorate when using the relevance term. Second, the results of the model with the new phase shift function are—apart from only few exceptions—better than the results obtained with the original model. This applies in particular to more difficult cases such as *spare.gph*, *wheel-8.gph*, and *houseoftriangles-3.gph*. Moreover, the results indicate that a rather high setting of

α, that is $\alpha \in \{0.75, 1.0\}$ seems to be best for both model versions. The results of Table 2, however, indicate that consistent results are rather achieved with smaller values for α, that is, $\alpha \in \{0.5, 0.75\}$. Finally, for the sake of completeness, Table 3 shows the average distances from the optimal distribution of the theta values.

Finally, it is also interesting to study the average number of communication rounds needed by the different algorithm versions before reaching the stopping condition. This information is given in Table 4. Note that the model using the new phase shift function needs significantly less communication rounds than the system using the original phase shift function. The faster convergence caused by the new phase shift function can also be observed graphically in Figure 4. More specifically, Figures 4(a) and 4(b) show the evolution of the average distance between the theta values of nodes connected by a link for graph *spare.gph*. Hereby, subfigure (a) presents the behavior of the original model, whereas subfigure (b) concerns the behavior of the model using the new phase shift function. Finally, Figures 4(c) and 4(d) show the evolution of the four theta values for a representative run. While subfigure (c) presents this evolution for the original model, sugfigure (d) concerns the evolution of the theta values as obtained from the model using the new phase shift function. Observe that the separation between the theta values is clearly better in subfigure (d).

4. CONCLUSIONS

In this paper we have implemented a model for the desynchronization of the calling of male Japanese tree frogs in sensor networks. Moreover, we have proposed possible extensions of this model concerning the weight that is given to different sensor nodes (relevance) and also concerning the phase shift function that is responsible for the desynchronization of the sensor nodes over time. The presented results have shown that especially the new phase shift function helps in improving the desynchronization capabilities of the system, both in quality and speed.

The proposed model may be used in various application scenarios appearing in sensor networks. First, the model may be used, for example, for distributed graph coloring. In fact, in [4] we have made an attempt in this direction. Another example concerns TDMA slot assignment for collision-free transmission.

Acknowledgements

This work was supported by grant TIN2007-66523 (FORMALISM) of the Spanish government, and by the EU project FRONTS (FP7-ICT-2007-1). In addition, Christian Blum acknowledges support from the *Ramón y Cajal* program of the Spanish Government, and Hugo Hernández acknowledges support from the *Comissionat per a Universitats i Recerca del Departament d'Innovació, Universitats i Empresa de la Generalitat de Catalunya* and from the *European Social Fund*.

5. REFERENCES

[1] I. Aihara, H. Kitahata, K. Yoshikawa, and K. Aihara. Mathematical modeling of frogs' calling behavior and its possible application to artificial life and robotics. *Artificial Life and Robotics*, 12(1):29–32, 2008.

[2] M. Bui, F. Butelle, and C. Lavault. A distributed algorithm for constructing a minimum diameter spanning tree. *Journal of Parallel and Distributed Computing*, 64(5):571–577, 2004.

[3] J. Degesys and R. Nagpal. Towards desynchronization of multi-hop topologies. In S. Brueckner, P. Robertson, and U. Bellur, editors, *Proceedings of the 2nd IEEE International Conference Self-Adaptive and Self-Organizing Systems*, pages 129–138. IEEE Press, 2008.

[4] H. Hernández and C. Blum. Distributed graph coloring: An approach based on the calling behavior of Japanese tree frogs. Available as ArXiv file `http://arxiv.org/abs/1011.5349`, 2010.

[5] S. Lee and R. Lister. Experiments in the dynamics of phase coupled oscillators when applied to graph coloring. In *Proceedings of ACSC 2008 – Proceedings of the thirty-first Australasian conference on Computer science*, pages 83–89. Australian Computer Society, Inc., 2008.

[6] A. Mutazono, M. Sugano, and M. Murata. Frog Call-Inspired Self-Organizing Anti-Phase Synchronization for Wireless Sensor Networks. In *INDS 09 – Proceedings of the* 2^{nd} *International Workshop on Nonlinear Dynamics and Synchronization*, pages 81 – 88. IEEE Press, 2009.

[7] K. Wells. The social behaviour of anuran amphibians. *Animal Behaviour*, 25:666–693, 1977.

Table 1: Success rates (in terms of the number of successful applications out of 100) calculated on the basis of an error margin of 5%.

Instance	Model using the new phase shift function								Original model by Aihara et al.							
	No relevance				Relevance				No relevance				Relevance			
	0.25	0.5	0.75	1.0	0.25	0.5	0.75	1.0	0.25	0.5	0.75	1.0	0.25	0.5	0.75	1.0
cycle-10.gph	7	65	70	66	0	8	54	68	4	79	72	77	0	3	70	78
cycle-3.gph	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100
cycle-4.gph	89	94	98	100	46	81	90	96	100	100	100	100	48	99	98	100
houseoftriangles-3.gph	18	13	9	10	22	30	15	16	0	0	0	0	27	8	1	0
line-10.gph	8	52	100	100	0	6	27	41	14	45	100	100	0	4	25	29
line-2.gph	100	100	100	100	100	100	100	100	100	100	100	100	99	100	100	100
line-3.gph	99	100	100	100	62	100	100	98	100	99	100	98	41	99	100	97
petersen.gph	8	2	1	0	1	1	5	6	0	0	0	0	2	0	0	0
rectriangle-9.gph	100	100	100	100	76	100	100	100	100	100	100	100	69	95	100	99
spare.gph	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
star-8.gph	99	100	99	100	0	0	0	93	98	97	99	100	0	0	0	84
wheel-8.gph	60	70	66	67	32	23	15	26	16	12	11	7	29	12	10	10
averages	57.333	66.333	**70.250**	**70.250**	36.583	45.750	50.500	62.000	52.667	61.000	**65.167**	**65.167**	34.583	43.333	50.333	58.083

Table 2: Success rates (in terms of the number of successful applications out of 100) calculated on the basis of an error margin of 15%.

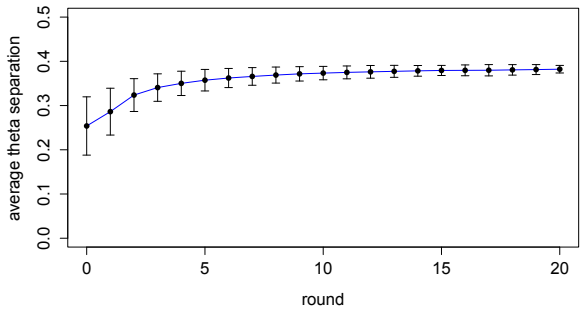
Instance	Model using the new phase shift function								Original model by Aihara et al.							
	No relevance				Relevance				No relevance				Relevance			
	0.25	0.5	0.75	1.0	0.25	0.5	0.75	1.0	0.25	0.5	0.75	1.0	0.25	0.5	0.75	1.0
cycle-10.gph	55	65	70	66	20	58	54	68	67	79	72	77	29	67	70	78
cycle-3.gph	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100
cycle-4.gph	89	94	98	100	65	81	90	96	100	100	100	100	89	99	98	100
houseoftriangles-3.gph	50	37	31	24	68	68	54	56	0	0	0	0	73	32	11	1
line-10.gph	75	100	100	100	20	52	91	100	76	100	100	100	35	66	97	100
line-2.gph	100	100	100	100	100	100	100	100	100	100	100	100	99	100	100	100
line-3.gph	99	100	100	100	99	100	100	98	100	100	100	98	99	100	100	99
petersen.gph	10	2	1	0	22	10	8	6	0	0	0	0	14	0	0	0
rectriangle-9.gph	100	100	100	100	100	100	100	100	100	100	100	100	93	100	100	100
spare.gph	80	88	90	93	23	2	0	0	0	0	0	0	0	0	0	0
star-8.gph	99	100	100	100	0	100	99	94	98	98	99	100	0	79	97	98
wheel-8.gph	95	98	98	100	80	87	82	87	93	99	100	100	85	49	28	19
averages	79.333	82.000	**82.333**	81.917	58.083	71.500	73.167	75.417	69.500	**73.000**	72.583	72.917	59.667	66.000	66.750	66.250

Table 3: Average distance to the optimal distribution of the theta values

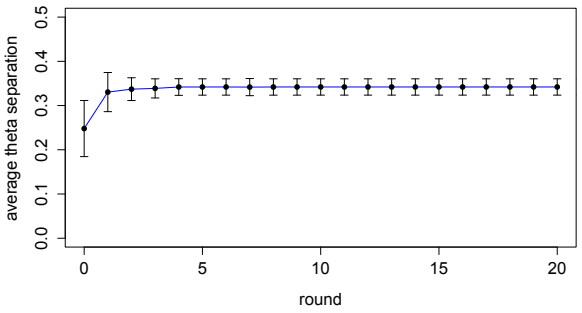
Instance	Model using the new phase shift function								Original model by Aihara et al.							
	No relevance				Relevance				No relevance				Relevance			
	0.25	0.5	0.75	1.0	0.25	0.5	0.75	1.0	0.25	0.5	0.75	1.0	0.25	0.5	0.75	1.0
cycle-10.gph	0.055	0.039	0.032	0.035	0.073	0.052	0.051	0.036	0.045	0.026	0.030	0.024	0.063	0.045	0.037	0.027
cycle-3.gph	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
cycle-4.gph	0.032	0.016	0.005	0.000	0.095	0.051	0.027	0.011	0.005	0.001	0.000	0.000	0.039	0.007	0.007	0.001
houseoftriangles-3.gph	0.028	0.032	0.036	0.042	0.022	0.021	0.025	0.024	0.064	0.070	0.070	0.071	0.019	0.032	0.044	0.050
line-10.gph	0.028	0.012	0.006	0.004	0.062	0.035	0.022	0.014	0.028	0.013	0.007	0.004	0.048	0.032	0.021	0.015
line-2.gph	0.002	0.000	0.000	0.000	0.002	0.000	0.000	0.000	0.002	0.000	0.000	0.000	0.007	0.000	0.000	0.000
line-3.gph	0.006	0.001	0.000	0.000	0.013	0.004	0.002	0.002	0.004	0.002	0.000	0.005	0.015	0.005	0.002	0.002
petersen.gph	0.039	0.044	0.045	0.046	0.035	0.039	0.040	0.041	0.047	0.050	0.050	0.050	0.034	0.046	0.047	0.048
rectriangle-9.gph	0.000	0.000	0.000	0.000	0.004	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.007	0.001	0.000	0.000
spare.gph	0.020	0.019	0.018	0.018	0.029	0.029	0.028	0.027	0.036	0.042	0.045	0.048	0.043	0.053	0.057	0.060
star-8.gph	0.004	0.001	0.001	0.000	0.061	0.025	0.016	0.019	0.006	0.003	0.001	0.000	0.093	0.036	0.019	0.014
wheel-8.gph	0.010	0.009	0.009	0.009	0.016	0.016	0.017	0.016	0.019	0.020	0.020	0.020	0.016	0.022	0.027	0.030
averages	0.019	0.014	**0.013**	0.013	0.034	0.023	0.019	0.016	0.021	0.019	0.019	**0.019**	0.032	0.023	0.022	0.021

Table 4: Average number of communication rounds executed before reaching the stopping condition.

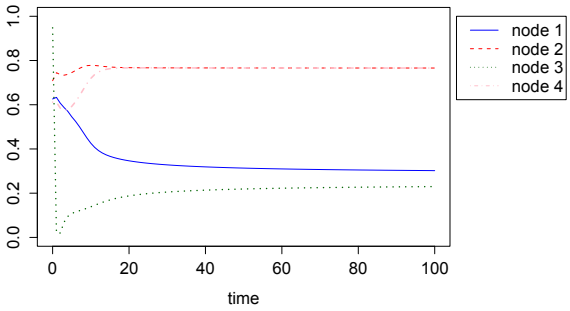
Instance	Model using the new phase shift function								Original model by Aihara et al.							
	No relevance				Relevance				No relevance				Relevance			
	0.25	0.5	0.75	1.0	0.25	0.5	0.75	1.0	0.25	0.5	0.75	1.0	0.25	0.5	0.75	1.0
cycle-10.gph	37.230	24.170	15.910	11.470	46.990	35.940	29.490	22.990	45.580	28.260	19.950	15.050	52.640	45.090	34.550	28.940
cycle-3.gph	10.740	6.300	4.680	4.850	17.490	10.860	8.100	6.340	23.020	14.170	10.040	8.670	31.300	20.940	16.460	14.470
cycle-4.gph	13.740	8.060	5.350	2.260	20.190	13.750	10.120	8.000	21.030	11.770	7.740	5.650	31.050	19.420	13.830	11.140
houseoftriangles-3.gph	22.500	13.830	10.290	7.880	34.540	30.340	27.450	23.130	55.210	36.520	23.370	19.060	46.950	51.070	51.380	49.060
line-10.gph	49.460	38.040	30.750	24.330	51.500	53.260	49.550	37.540	51.000	44.050	33.190	27.700	49.580	51.190	47.260	44.360
line-2.gph	8.750	5.350	3.740	2.000	8.520	5.170	3.610	2.000	10.200	6.030	4.400	3.330	9.980	6.050	4.450	3.330
line-3.gph	12.840	7.600	5.130	2.490	19.960	13.160	9.360	7.190	16.320	9.600	6.920	4.740	24.810	14.550	11.490	8.850
petersen.gph	18.430	10.910	7.860	6.690	34.230	23.690	18.550	15.040	33.290	24.430	19.820	19.480	48.520	38.320	33.030	29.530
rectriangle-9.gph	30.180	18.440	14.280	9.910	32.340	35.350	36.010	32.930	41.210	37.310	27.560	25.270	32.190	25.950	27.900	29.500
spare.gph	12.150	7.540	5.480	5.090	23.410	15.290	11.840	10.090	26.780	19.120	16.910	14.640	43.780	33.390	26.980	23.230
star-8.gph	16.240	9.060	5.920	3.090	55.230	41.250	32.030	25.340	22.360	12.990	9.170	7.110	63.430	57.220	43.260	36.430
wheel-8.gph	18.040	10.730	8.110	6.920	37.740	30.200	26.200	21.330	51.910	37.710	29.130	20.250	48.780	50.020	43.990	39.850
averages	20.858	13.336	9.792	**7.248**	31.845	25.688	21.859	17.660	33.159	23.497	17.350	**14.246**	40.251	34.434	29.548	26.558



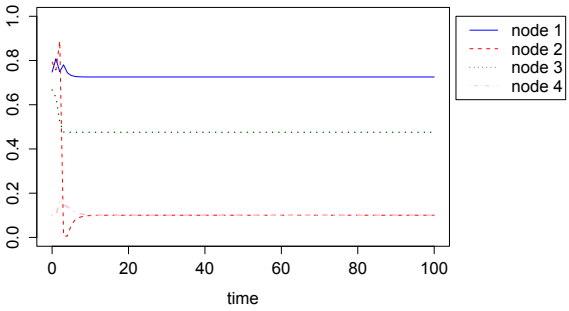
(a) Original model by Aihara et al.



(b) Model using the new phase shift function



(c) Original model by Aihara et al.



(d) Model using the new phase shift function

Figure 4: Experimental results concerning topology *spare.gph*. (a) and (b) show the evolution of the average distance between the theta values of connected nodes. While (a) refers to the original model, (b) concerns the model using the new phase shift function. (c) and (d) show—for a representative run of the system—the evolution of the four theta values. The graphic in (c) represents the behavior of the original model, while (d) shows the behavior of the model using the new phase shift function.