Bio-inspired Approach to Time Synchronization in Multi-Agent System

Iva Bojic and Mario Kusek

University of Zagreb, Faculty of Electrical Engineering and Computing, Unska 3, HR-10000, Zagreb, Croatia {iva.bojic, mario.kusek}@fer.hr

Summary. Multi-Agent Systems (MASs) are very dynamic since agents are entering and leaving system frequently. In those systems self-organization without centralized control is crucial for efficient system operations, such as time synchronization. To accomplish self-organized time synchronization in MAS, bio-inspired approach observed in flashing fireflies, is used. Fireflies are known to emit flashes at regular intervals when isolated, while in the group their pulses converge upon the same rhythm until they synchronize. This paper investigates the influence of topology and metric on the synchronization.

1 Introduction

In distributed systems, such as Multi-Agent Systems (MASs), a time synchronization is a prerequisite for many processes: maintaining the consistency of distributed data, checking the authenticity of a request sent to the server, eliminating the processing duplicate updates, etc.

Time synchronization can be achieved using different approaches. The bio-inspired approach that will be presented in this paper is based on a phenomenon of self-organized fireflies flashing. Firefly-based time synchronization is inspired by nature where male fireflies gather on trees and start emitting flashes regularly. Over time synchronization emerges from a random situation, making it seem as though the whole tree is flashing in perfect synchrony.

Self-organization, a process in which patterns at the global level of a system emerge solely from numerous interactions among lower-level components in the system [5], can be widely used in MAS [13]. The important properties of self-organization, which are of the vital importance in MASs, are robustness, scalability and adaptability to new situations (e.g. agents mobility).

The rest of the paper is organized as follows. Section 2 presents firefly model for time synchronization. Section 3 gives a short introduction in a field of related work, while Section 4 explains our approach. Furthermore, Section 4 presents simulation results. And finally, Section 6 concludes the paper.

2 Firefly synchronization model

Fireflies synchronization in MAS is modeled using the theory of coupled oscillators and can be described with a following set: $FS_{MAS} = \{A, Z, P, T, M\}$. Set $A = \{a_\imath \mid \imath \in \mathbb{N}^+\}$ denotes a set of agents such that $|A|$ (i.e. cardinality of a set A) is equal to the number of agents in MAS. Set $Z = \{z_\imath \mid \imath \in \mathbb{N}^+; z_\imath \in [0,1]\}$ denotes a set of voltage-like *state variables* $z_\imath$. Furthermore, set $P = \{(x_\imath, y_\imath) \mid \imath \in \mathbb{N}^+; x_\imath, y_\imath \in \mathbb{Q}^+\}$ denotes a set of coordinates for each agent $a_\imath$. Finally, set $T = \{$*fully-meshed, line, mesh, ring, star*$\}$ denotes different topologies, while set $M = \{$*Chebyshev, Euclidean, Mahalanobis, Manhattan*$\}$ denotes a set of different metrics used for distance calculation.

A theoretical framework for the convergence to synchrony was given by Mirollo and Strogatz (M&S) in 1990 [10] where an oscillator represents agent's internal clock dictating when to flash:

$$z_\imath^{'} = f_\imath\left(z_\imath\right) + \sum_{\jmath=1}^{|A|} \varepsilon_{\imath\jmath}\ g_{\imath\jmath}(a_\imath, a_\jmath)\ \delta\left(t - t_\jmath^*\right), \tag{1}$$

where $z_\imath^{'}$ denotes first derivation of $z_\imath$ over a time of duration Δt, $f_\imath\left(z_\imath\right)$ describes the dynamics of the $a_\imath$-*th* oscillator, $\varepsilon_{\imath\jmath}$ is the small coupling constant, $g_{\imath\jmath}(a_\imath, a_\jmath)$ is the coupling function between oscillators $a_\imath$ and $a_\jmath$, δ is Dirac delta function and $t_\jmath^*$ is the firing time of the $a_\jmath$-*th* oscillator.

When $z_\imath$ reaches the upper boundary (e.g. value is equal to 1), the $a_\imath$-*th* oscillator "fires" and then $z_\imath$ jumps back to the lower boundary (e.g. value is equal to 0). State variable $z_\imath$ does not only depend on its previous state, but also is adjusted upon reception of a pulse from other oscillators, making it possible that over time, pulses from different oscillators are transmitted simultaneously. Namely, $z_\jmath$ increments $z_\imath$ by an amount given by $\varepsilon_{\imath\jmath}\ g_{\imath\jmath}(a_\imath, a_\jmath)$:

$$z_\jmath = 1 \Rightarrow \begin{cases} z_\imath \rightarrow z_\imath + \varepsilon_{\imath\jmath}\ g_{\imath\jmath}(a_\imath, a_\jmath) & if\ \ z_\imath + \varepsilon_{\imath\jmath}\ g_{\imath\jmath}(a_\imath, a_\jmath) < 1 \\ z_\imath \rightarrow 0 & otherwise \end{cases}. \tag{2}$$

If the oscillators evolve according to identical uncoupled dynamics, we say

$$f_\imath\left(z\right) = f\left(z\right) > 0,\ \forall z \in [0,1]. \tag{3}$$

Moreover, equation (1) can be then rewritten as

$$z_\imath^{'} = f\left(z_\imath\right) + \sum_{\jmath=1}^{|A|} \varepsilon_{\imath\jmath}\ g(a_\imath, a_\jmath)\ \delta(t - t_\jmath^*), \quad z_\imath \in [0,1]\,. \tag{4}$$

The coupling function $g(a_\imath, a_\jmath)$ between agents $a_\imath$ and $a_\jmath$ is defined as:

$$g(a_\imath, a_\jmath) = \begin{cases} 1 & if\ agent\ a_\jmath\ is\ a\ neighbor\ of\ agent\ a_\imath \\ 0 & otherwise \end{cases}. \tag{5}$$

In order to calculate function $g(a_i, a_j)$ we have to find whether agent a_j is a neighbor of agent a_i. This calculation can be done using coordinates (x_i, y_i) and (x_j, y_j) from set P, type of topology from set T and type of metric from set M. Moreover, the small coupling constant ε_{ij}, calculated as:

$$\varepsilon_{ij} = d_{m \in M}(a_i, a_j) \tag{6}$$

denotes the importance of the influence that agent a_j has on agent a_i. This small coupling constant is equal to the distance between agents a_i and a_j in chosen metric $m \in M$. Since, types of topologies and metrics are important for the synchronization process, the rest of this section firstly describes several most common network topologies and algorithms used for defining such topologies, and then introduces used metrics.

2.1 Network topologies

Network topologies used for calculating neighborhood between agents are fully-meshed, line, mesh, ring and star topologies.

Fully-meshed topology

Fully-meshed topology is a topology where every agent is connected with every other agent in the system. Algorithm for getting fully-meshed topology from set of agents A is described in Algorithm 1.

Algorithm 1 Algorithm for calculating fully-meshed topology

1. **foreach** agent $a_i \in A$
2. **foreach** agent $a_j \in A$ **such that** $(a_i \neq a_j)$
3. $g\,[a_i, a_j] = 1$ *%% 1 denotes that agents are connected*
4. **end foreach**
5. **end foreach**

Line topology

Line topology is a topology where the first and the last agent have only one neighbor, unlike others that have two neighbors. The disadvantage of line topology is – if a failure of agent or connection between two agents occurs, two unconnected groups of agents will be formed.

Algorithm for defining a line topology from set of agents has two parts and is shown in Algorithm 2. First part of algorithm finds the first agent (*firstAgent*) in the line. Distance in steps 3. and 17. is calculated used Equations (7) – (10) from Section 2.2. The second part of algorithm determines connectivity between agents.

Algorithm 2 Algorithm for calculating line topology

%% first part

foreach agent $a_i \in A$
  foreach agent $a_j \in A$ **such that** $(a_i \neq a_j)$
    $fullDistance\,[a_i]$ += $d_{\,m \in M}(a_i, a_j)$
  end foreach
end foreach
foreach distance i in *fullDistance*
  if $(maxDistance < fullDistance[i])$
    $firstAgent = a_i$
    $maxDistance = fullDistance[i]$
  end if
end foreach

%% second part

$freeAgents \leftarrow A \setminus \{firstAgent\}$
$currentAgent = firstAgent$
$lineAgents$.add($currentAgent$)
while ($|\,freeAgents\,| > 0$)
  foreach agent $a_i \in freeAgents$
    $tmpDistance = d_{\,m \in M}(a_i, currentAgent)$
    if $(minDistance > tmpDistance)$
      $firstAgent = a_i$
      $minDistance = tmpDistance$
    end if
  end foreach
  $currentAgent = firstAgent$
  $lineAgents$.add($currentAgent$)
  remove *currentAgent* from *freeAgents*
end while
foreach agent a_i in *lineAgents*
  $g\,[a_i, a_{i+1}] = 1$ *%% 1 denotes that agents are connected*
end foreach

Mesh topology

Mesh topology is a topology where every agent has one or several agents for its neighbor(s) denoted with *numberForMesh* value. Unlike other mentioned overlay topologies, a mesh topology represents a directed graph in which each edge is given an independent orientation at each end. Algorithm for getting mesh topology is shown in Algorithm 3.

For example *mesh(3)* is topology where every agent has three neighbors (see Figure 1) and where a connection (thick line) between two nodes without an arrow is bidirected and a connection (dotted line) with an arrow on the end is directed. Therefore, neighbors of a_1 are a_2, a_3 and a_6, but a_1 is not a neighbor of a_3. Neighbors of a_3 are a_2, a_4 and a_5.

Algorithm 3 Algorithm for calculating mesh topology

1. **foreach** agent $a_i \in A$
2. **foreach** agent $a_j \in A$ **such that** $(a_i \neq a_j)$
3. *distance* $[a_j] = \{\ d_{m \in M}(a_i, a_j),\ a_j\ \}$
4. **end foreach**
5. *distance*.sort()
6. *counter* = *numberForMesh*
7. **while** (*counter* > 0)
8. a_j = *distance* [*numberForMesh* - *counter*] {1}
9. $g\,[a_i, a_j] = 1$ *%% 1 denotes that agents are connected*
10. *counter* - -
11. **end while**
12. **end foreach**

Ring topology

Ring topology is very similar to line topology. Only difference is that the first and the last agent are connected with additionally connection (see Figure 1). Algorithm for defining a ring topology has all steps from previous algorithm for defining line topology plus one additional step. After the algorithm for line finishes, this algorithm connects the first and the last agent in the line.

Star topology

Star topology is a topology with one central agent that has $|\,A\,| - 1$ neighbors, while other agents have only one neighbor: this central one. Algorithm for defining a star topology from a set of agents has three parts (see Algorithm 4). First of all a *centralAgent* in the system has to be determined, then second part is to determinate closest agent to the *centralAgent*. Finally, in the third part of algorithm *centralAgent* is being connected with all other agents.

Algorithm 4 Algorithm for calculating star topology

%% first part

1. *easternmost* = *westernmost* = x_0
2. *southernmost* = *northernmost* = y_0
3. **foreach** agent $a_i \in A$
4. **if** $(x_i >$ *easternmost*$)$ **then** *easternmost* $\leftarrow x_i$
5. **if** $(x_i <$ *westernmost*$)$ **then** *westernmost* $\leftarrow x_i$
6. **if** $(y_i >$ *southernmost*$)$ **then** *southernmost* $\leftarrow y_i$
7. **if** $(y_i <$ *northernmost*$)$ **then** *northernmost* $\leftarrow y_i$
8. **end foreach**
9. *centralPointX* = (*westernmost* - *easternmost*) / 2
10. *centralPointY* = (*southernmost* - *northernmost*) / 2

%% second part

foreach agent $a_i \in A$ **such that** $(a_i \neq centralAgent)$
  $tmpDistance = d_{m \in M}(a_i, centralAgent)$
  if $(minDistance > tmpDistance)$
    $centralAgent = a_i$
    $minDistance = tmpDistance$
  end if
end foreach

%% third part

foreach agent $a_i \in A$ **such that** $(a_i \neq centralAgent)$
    $g\,[a_i, centralAgent] = 1$ *%% 1 denotes that agents are connected*
end foreach

2.2 Metrics

In this paper metrics are calculated using equations for calculation of Chebyshev, Euclidead, Mahalanobis and Manhattan distances. Metrics are used for the two separate purposes:

- to calculate a real distance between two agents in the network used in algorithms for topology calculation and
- to determinate the influence that one agent has on the other agents used for ε_{ij} calculations.

Chebyshev distance

The Chebyshev distance is defined on a vector space where distance between two agents is the greatest of their differences along any coordinate dimension. Mathematical expression for Chebyshev metric is:

$$\varepsilon_{ij} = d_c(a_i, a_j) = \max\left(\mid x_i - x_j \mid, \mid y_i - y_j \mid\right) \tag{7}$$

Euclidean distance

The Euclidean distance between two agents a_i and a_j is the length of the line segment connecting them. In mathematical expression Euclidean distance looks like:

$$\varepsilon_{ij} = d_e(a_i, a_j) = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2} \tag{8}$$

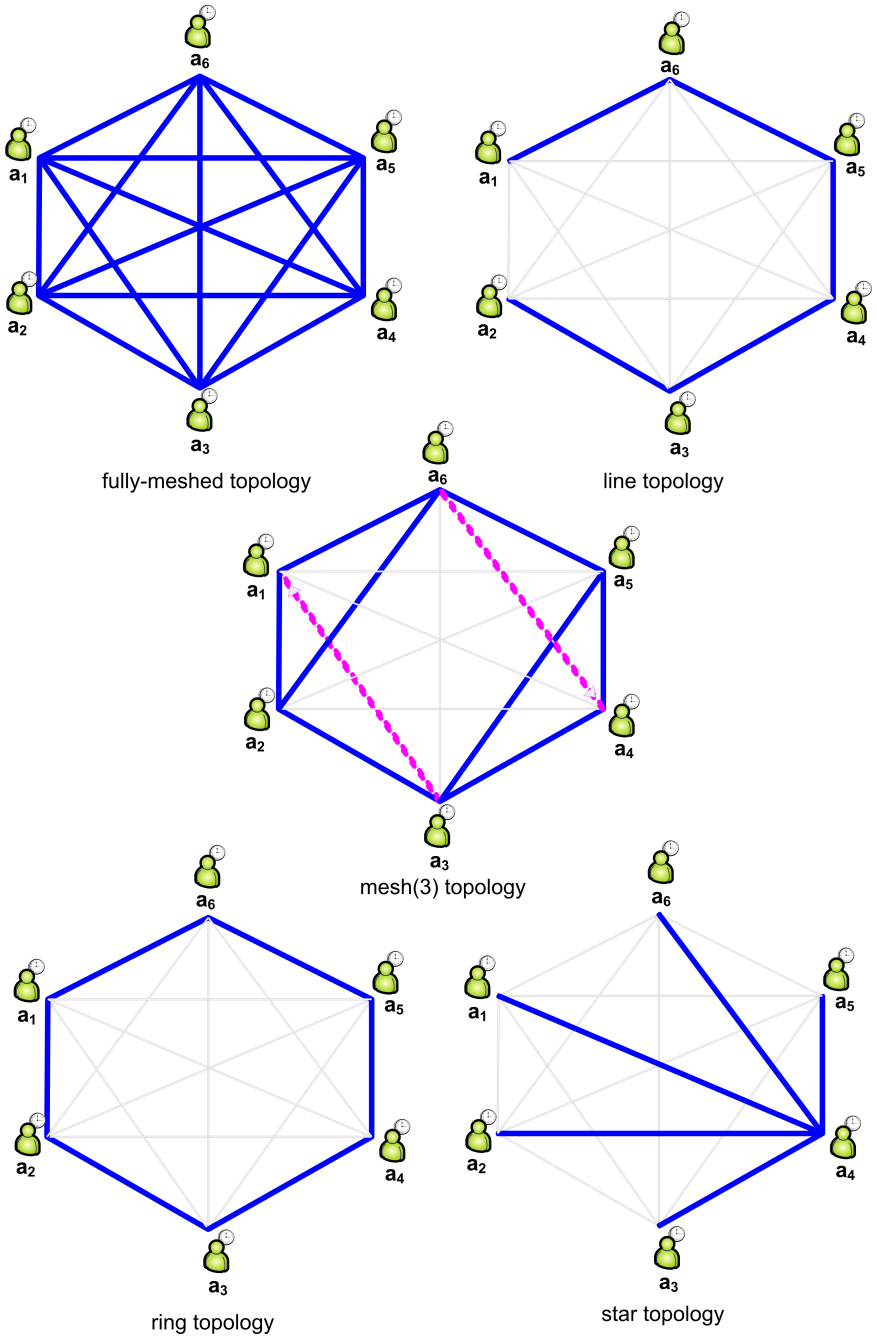

Fig. 1. Example for different topologies

Mahalanobis distance

The Mahalanobis distance is based on a correlation between agents by identifying and analyzing different patterns. It is calculated with next mathematical σ expression:

$$\varepsilon_{ij} = d_h(a_i, a_j) = \sqrt{\frac{x_i - y_i}{\sigma_1^2} + \frac{x_j - y_j}{\sigma_2^2}} \tag{9}$$

Manhattan distance

The Manhattan distance or taxicab geometry is based on calculating distance between two agents as sum of absolute differences of their coordinates. Mathematical expression for Manhattan metric is:

$$\varepsilon_{ij} = d_n(a_i, a_j) = \mid x_i - x_j \mid + \mid y_j - y_j \mid \tag{10}$$

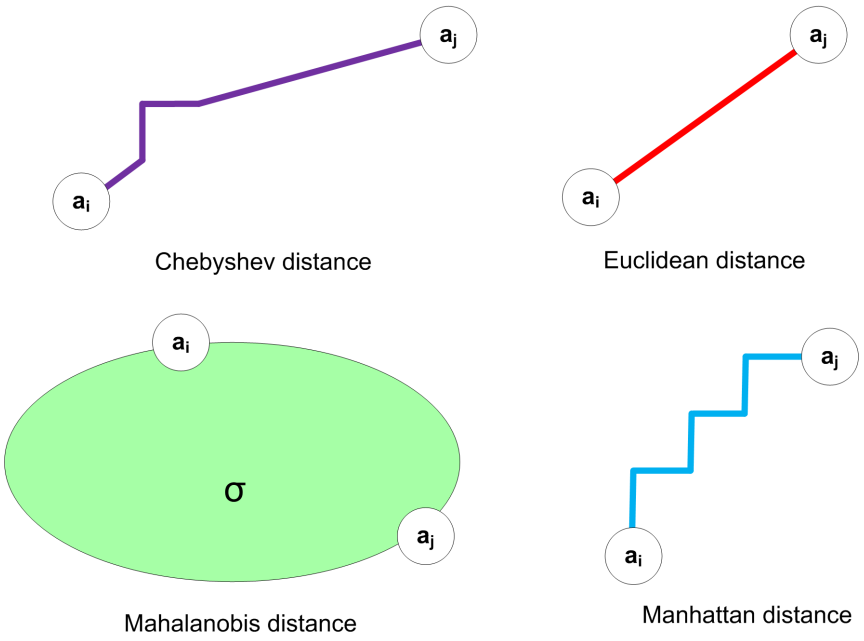

Fig. 2. Example for different metrics

3 Related work

The M&S model [10] a theoretical framework for the convergence to synchrony in fully-meshed networks. In 2004 Lucarelli and Wang (L&W) [8] showed that the algorithms developed by M&S would converge for any connected topology (i.e. fully-meshed assumption was released to a mesh). Hence, the M&S and L&W models differ on the physical connectivity layer. The M&S model considers that there exists physical connectivity among all agents in the system, while the L&W model specifies the interconnection or topology of the network with a graph in which edges join neighboring nodes.

In our work, we use the L&W model investigating how different overlay network topologies and metrics affect percentage of successful synchronization in MAS (i.e. which parameter has a greater impact on synchronization success). The choice of overlay network topology determines the way agents communicate (i.e. are connected), while usage of different metric denotes the intensity of influence that connected agents have on each other. These two parameters have already been studied in related work, but to our knowledge, no previous study has compared their mutually effect on synchronization process.

Usage of firefly synchronization can be found in various algorithms (e.g. geographic routing algorithms [11] and the Reachback Firefly Algorithm (RFA) [15]), protocols (e.g. gossip protocols [16]) or data gathering mechanisms (e.g. data gathering scheme in sensor networks [14]). All of the aforementioned examples of firefly synchronization assume partly connected networks and that there is no difference between agents (i.e. $\varepsilon_{ij} = \varepsilon$).

In every step of geographic routing [11], a node can communicate only with its k-neighbors, referred to as the Connected k-Neighborhood (CKN) problem. This k-connectivity is also used in [2], where for every node, k-neighbors are selected randomly. In the RFA four different topologies have been investigated: fully-meshed and mesh [15], chain [7] and ring [6]. In chain topology nodes are ordered in a chain and can only communicate with their immediate neighbors, while the ring topology was simulated using asynchronous communication patterns, i.e. unidirectional communication links.

Although, in most of related work equal *credibility* is assumed for all agents (i.e. the influence agents have on each other is always the same), there are some projects where small coupling constant ε_{ij} is different for different agents. In [12] authors propose that agent's credibility depends on the degree-based weighting. Being influenced by more agents means that agent is likely to be more reliable. Furthermore, in [1] authors assumed that all the agents coupling credibility stayed constant and were distributed in a close interval. They proved the following theorem:

Theorem 1. *Synchronization condition of two coupled oscillators Given two oscillators a_i and a_j with their coupling strengths satisfying $\varepsilon_{ij} \neq \varepsilon_{ji}$ they will achieve synchronization.*

In our previous work [3, 4] we calculated the small coupling constant ε_{ij} using the *Euclidean* metric. In this paper we extended the usage of *Euclidean* metric to the *Chebyshev*, *Mahalanobis* and *Manhattan* metrics. The following example shows how these calculations can be made. Assume there are five agents $A = \{a_1, a_2, a_3, a_4, a_5\}$ in MAS. Their coordinates are given by $P = \{(2, 3), (7, 26), (27, 18), (13, 13), (20, 5)\}$. Chosen topology is *star* (explain in Section 2.1) and metric *Euclidean* given by Equation (8) from Section 2.2.

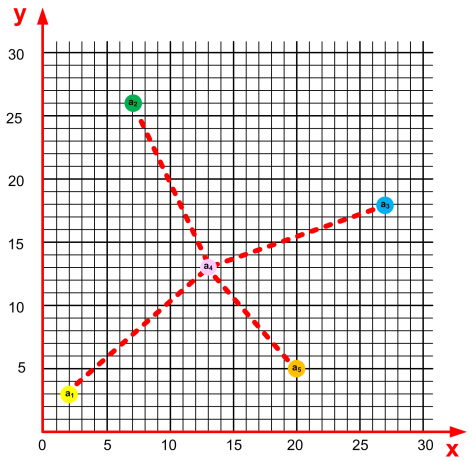

Fig. 3. Example for ε_{ij} value calculation

Red thick dashed line in Figure 3 presents communication channels between agents. For example, agents a_1 and a_3 cannot communicate (i.e. they are not directly connected), while agents a_2 and a_4 can (i.e. between them exists red line – communication channel). After we have determined whether the small coupling constant ε_{ij} is different from zero (i.e. agents are directly connected), we have to calculate real values for the constant. This can be achieved by calculating the distance between connected agents using Equation (8) (see the following table).

Table 1. Values for the small coupling constant ε_{ij}

	a_1	a_2	a_3	a_4	a_5
a_1	0	0	0	14.87	0
a_2	0	0	0	14.32	0
a_3	0	0	0	14.87	0
a_4	14.87	14.32	14.87	0	10.63
a_5	0	0	0	10.63	0

4 Simulation of firefly synchronization in MAS

The simulation is run using the Multi-Agent Simulator of Neighborhoods (MASON) – a single-process discrete-event simulation core and visualization toolkit written in Java. In this paper, we will not explain how agents activity is scheduled in the simulator, nor other details about the simulation process, since it can be find in MASON's documentation [9]. In simulation, the behavior of each agent is governed by the following rules [5]:

- each firefly has an intrinsic flashing frequency determined with threshold parameter, and when left alone it will flash at periodic intervals;
- the flashes are timed by the progressive excitation within each firefly; the excitation increases until it reaches a threshold, at which point a flash is emitted and the excitation is reset to zero;
- if a firefly senses a certain amount of luminescence from its neighbors, it will reset its excitation to zero in order to flash simultaneously with those neighbors in the future; but, if the excitation is close enough to the flashing threshold, the flash has already been started and will proceed as planned even though the excitation is reset to zero; this is determined with buffer parameter.

Simulation process of firefly synchronization can be divided into two behaviors – passive and active one (see Algorithms 5 and 6). In active behavior each agent sends its flash-message to its neighbors when the value of its *excitation* reaches *threshold* value. Excitation evolution is governed by the following equation $z_i^{'} = f_i(z_i)$. Mirollo and Strogatz demonstrated that synchronization can be achieved when using a monotonic and concave down function [10]. Finally, neighborhood of each agent can be calculated using algorithms for topologies and metrics described in Section 2.

Algorithm 5 The simulation skeleton: active behavior

```
1. while (true)
2.    wait until (excitation == threshold)
3.    neighbors ← findNeighbors()
4.    send flash to all neighbors
5. end while
```

In passive behavior each agent first receives message and then processes it by sensing a certain amount of luminescence from its neighbors. If that amount of luminescence is larger than a *trigger* and agent's excitation is smaller than *buffer*, agent will flash and reset its excitation to zero. Therefore, the *trigger* sets the amount of luminescence required for a firefly to reset its excitation prematurely, while *buffer* prevents it to happen too often.

Algorithm 6 The simulation skeleton: passive behavior

```
1.  while (true)
2.    receiveFlash()
3.    neighbors ← findNeighbors()
4.    sumLights = 0;
5.    foreach agent a_i in neighbors such that (a_ı ≠ a_me)
6.      sumLights += ε_ı me * light(a_ı)
7.    end foreach
8.    if (sumLights > trigger && excitation < buffer)
9.      excitation = 0
10.   end if
11. end while
```

The simulation parameters include *numberFireflies*, *threshold* and *buffer*. The *numberFireflies* sets the number of fireflies in the simulation. The *threshold* sets the excitation threshold at which point a agent will flash and reset its excitation to zero. The *buffer* sets how many time steps are necessary for the flashing signal to evolve and terminate in a flash. If an agent is triggered to reset its excitation when the excitation is within "buffer" of the threshold, the flash will proceed as planned despite the resetting.

In our simulations, the *numberFireflies* parameter was set to 10 (i.e. we have small overlay networks with a maximum of up to 10 nodes). To determine the *threshold* and *buffer* values, in our previous work [4] we made two series of simulation experiments where we determined that the *threshold* value should be equal to 20 and the *buffer* value to 2.

5 Simulation results

In our previous work [3, 4] we used only *Euclidean* metric with combination of different topologies (e.g. star, line, mesh, fully-meshed, ring) and have concluded that best topologies on condition of successful synchronization are fully-meshed, mesh and star topologies. MAS is said to achieve synchronicity when all agents flash simultaneously.

In this work we extend our previous results adding additional metrics (*Chebyshev*, *Mahalanobis* and *Manhattan*). Thus, for testing influence of topologies and metrics on synchronization process we used combinations of all metrics and all topologies (i.e. 40 combinations). Evaluation metric is the same as in previous papers [3, 4]:

- **successful synchronization**: percentage of successful synchronization;
- **network traffic**: denotes the amount of messages sent between agents during the synchronization process and
- **time to sync**: denotes the time (discrete time units $\triangle t$) until all agents have entered the synchronization state.

Table 2. Simulation results - successful synchronization in %

		Metrics			
		Chebyshev	Euclidean	Mahalanobis	Manhattan
Topologies	Fully-meshed	94%	97%	94%	96%
	Line	70%	69%	72%	72%
	Mesh(3)	74%	78%	78%	84%
	Mesh(4)	77%	84%	84%	89%
	Mesh(5)	91%	93%	93%	93%
	Mesh(6)	89%	92%	92%	93%
	Mesh(7)	88%	94%	94%	96%
	Mesh(8)	96%	93%	93%	95%
	Ring	71%	70%	71%	72%
	Star	94%	94%	94%	94%

First we investigate how different metrics influence on percentage of successful synchronization and then we investigated which parameter is more important: overlay topology or metric (see Table 2). Results in Table 2 show that Manhattan distance is resulting with best result of successful synchronization. For mesh(3) topology, when using Manhattan distance, results are about 10% better than when using Chebyshev metric.

Table 3. Simulation results - time to sync/network traffic

		Metrics			
		Chebyshev	Euclidean	Mahalanobis	Manhattan
Topologies	Fully-meshed	47/217	42/198	47/217	43/202
	Line	128/83	128/83	128/83	128/83
	Mesh(3)	64/76	64/77	64/77	68/83
	Mesh(4)	54/92	56/97	56/97	56/98
	Mesh(5)	54/124	54/123	54/123	53/120
	Mesh(6)	49/139	48/136	48/136	46/130
	Mesh(7)	47/159	46/157	46/157	46/158
	Mesh(8)	49/197	45/183	45/183	44/175
	Ring	95/70	95/70	95/70	95/70
	Star	40/29	40/29	40/29	40/29

Furthermore, from the same table it can be concluded that choice of network overlay topology has greater influence on percentage of successful synchronization than the chosen metric. For example, the most significant difference is between fully-meshed and ring topology (28%), while the biggest difference between metrics is only 12% (for Chebyshev and Manhattan metric in mesh(4) topology).

Finally, we measure the number of time units (steps) until the system achieves synchronicity and the network traffic (number of exchanged messages) considering the usage of different combinations of network topologies and metrics (Table 3). We can conclude that the choice of metric does not have significant impact on those two parameters.

6 Conclusion

In this paper we proposed a synchronization scheme for using swarm-intelligence based on firefly behavior. The system consists of firefly agents placed at each node in the MAS. These agents are responsible for synchronization and coordination processes in the system.

Presented fireflies model was tested with different overlay topologies and metrics used for neighborhood determination in order to find the most suitable combination. From results presented in this work, we can conclude that choice of overlay topology is more important than choice of used metric. This means that the way agents communicate (i.e. are connected) is more important than the intensity of influence that connected agents have on each other (that is determinate by the choice of metric).

However, there are few differences between our simulation and real world scenarios. Simulation conducted in our work does not support message delay as neither the possibility that message can be lost which happens in the real world. Also the firefly in the real world cannot transmit and receive flash at the same time and this is possible in the simulation.

Therefore, in our future work we will make a series of experiments in real word environment where messages between communicating agent can be lost or can be delayed. In those cases we will adopt some changes in our algorithm in order to have high percentage of successful synchronizations and still reasonable simple synchronization protocol that does not influence on other process in MAS.

Acknowledgements.

The authors acknowledge the support of research project "Content Delivery and Mobility of Users and Services in New Generation Networks" (036-0362027-1639), funded by the Ministry of Science, Education and Sports of the Republic of Croatia.

References

1. An, Z., Zhu, H., Zhang, M., Xu, C., Xu, Y., Li, X.: Linear pulse-coupled oscillators modela new approach for time synchronization in wireless sensor networks. Advances in Complex Systems 2(2), 108–114 (2010)
2. Babaoglu, O., Binci, T., Jelasity, M., Montresor, A.: Firefly-inspired Heartbeat Synchronization in Overlay Networks. In: 1st IEEE international conference on Self-Adaptive and Self-Organizing Systems. pp. 77–86 (2007)
3. Bojic, I., Kusek, M.: Fireflies Synchronization in Small Overlay Networks. In: Proceedings of 32nd International Conference on Information and Communication Technology, Electronics and Microelectronics. pp. 27–32 (2009)
4. Bojic, I., Podobnik, V., Ljubi, I., Jezic, G., Kusek, M.: A self-optimizing mobile network: Auto-tuning the network with firefly-synchronized agents. Information Sciences 182(1), 77–92 (2012)
5. Camazine, S., Deneubourg, J.L., Franks, N., Sneyd, J., Theraula, G., Bonabeau, E.: Self-Organization in Biological Systems. Princeton University Press (2003)
6. Leidenfrost, R., Elmenreich, W.: Establishing wireless time-triggered communication using a firefly clock synchronization approach. In: International Workshop on Intelligent Solutions in Embedded Systems. pp. 1–18 (2008)
7. Leidenfrost, R., Elmenreich, W.: Firefly clock synchronization in an 802.15.4 wireless network. EURASIP Journal on Embedded Systems pp. 7:1–7:17 (2009)
8. Lucarelli, D., Wang, I.J.: Decentralized synchronization protocols with nearest neighbor communication. In: Proceedings of the 2nd international conference on Embedded networked sensor systems. pp. 62–68 (2004)
9. MASON web site: http://www.cs.gmu.edu/ eclab/projects/mason/
10. Mirollo, R.E., Strogatz, S.H.: Synchronization of pulse-coupled biological oscillators. SIAM Journal on Applied Mathematics 50(6), 1645–1662 (1990)
11. Nath, S., Gibbons, P.: Communicating via fireflies: geographic routing on duty-cycled sensors. In: Proceedings of the 6th international conference on Information Processing in Sensor Networks. pp. 440–449 (2007)
12. Scholtes, I., Botev, J., Esch, M., Sturm, P.: Epidemic self-synchronization in complex networks of kuramoto oscillators. Advances in Complex Systems 13(1) (2010)
13. Sudeikat, J., Renz, W.: Engineering environment-mediated multi-agent systems. chap. Toward Systemic MAS Development: Enforcing Decentralized Self—organization by Composition and Refinement of Archetype Dynamics, pp. 39–57. Springer-Verlag (2008)
14. Taniguchi, Y., Wakamiya, N., Murata, M.: A distributed and self-organizing data gathering scheme in wireless sensor networks. In: Proceedings of 6th Asia-Pacific Symposium on Information and Telecommunication Technologies. pp. 299–304 (2005)
15. Werner-Allen, G., Tewari, G., Patel, A., Welsh, M., Nagpal, R.: Firefly-Inspired Sensor Network Synchronicity with Realistic Radio Effects. In: Proceedings of the 3rd international conference on Embedded Networked Sensor Systems. pp. 142–153 (2005)
16. Wokoma, I., Liabotis, I., Prnjat, O., Sacks, L., Marshall, I.: A Weakly Coupled Adaptive Gossip Protocol for Application Level Active Networks. In: Proceedings of the 3rd international Workshop on Policies for Distributed Systems and Networks. pp. 244–247 (2002)