ADCAIJ: Advances in Distributed Computing and Artificial Intelligence Journal
Regular Issue, Vol. 7 N. 4 (2018), 53-64
eISSN: 2255-2863
DOI: http://dx.doi.org/10.14201/ADCAIJ2018745364

Network Synchronization and Consensus Regions in a Multi-Agent System

Nuria de la Parra[b], Guillermo Reguera[a], Juan José Salvo[b], and Óscar Sánchez[b]

[a] CM (guiller98reguera@gmail.com)
[b] University of Salamanca (parra.nuria98@usal.es, jjsalvo@usal.es, oscar.sanchez@usal.es)

KEYWORD

Network system; Consensus; Network synchronization; Multi-agent system

ABSTRACT

Two problems related to complex network systems are explained in this paper. These two main problems are: consensus and network synchronization. Both have been studied separately in previous years but only a few people have studied these two problems together as one. Our intention in this article is to show how these two problems might 'work' together by simulating a network system with routers and agents, we will try to show how agents must negotiate and communicate in order to achieve their aim.

1. Introduction

A multi-agent system is considered a system with many agents that interact which each other to solve problems. In this sort of systems, every agent has its own aims but in order to achieve them it must communicate with the other agents that are part of the system. These systems are usually used to solve the type of problems that are often difficult or even impossible to solve for an agent on its own.

Multi-agent systems (MAS) are used in many disciplines, for instance, Multi-agent systems can be used to optimize the dynamic flow of vehicle traffic (Khatua *et al.,* 2011). MAS can also be used in fields such as online shopping webs, in order to guide the user during their purchase (Lee *et al.,* 2002). Related to that, multiagent systems are usually used when a shopping web provides personalized recommendations based on the user most visited products (González-Briones *et al.,* 2018). Furthermore, multiagent systems can also be used in the field of energy optimization due to their capacity for the communication, coordination and cooperation of agents among other things. In addition, multi-agent systems can be used in videogames, for example, in online role-playing games where there will be a huge number of players playing at the same time (Okresa Duric *et al.,* 2015). Real-time crowds are used in many videogames and many times these crowds (as well as some of them in films) are created by using multi-agent systems.

Related to the processing of images we have found an interesting article in which the authors use a multi-agent system to acquire, preprocess and automaticaly classify human images based on age and gender which might be useful in marketing or even in the adaptation of content for Smart TVs (González-Briones *et al.,* 2018). Moreover, MAS can be used in many topics related to medicine,for instance, it can be used in order to schedule the different tasks performed on a hospitalised patient. Talking about this field, MAS can as well be

used to manage the coordination of organ transplants between hospitals (Moreno A. 2003). Strangely enough, we can see the presence of MAS in GPS systems (Harding G. 2015) or even in films and tv shows like The Walking Dead. When in this show there is a huge horde of walkers, they use MAS to recreate them (Massive software) (Sáez Encinar, 2015).

Our work is inspired by the work of Feng *et al* (2018), titled *"Consensus of multi-agent systems with fixed inner connections"*. In this paper, the authors talk about two problems of coordination in systems they centre their attention in discussing network synchronization and consensus. Consensus means that a team of agents reach an agreement by interacting with each other through a communication network (Feng *et al.,* 2018). The authors chose to mix this issues because, as they explain, there are only a few articles who have done research of this two problems as one. They show how both issues have been studied separately not only in articles related to multi-agent systems but also in other disciplines.

Network synchronization has been studied in the MAS' field thoroughly with many different applications, but it also has been studied in other fields. For instance, they talk about network synchronization being studied in fields like electric power grids and wireless communication networks to which we can add that, for example, it is also studied in hippocampal neurons by seeing neurons as a network (Penn *et al.,* 2016). On the other hand, talking about consensus they say that it has been studied in fields like satellite formation, sensor networks, etc. To which we can add that consensus is as well important in multi-robot networks (Khoo *et al.,* 2009). We can add that, for instance, consensus is fundamental to the operation of distributed systems such as social networks, like it is proposed in (Mossel and Schoenebeck, 2009) they show this by an example where the agents must all agree on one of two otherwise indistinguishable colours.

It is explained that the problem of consensus has been studied in the MAS' area from different approaches since a studied carried out in (Ren, Beard, 2005) where they expanded the (Olfati *et al.*, 2004) results on consensus problems for networks of dynamic agents with fixed and switching topologies. They expanded it by using connected graphs.

Focusing on the main topic of the article, network synchronization and consensus as one issue, we can see how they centre their attention on a mathematical analysis considering and we quote: "[…] a coupled network of N agents where the physical connections between agents are represented by solid lines while the controllers are connected via communication links shown by dashed lines. Assume that communication links do not have any bandwidth limitation, data loss, or network delays." As a result of their research and theoretical demonstrations, they have achieved to propose a new framework where "the notions of synchronization region in complex networks and consensus region in multi-agent systems have been unified." (Feng *et al.,* 2018). The article is structured as it follows: first, other articles are revised. After that, a proposal of architecture is exposed. Following this, experimental results are shown. Finally, the program is explained, and conclusions are presented.

2. State of the Art

In recent times, there have been studies about different topics in multi-agent systems. With the rapid growth of the Internet and its use, multiagent systems are used more and more in order to achieve better solutions for complex problems.

Due to the complexity of the chosen article, we have decided to make an architecture proposal based on a more general theoretical knowledge and not a mathematical approach. In our investigation we have focused on finding articles related to network synchronization and consensus since they are our aim. According to what is written in the article "`Network synchronization`", network synchronization plays an important role in many different applications, such as computer communication, control systems, position determination, etc.

The authors explore topics like network stability or the nonlinear behaviour of synchronization systems and they also explain that there are two types of network synchronization: synchronous and asynchronous (Lindsey et al., 1985). Following the synchronous and asynchronous topic we found that it is likewise exposed in the article "Complexity of network synchronization" we were able to difference the variety of techniques used in network synchronization. In particular, the article centre its attention in a technique which simulates a synchronous

network by using an asynchronous network. It provides a simpler way to design synchronous network, taking the advantages of the efficiency seen in the asynchronous one (Awerbuch, 1985).

Related to consensus and cooperation among the agents, information it is included in the article "Consensus and Cooperation in Network Multi-Agent Systems" that provides a theoretical framework for analysis of consensus algorithms for multi-agent networked systems with an emphasis on the role of directed information flow, robustness to changes in network topology due to link/node failures, time-delays, and performance guarantees (Olfati-Saber *et al.,* 2007). Useful information has also been found in "Synchronization of Complex Networks: A Unified Viewpoint" that gives us a vision about consensus problem of multiagent systems with a time-invariant communication topology consisting of general linear node dynamics (Feng *et al.*, 2018).

Following with the same topic, we have found an article which adresses consensus in terms of blockchain (Castillo Prieto *et al.,* 2017). Communication between nodes in a blockchain works similar as agents communication. It means that nodes in a blockchain are distributed in order to create a network. A blockchain is governed by a miner, which reaches a consensus in terms of choosing the destination node. In comparison, our miner is the algorithm which decides the route a package follows until it reaches the destination router. Finally, the article "Optimal Design for Synchronization of Cooperative Systems: State Feedback, Observer and Output Feedback" studies synchronization of identical linear systems on a digraph using a spanning tree. A node takes place as a leader and generates the desired tracking trajectory (Zhang *et al.,* 2011).

With all te information collected, inlcuding the one in the article we chose to base our experiment, we have managed to develop an architecture proposal that allows us to study the main point of the chosen article.

3. Proposed Architecture

In this article we are trying to demonstrate how network synchronization and consensus might be together, in order to do it, we are simulating a situation where agents have to send a package of information to a specific router according to its capacity.

So as to explain our proposal, this is the architecture we propose:

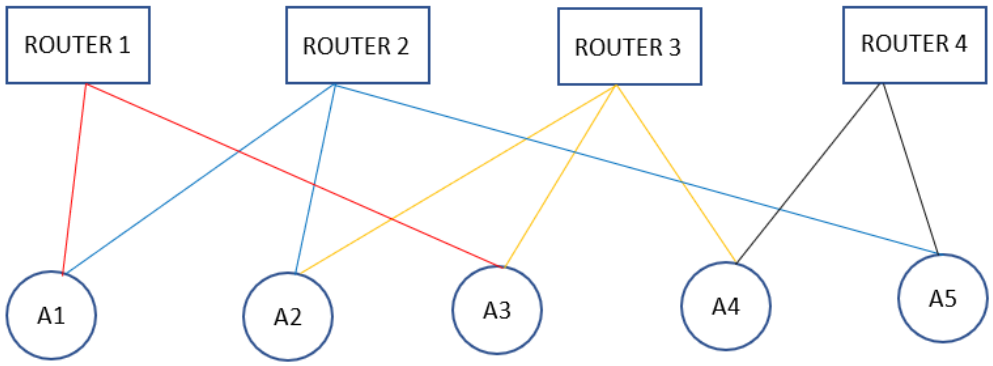

Figure 1. Relation between agents and routers

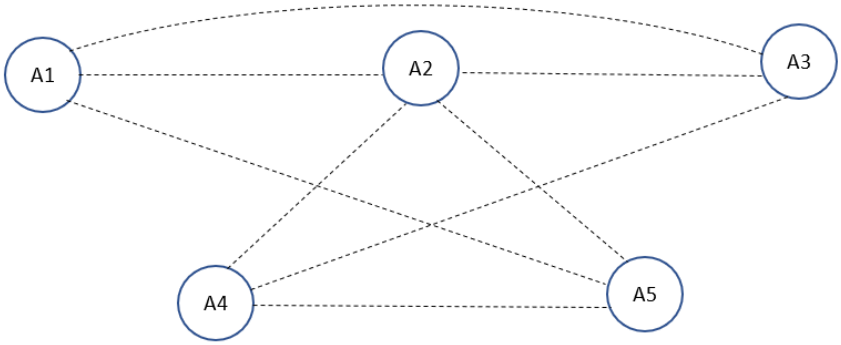

Figure 2. Connections between agents

Figure 1 shows the imaginary representation we used for our architecture, it shows the different connections between the agents. The rectangles of the top represent the different routers and the circles on the bottom represent the agents. Figure 2 shows the direct connection between agents if the routers of Figure 1 didn't exist. For instance, A1 and A2 are connected by Router 2 so that means they have a connection and that is why they have a line joining them.

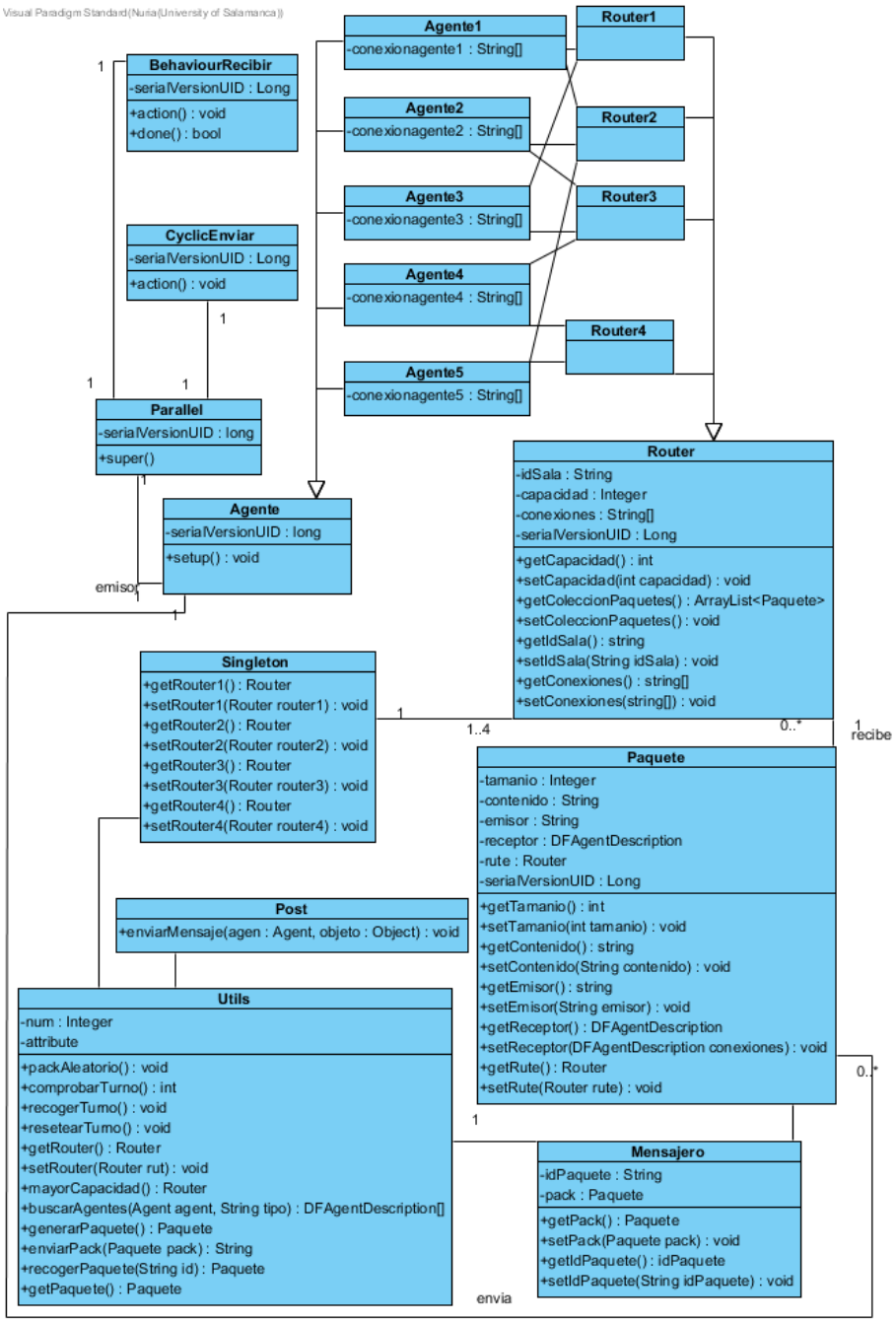

Figure 3(above). Class diagram

Figure 3 shows our class diagram, with the different classes we have used and each class has its attributes and the methods it uses. Agents have an id so that we can identify the agents and an array which allows them to know the connections they have. The class Router has an idSala(roomId) which identifies the different routers, capacidad(capacity) which is the actual capacity of the router so that we can know if a package fits or does not fit and conexiones(conections) which are the conections that a router has with the different agents. The

class Package has tamanio(size), contenido (content), emisor (transmitter) which is the router that creates the package, receptor (receiver) which is the id of the agent that we are going to send the package to (if necessary).

Utils has the functions we use to develop our programme:

- *packAleatorio()* : this function sees (randomly) which agents generates the random package
- *int comprobarTurno()* : it returns the id of the agent that has to generate the package
- *recogerTurno()* : if the agent is the one who has to generate the package, it assignes the value 0 to a variable (turn) so that the rest of the agents can't get there and so that no other package will be generated.

  The turns are organized like this:

  -1 The previous package has been delivered and it has to generate another one.

  0 The package is being sent.

  Any other number from 1 to 5 means that someone has to generate the package.
- *resetearTurno()* : it leaves the num variable at -1 so that it indicates that the package has been sent and so it can generate another one.
- *Router mayorCapacidad()* : it returns the router with the maximum capacity
- *DFAgentDescription buscarAgentes(Agente agente,String tipo) :* this function returns the agents that has the service you are passing as an argument.
- *Paquete generarPaquete()* : this function returns a package that it generates randomly
- *String enviarPack(Paquete pack)* : this function gives the package to the messenger and it returns the id which is what agents send to each other.
- *Paquete recogerPaquete(String id)* : it checks the id of the agent and if it matches with the id of the messenger this function returns the package.

In this class diagram we can also see the connections between agents and routers (see also Figures 1 and 2).

An agent sends an id which will be kept in the messenger which is an object located in the class Utils. When a package is sent, the id of the agent is kept in that object. An agent receives an id and goes to the Utils class to find the id from the messenger, if the id matches then it is the package it has to get.

In our case, the agents used are hibrids but mostly reactives. They are reactives since they realize which router is the best one to store the package, that is to say, they perceive their environment; they also realize which connections they can use. On the other hand, they are proactives since they take the decision of sendind the package to another agent or to the router.

The performative model is (see Figure 4 below): when an agent is not connected to the destination router directly, it sends the package to the next agent for which it uses an algorithim to know which one is the next, if the agent is connected to the router it does not send the package to another agent, it just updates the capacity of the router.

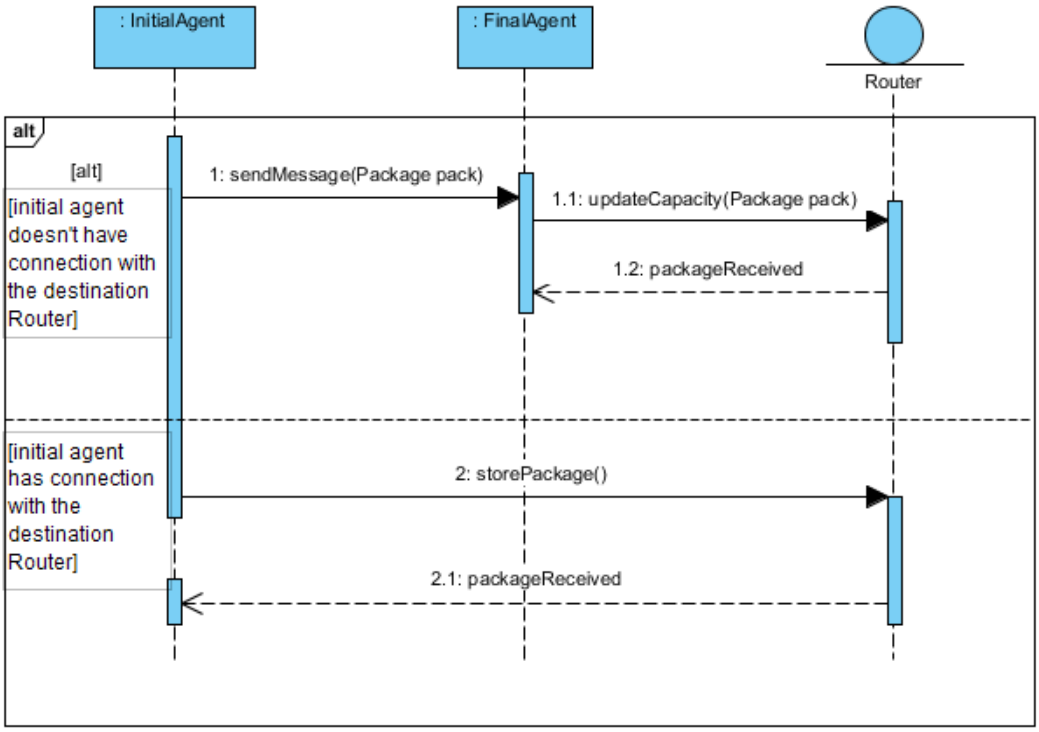

Figure 4. Sequence diagram

Figure 4 shows the sequence diagram of our problem where it is explained the basic idea of it. Initial agent sends the message with the package of data to the destination agent and this final agent updates de capacity of the router. If the initial agent is the one connected to the destination router, it will update the capacity itself without having to send the package to other agent.

4. Experimental Results

We have tried to simulate a network system using four routers and five agents (see Figure 1 and Figure 2). Imagine that we are on the Science faculty and we must connect to a router which might be near us or a little bit far, depending on where classroom are we on.

In orther to achieve our goal, the agents will have to communicate to see which router is better for us depending on whether the router has adequate space to store the package or not.The agent is going to try to connect to the router with the biggest capacity so as to not overflow the remaining routers. In order to do so, the agent will have to communicate with the other agents so that they can send the information to the best option (router).

Explaining it bit by bit:

- **Routers behaviour:**

The routers will be some sort of data container with a maxim capacity. Every router will communicate with at least two agents. When a package arrives to the router, its capacity will change, and the package will be stored in the router. The different routers will also be a connection between some agents.

- **Agents basic behaviour:**

Agents will get packages of data with a specific size (until they deliver the package in the destination router, they will not receive another one) and they will have to communicate with each other using their services, whether they have direct connection to an agent or by passing through the router/s they have in common, so that they can send these packages to the destination router in the most efficient way. That is to say, if the destination router is A1(Computing classroom 1) and it has capacity enough to store the package, the agents will communicate in order to deliver the package there.

If the destination router is full or it does not have enough free space it means that the package will not fit there, and the programme will terminate because it will mean that the package does not fit in any of the routers.

- **Agent-agent connection:**

Connection between agents will be two-way, in other words, if agent 1 is connected to agent 2, agent 1 will be able to send data to agent 2 and vice versa. An agent will be connected to another agent if they have a direct connection or if they have got a router in common. If an agent A does not have either a direct connection or a router in common with an agent B, it means that agent A will have to find an agent or agents which is/are related to both.

- **Agent-router connection:**

It is a simple connection; every single agent will be allowed to control the state and send data to a limited number of routers (in this case 2) in order to secure the integrity of the system. The agent will send the data packages to the destination router provided that the router has enough space and the agent has a possible route to get to him.

- **Notes:**

Because of the fact that we pretend it to be a scalable system, all the agents will have the same programming code. They will only differ in the connection ids with the routers and their agents. With this we make sure that routers and agents can be added in an easy and fast way. In order to prove the integrity of the system, control will be held over the flow and the number of data packages the agents receive to measure system's stress. If it is necessary, the behaviour of the package delivery will be changed so as to achieve the optimum solution.

5. Discussion

As we explained before, in our experiment we simulate a system of different routers situated in different classrooms and how agents must work with each other in order to achieve their aim, which in this case is delivering a package in a destination router. It works as it follows, an agent chosen randomly by an algorithm is the one who starts the communication, this agent generates a package, also randomly. Using an algorithm that choses the best router to store the package, the agent verifies if it is connected to that router; if it is connected to it, it will store the package in the router and will also update the router's capacity.

If it is not connected to the router, it will send the package to the next agent who will do the verification itself. This procedure is repeated until the routers are full or until a package will not fit in any of the routers available. We decided to do this experiment because we wanted to show how network synchronization might work and how agents must communicate to achieve an aim (consensus). This was the intention of the article we chose, to show these issues of multi-agent systems together but, as we previously said, they have done it by analysing it with maths while we have done it in a practical way with the architecture we have proposed. We also wanted to demonstrate how multi-agent systems are helpful in situations related to our daily life, in this case, connecting to the internet.

An example of the execution of our programme:

```
generando pack...
paquete generado
Mayor capacidadAI4
Tengo conexion directa con el AI4
Subiendo al router AI4 capacidad:635
```

(...)

Figure 5. Execution of the program

```
generando pack...
paquete generado
Mayor capacidadAI2
No tengo conexion directa con el router, procediendo a enviar mensaje
soy el agente Agente3enviando mensaje a Agente2
mensaje enviado
Comprobando mensaje, soy el Agente2...
Subiendo al router AI2 capacidad:98
```

(...)

Figure 6. Execution of the program

```
paquete generado
Mayor capacidadAI4
Tengo conexion directa con el AI4
Subiendo al router AI4 capacidad:17
generando pack...
paquete generado
No hay suficiente capacidad, lo sentimos...
```

¡Figure 7. Execution of the program

(…) Means that there are more similar results and here we wanted to show an example of each case possible. Translation of figures 5, 6 and 7:

Figure 5
Generating package…
Package generated
Biggest capacity AI4
I have direct connection with AI4
Uploading to the router AI4 capacity: 635

Figure 6
Generating package
Package generated
Biggest capacity AI2
I don't have direct connection with the router, proceeding to send the message
I am Agent 3 sending message to Agent 2
Message sent
Validating message, I am Agent 2
Uploading to router AI2 capacity:98

Figure 7

Package generated

Biggest capacity AI4

I have direct connection with router AI4 capacity:17

Generating package

Package generated

There's not enough capacity, we are sorry...

In this paper, network synchronization and consensus have been studied by creating an example of their application. An architecture with agents and routers has been proposed in order to explain the two problems. Network synchronization is the issue itself, routers connections make a network, and consensus is stablished by the agents while they want to send a package. It has been shown how useful multi-agent systems are in our daily-life. Many examples of its use have been given, not only in our proposed architecture (connection to internet) but also in the examples we found in our research (GPS, films…).

6. Conclusion

Network synchronization and consensus have been explained and a few examples of each issue have been presented. These two topics have been studied in our article by creating an example of their application in a real-life problem. This paper is an attempt to provide another point of view of the proposal by (Feng *et al.*, 2018).

An architecture with agents and routers has been proposed in order to explain the two problems. Network synchronization is the issue itself, routers connections make a network, and consensus is stablished by the agents when they need to send and receive messages in other to take the package to the destination router. Different classes (routers, packages…) have been used in our programme to recreate this problem using a jade application. The result of our programme shows how agents communicate with each other as a team to solve a problem that would have been very difficult for an agent on its own.

7. References

Awerbuch B (1985) Complexity of network synchronization.

Castillo Prieto F, Kush S, Corchado Rodríguez J.M (2017) Distributed Sequential Consensus in Networks. Analysis of Partially-Connected Blockchains with Uncertainty. Complexity.Volume, pp.1-11. Hindawi

Feng Y, Duan Z, Ren W, Chen G (2018) Consensus of multi-agent systems with fixed inner connections. Int J Robust Nonlinear Control;28:154–173.

González-Briones A, De la Prieta F, Mohamad M, Omatu S, Corchado J.M (2015) Multi-Agent Systems Applications in Energy Optimization Problems: A State-of-the-Art Review.

González-Briones A, Villarrubia G, De Paz J.F, Corchado J.M (2018) A multi-agent system for the classification of gender and age from images.

Harding G (2015) Multi-agent system for global positioning system (GPS) web services.

Khatua S, Yuheng Z, Das A, Iyengar. N.Ch.S.N (2011) A multi agent based e-shopping system.

Khoo S, Xie L, Man Z (2009) Integral terminal sliding mode cooperative control of multi-robot networks..

Lee W-P, Liu C-H, Lu C-C (2002) Intelligent agent-based systems for personalized recommendations in Internet commerce.

Lindsey W.C, Ghazvinian F, Hagmann W.C, Dessouky K (1985) Network synchronization.

Moreno A (2003) Medical Applications of Multi-Agent Systems.

Mossel E, Schoenebeck G (2009) Reaching Consensus on Social Networks.

Okresa Duric B,Tomicic I, Schatten M (2015) Multi-agent modeling methods for massivley multi-player online role-playing games.
Olfati-Saber R, Fax J.Alex, Murray R.M (2007) Consensus and Cooperation in Networked Multi-Agent Systems.
Olfati-Saber R, Murray R.M (2004) Consensus problems in networks of agent with switching topology and time-delay.
Penn Y, Segal M, Moses E (2016) Network synchronization in hippocampal neurons.
Ren W, Beard R.W (2005) Consensus seeking in multiagent systems under dynamically changing interaction topologies.
Sáez Encinar (2015) A.I. La inteligencia artificial en el cine.
Zhang H, Lewis Frank L., Das A. Optimal Design for Synchronization of Cooperative Systems: State Feedback, Observer and Output Feedback.