Implementing MAS Agreement Processes Based on Consensus Networks

A. Palomares, C. Carrascosa, M. Rebollo, and Y. Gómez

Universitat Politècnica de València,
Camino de Vera s/n 46022 Valencia (Spain)
{apalomares, carrasco, mrebollo, ygomez}@dsic.upv.es

Abstract. Consensus is a negotiation process where agents need to agree upon certain quantities of interest. The theoretical framework for solving consensus problems in dynamic networks of agents was formally introduced by Olfati-Saber and Murray, and is based on algebraic graph theory, matrix theory and control theory. Consensus problems are usually simulated using frameworks as MATLAB or Mathematica. However simulation using multi-agent system platforms is a very difficult task due to problems such as synchronization, distributed finalization, and monitorization among others. The aim of this paper is to propose a protocol for the consensus agreement process in MAS, implement it in the MAGENTIX2 platform, and integrate the results with a MATLAB in order to get a previous simulation of the system, check the correctness of the algorithm and validate the protocol.

Keywords: Self-Organization, Agreement, Consensus, Networks

1 Introduction

In agent-based networks, 'consensus' is referred to reach an agreement about a certain quantity of interest or distribution function that depends on the state of all agents [7]. Consensus algorithms can be modeled as iterative processes in which autonomous agents work in a distributed fashion, without the necessity of sending information to a central node that acts as coordinator.

When it is implemented in a real system, some considerations about how the communication process will take place must be taken into account. The set of rules that specifies the information exchange between an agent and all of its neighbors on the network are specified in a consensus protocol.

Distributed Constraint Optimization Problems (DCOP) [6, 11] are a model for representing multiagent systems (MAS) in which agents cooperate to optimize a global objective, which could be used to deal with this type of problem too. A DCOP is a formalism that captures the rewards and costs of local interactions in a MAS where each agent chooses a set of individual actions. In a DCOP each agent receives knowledge about all relations that involve its variable(s). There are two main types of complete algorithms to solve DCOPs: search and dynamic programming. Search algorithms, like ADOPT (or extensions such as IDB-ADOPT), require an exponential number of linear-size messages. Dynamic programming algorithms, like the distributed pseudo-tree optimization

procedure (DPOP) and its extensions, only require a linear number of messages, but their complexity lies on the message size, which may be very large.

Nonetheless, as [9] details, there are some limitations in the way DCOP algorithms approaches the problem. Firstly, they assume that the environment is deterministic and fully-observable, meaning that agents have complete information about the utility of the outcomes of their possible decisions. Thus, agents' utilities do not change while the problem is being solved. Moreover, agents' actions are only applied once the problem is solved. Furthermore, the set of agents in the system is constant, not allowing the entrance or exit of the system.

This paper propose a protocol for the consensus agreement process in MAS. This protocol has been implemented it in the MAGENTIX2 MAS platform, addressing the problems that a real implementation introduces in a theoretical mathematical model. To check its correctness, the results of the MATLAB simulations are compared with the results obtained by the real execution of the MAS in MAGENTIX2.

2 Consensus Networks

The theoretical framework for solving consensus problems in dynamic networks of agents was formally introduced by Olfati-Saber and Murray [7]. The interaction topology of the agents is represented by a graph and *consensus* means to reach an agreement regarding a certain quantity of interest that depends on the state of all agents in the network. This value represents the variable of interest in our problem *(agreement term)*, and might be for example a physical quantity, a control parameter or a price among others.

Let G be a graph of order n with the set of entities E as nodes and weighted adjacency matrix $A = [a_{ij}]$. Let (G, X) be the state of a network with value X and topology G, where $X = (x_1, \ldots, x_n)^T \in \mathbb{R}^n$, where x_i is a real value associated with the node E_i. The value of a node might represent physical quantities measured in a distributed network of sensors (temperatures, voltages, velocities, prices, qualities,...). A network is a dynamic system if (G, X) evolves in time. A consensus algorithm is an interaction rule that specifies the information exchange between the agents and all of its neighbors on the network in order to reach the agreement. Consensus of complete network is reached if and only if $x_i = x_j \ \forall i, j$. Distributed solutions of consensus problems in which no node is connected to all nodes are especially interesting. The most commonly used consensus protocols are Average, Maximum and Minimum because the have broad applications in distributed decision-making multi-agent systems.

These authors have demonstrated that a convergent and distributed consensus algorithm in discrete–time can be written as follows:

$$x_i(k+1) = x_i(k) + \varepsilon \sum_{j \in N_i} a_{ij}(x_j(k) - x_i(k)), \tag{1}$$

where N_i denotes the set formed by all nodes connected to the node i (neighbors of i). The collective dynamics of the network for this algorithm can be written as

$$x(k+1) = P\, x(k) \tag{2}$$

Algorithm 1 Consensus algorithm. Power iteration method for the matricial form

1: $D = \sum_{i \neq j} a_{ij}$
2: $L = D - A$
3: assign a $\varepsilon < 1/\Delta$
4: $P = I - \varepsilon L$
5: init x with random values
6: **repeat**
7: $\quad x = P * x$
8: **until** the system converges or maxiter reached

where P is the Perron matrix of a graph with parameter ε, defined as

$$P = I - \varepsilon\, L \tag{3}$$

with I is the identity matrix, $\varepsilon > 0$ is the step size, and L is the Laplacian matrix, $L = D - A$, that is the difference of the degree matrix D and the adjacency matrix A of the graph. The algorithm converges to the average of the initial values of the state of each agent and allows computing the average for very large networks via local communication with their neighbors on a graph. Algorithm 1 is the specification of the consensus process in matricial form.

Solving consensus is a cooperative task because if a single agent decides not to cooperate and keep its state unchanged, all others asymptotically agree with them. However, if there are multiple non cooperative agents then no consensus can be asymptotically reached. When an agent does not cooperate with the rest its links in the network might be disconnected in order all others will asymptotically agree. In this case it is impossible for all agents to reach an agreement but is possible to reach an agreement for the rest of the agents.

3 Implementation of Consensus Protocol for MAS

Usually, consensus algorithms are checked by means of simulation executions, where the multi-agent system is modeled as a matrix. Using a power iteration method, the matrix is applied over the solution vector until the changes between successive iterations are under a threshold. This kind of solutions are easier to implement and to test than real executions, because they avoid to deal with problems such as:

- Synchronization: how the agents coordinate to deal with the consensus process. To get this, a new consensus protocol is needed.
- Distributed finalization: how any agent detects when the consensus is reached, or when the process has to end.
- Monitorization: how can be observed the dynamic of the consensus process.
- Bottlenecks: are there any bottleneck in the process? and if so, how it is dealt with.
- Communication: how to deal with the overload of messages, and the ordering of the different messages from the different agents any agent is connected to?

– Scalability: is it possible to scale the MAS trying to reach the consensus? how it affects to the above mentioned problems?
– Lack of MAS design toolkits with connection networks (point-to-point agent communication), allowing the automatization in the developing of different scenarios.

Nevertheless, the use of a matricial system assumes a complete knowledge scenario in which all agents know the entire network structure and the values. This approach can be useful in centralized and relatively small systems, but if the size of the network grows or participants play in conditions of bounded rationality, a centralized approach is not feasible.

MAGENTIX2[1] agent platform [10] has been used to implement real agents that follow the consensus algorithm to reach agreements. In a platform with real agents, the formerly mentioned problems have to be addressed and properly solved.

The next subsections details, first of all, the simulation calculus that was realized using MATLAB[2] to test in a first installment the validity of the proposal, and after that, the real implementation in an agent platform (MAGENTIX2), along with the developed mechanism that allows from a GraphML[3] file indicating the connection network, generate automatically the MAS following such connections and execute it to reach a consensus (see Figure 1).

3.1 MATLAB Implementation

Algorithm 1 describes the implementation of equation 2, which simulate the behavior of a network following the matricial form. Languages that include numerical libraries can be used to program the model and simulate the results. Some possibilities are MATLAB, Mathematica, Fortran, Python (with Numpy library) among others. In our case, we have decided to use MATLAB. Algorithm 2 contains the code that corresponds to the MATLAB implementation of the consensus process.

At the beginning, adjacency matrix A is imported from a GraphML file. After that, the Perron matrix P is calculated. Finally, the code enters in a loop in which the values x_i of the variables for each agent are updated until the maximum number of iterations `maxiter` is reached.

The main limitation of this method is the memory needed to store the adjacency matrix and the Perron matrix. But this can be solved using sparse matrices and adapting the corresponding functions to deal with them. With this change, a model can be easily created with a maximum of 10^7 agents, which is the bound for the 32-bit version of MATLAB.

Nonetheless, the numerical method is a valid tool to (i) simulate the behavior of the system, (ii) check the correctness of the consensus algorithm in different situations other than the original Olfati-Saber and Murray proposal, and (iii) validate the implementation.

[1] http://magentix2.gti-ia.upv.es
[2] http://www.mathworks.com/products/matlab/
[3] http://graphml.graphdrawing.org

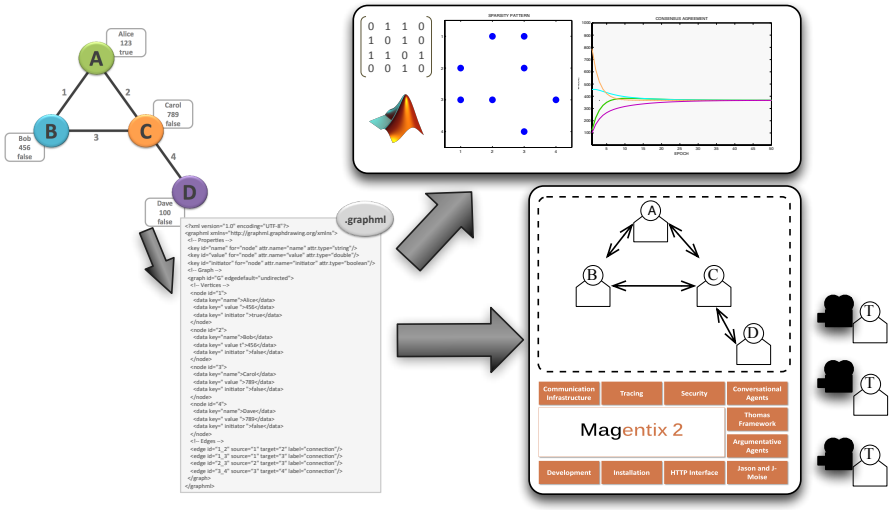

Fig. 1: Implementation process. The network is described in a GraphML file that feeds the MATLAB simulation and the initial configuration of the real MAS. A set of tracer agents can monitor the evolution of the process

3.2 MAGENTIX2 Implementation

MAGENTIX2 is an agent platform that follows the levels proposed in [5], providing support for organization (according to the THOMAS framework specification [1]), interaction (by means of FIPA-ACL messages) and agent levels, through a java language API. MAGENTIX2 provides developers with a predefined set of FIPA-based interactions protocols. Consensus protocol can not be created by combination of these basic protocols, so a new protocol has been implemented. MAGENTIX2 allows the definition of protocols based on a finite-state machine that, once they has been specified, the agent can automatically run them using the so called *conversation factories* [4].

Algorithm 2 MATLAB Code

```
n = size(A,1)
L = spdiag(sum(A)) - A
eps = 1 / max(sum(A))
P = speyes(n) - eps * L
x = rand(n,1)
for i = 1:maxiter
  x = P * x
end
```

The logic associated to the consensus protocol is the one specified in equation 1. We assume the agents are homogeneous and benevolent, so all of them implement the same process and follow strictly the protocol specified.

Moreover, all the generated agents that participate in the consensus process make use of the MAGENTIX2 tracing services [2] to generate events with all the needed information about the consensus process. So, any number of monitoring agents can be generated to connect to the events generated by such agents. These monitoring agents have their own Ubigraph[4] server, a free software for visualizing dynamic networks. The server is provided with all the information from the events generated by the agents involved in the consensus process. In this way, it is easy to follow the consensus process and check if and how the MAS reach a consensus through the visualization of the graph using a color codification to describe the current value of each agent and different shapes of the nodes to describe the state of each agent in the consensus protocol.

UbiGraph tool has been used to visualize the evolution of the agents during the consensus agreement proccess in MAGENTIX2. The state of each agent in the network at discrete time k, $x(k)$, is represented by the color of the node property in UbiGraph, that is calculated with a previously defined color-scale (see Figure 5). Initially, colors of the agents are very different and tends to be more similar when consensus evolves until the convergence is reached and all agents process have the same color.

It has been generated a system that has as input a GraphML file that describes the connection network of a MAS, that is, nodes represent agents, and edges represent the connections between them (who knows who, who can communicate with who). From this file, it generates a real MAS composed of MAGENTIX2 agents.

Consensus Protocol A general consensus protocol has been designed in order to allow a set of agents connected in a network to achieve an agreement exchanging information exclusively with their direct neighbors. Agents do not have information about the size of the network (how many agents are participating) nor how they are connected. The protocol is characterized for the absence of any kind of controller agent and all decisions are taken autonomously and in a decentralized way.

When an agent decides to start a consensus process, it sends a consensus proposal to all its neighbors (Figure 2). Each one of these neighbors decide to participate or not. If it participates, confirms its participation to the sender and re–sends the proposal to its corresponding neighbors. The process continues until all agents in the network have received the proposal. Therefore, all agents play two roles: initiator and participant in the consensus.

Once agents have received the confirmation of the neighbors desiring to participate in the consensus, the next step begins. Agents exchange their values with the neighbors and, after each iteration, update their internal values following the equation 1. In this step, all agents act as *participant* and they just send its value and recalculate it from the values received from their neighbors until a maximum number of iterations are executed or the difference with the value of the previous iteration $|x_i(k) - x_i(k-1)|$ is below a threshold (see termination problems for a more detailed analysis of this question).

[4] http://ubietylab.net/ubigraph/

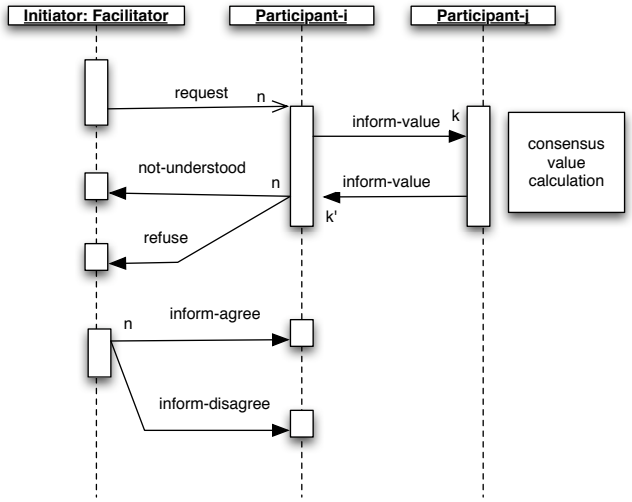

Fig. 2: General Consensus Protocol. An Initiator decides to start a consensus process and sends a proposal to its neighbors, that is propagated to the complete network. After that, agents exchange their values until the consensus is reached.

Agent implementation The protocol has been adapted to match the requirements of MAGENTIX2 platform. A *conversation factory* has been created that follows the finite-state machine shown in Figure 3. States are divided into three types: *send* states (in which the result is a message sent to the system), *receive* states (in which the agent is automatically awake to treat a concrete message) and general states.

The protocol has two steps. In the first one, agents that are interested in the consensus process are identified (an agent can individually decide if it participates or not). After that, the actual consensus step begins and all involved agents exchange their values and updates them until some finish condition is met.

The main problem that has to be addressed by the protocol is the synchronization of the agents. The MATLAB model assumes that all agents update their value at the same time and all of them have available the current value of their neighbors. But in the case of the real system, with delays or even failures in the network, this can not be ensured. And if the agents have not properly updated information, the final average value obtained by the network can differ from the theoretical average value.

When the platform starts, all agents instantiate a factory for the consensus protocol that stay in a wait state until some agent decides to launch a consensus process. Eventually, some agent will decide to begin a consensus. It sends a `propose` message to all its neighbors and it waits until all its neighbors have answered. Each agent decides if it wants to participate, in which case an `agree` answer is sent, or not, replying with a `refuse` message. Any agent accepting the consensus, for its part, sends a propose to its neighbors until the complete network acknowledges the propose. In order to let the second part of the consensus start, each agent must receive an answer from its neighbors.

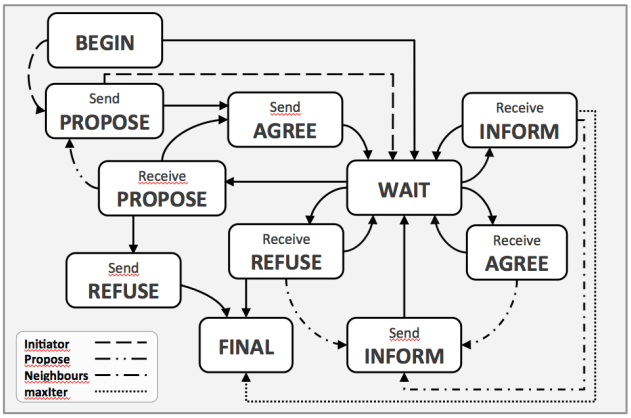

Fig. 3: State Automaton of the Consensus Communication Protocol in MAGENTIX2.

In a second step, all participating agents inform their neighbors with their current values. The agent waits until all agents that have answered with their corresponding values. When all values have been received, each agent calculates their new updated value using equation 1 and informs its neighbor about the new value. This behavior makes the agent to be **synchronized** and evolve step by step at the same time.

As stated above, the messages may have a delay. Furthermore, the agents are free to inform when their direct neighbors have answered. These facts could cause that each agent receive messages belonging to different iterations. In order to solve this problem, a buffer has been created, in which the information of different iterations are saved, ensuring the agents to have a properly updated information.

The **distributed finalization** problem refers to how an individual agent can detect that the calculated value corresponds to the true consensus value or it is the process which has an slow dynamics. The considered possibilities are

1. to introduce a timeout
2. to specify a maximum number of iterations
3. to establish a threshold for two consecutive iterations
4. to include other dynamic conditions (for example, if one agent leaves the consensus)

In this work, static networks are considered only, so the last option is not applicable. The limit by timeout or number of iterations can be easily implemented by exchanging the corresponding value as a consensus parameter in the initial `propose` message. Each agent will stop itself when the specified condition is met. The third case is more complicated and requires the modification of the finite-state-machine to include additional states to handle a new set of messages. Basically, any agent can propose to stop the process because it has reached a possible consensus value and this message is prop-

agated throughout the entire network (as has been done with the propose message). The consensus will stop when all the agents answer affirmatively to this question.[5]

MAGENTIX2 platform has a trace system available for developers to supervise the evolution of the system. It is implemented based on an event subscription scheme. All the agents can generate events to make notice diverse changes: internal state, activation, deactivation, message incoming, message sending, etc. Any other agent can subscribe to a set of events, so it will eventually receive any change in the system related with the event it is interested in. Using this mechanism, **monitoring** agents can be created and subscribed to changes in the consensus value of all the agents. When an agent changes its internal value, it triggers an event. The monitoring agent catches and treats it. An event is lighter than a message, so the overload introduced in the system is tractable.

As the proposed implementation is a completely decentralized system, there is no functional **bottlenecks**. Nonetheless, regarding with the network topology, it is possible for a a reduce number of agents to concentrate the responsibility of the consensus process. This fact is related with the convergence speed of the network and some spectral properties of the adjacency matrix whose analysis is out of the scope of this paper.

Regarding with the cost of the **communications**, the number of exchanged messages needed to complete the consensus process depends on the connectivity of the network. In each iteration, each agent sends exactly one message to each one of its neighbors. If we assume the network to be undirected, each link is used twice. In the case of a complete network, the number of messages in each step is $n(n-1)$ (where n is the number of agents in the network), that belongs to $O(n^2)$. Nevertheless, communication networks tend to be very sparse (with a low density: only a small fraction of nodes are connected). In that case, the cost of the messages can be considered as $\Theta(n\bar{d})$, where $\bar{d}$ is the average degree of the network.

Finally, **scalability** of the system is provided by the MAGENTIX2 platform. It is supported by AMQP standard for the communication, which is widely used in the industry as messaging middleware (see [2, 4, 10] for a more detailed explanation of the internals of the platform).

4 Practical Usage

We have developed for this paper a case of study where we have 100 nodes (though in the end, there is only 96 because 4 of them are not connected to the rest). This nodes are structured in a random network with a average degree $d = 3$. Figure 4 (top–left) shows the topology of the generated network and the consensus process as the result of the MATLAB simulation Figure 4 (bottom). The agents has been reordered, but this step is not mandatory and the same results can be obtained without the reordering (Figure 4 (top–right)).

It can be seen how the systems converges to the average value quickly. After 20 epochs, all agents have approached sufficiently to the consensus value and, after 100 epochs, almost all the agents have arrived to the definitive value. The speed of the

[5] The variation has not been explained because it does not introduce any change in the consensus process but complicates the comprehension of the protocol. The complete version has been implemented and it is available to download.

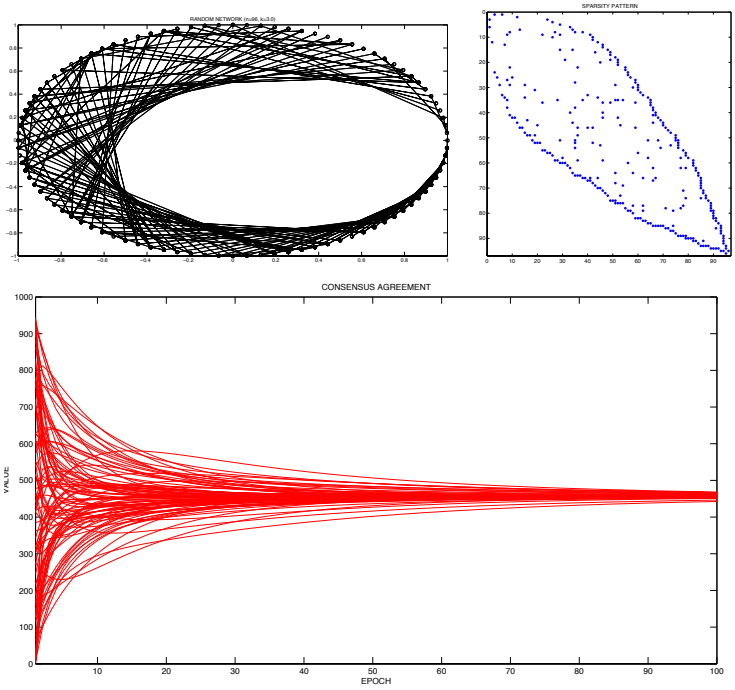

Fig. 4: Network structure (top) and evolution simulated in MATLAB (bottom)

convergence does not depend on the size of the network, but on its connectivity, as can be easily inferred from the theoretical model. For example, in the case of a complete network (all nodes are connected with the rest of the network), the processes finishes just in one step, as all the agents receives all the values and calculates directly the true average value. Nevertheless, in other topologies, such as a ring, the convergence is slower because each agent only is connected with its left and right neighbors and any change in the values has to be propagated node by node until it reaches the opposite side of the ring. The network topology is another determinant too. There exist several well known techniques to analyze it in advance, but it is out of the scope of this paper. We refer the reader to [8] for a broader explanation of this topic.

Figure 5 shows the real execution of the same consensus process in the MAGENTIX2 implementation. Some snapshots of the animation generated in real–time by the monitoring agents have been taken in different time instants. Monitoring agents are independent entities from the consensus process which are subscribed to the events generated by the agents participating in the consensus. The color of each node reflects the current consensus value, which are mapped in the spectra that appears on the top. It can be seen how, beginning from a random initial situation, (epoch 1), the agents gradually get close each other and, after 100 epochs, the values are almost the same.

This section has shown the possibilities of the developed system to simulate and execute in a real MAS the same connection network to reach an agreement. Nevertheless, intensive tests have been run in the simulated process and in the MAGENTIX2 platform.

5 Conclusions

This paper shows the application of consensus networks in a distributed and self-organized fashion, implemented in an agent platform with real agents.

The contribution of the paper is the proposal of a protocol that allows a network of agent achieve agreements using the consensus algorithm. This protocol has been implemented in the MAGENTIX2 MAS platform and tested, comparing the obtained results with the MATLAB simulation data. Both implementations arrive to the same numerical result, validating the correctness of the proposed protocol.

As future work, the integration with a development tool is planned, in order to create virtual organizations that follow a concrete network topology. After that, is possible to automatically generate templates for the agents forming a network that follows the links specified at design time. Different network topologies can be easily generated just providing some basic configuration parameters, such as the topology itself (random, preferential attachment, small-world, growing), degree distribution, clustering, or assortativity index. Furthermore, more complicated scenarios can be tested, as the effect of time delays, changes in the dynamics when agents are allowed to enter or leave the system, multi–variable consensus and consensus in coupled systems, generated when several consensus processes take place at the same time in different groups with some common participants.

References

1. Argente, E., Botti, V., Carrascosa, C., Giret, A., Julian, V., Rebollo, M.: An Abstract Architecture for Virtual Organizations: The THOMAS approach
2. Búrdalo, L., García-Fornes, A., Julian, V., Terrasa, A.: TRAMMAS: A tracing model for multiagent systems
3. Carrascosa, C., Rebollo, M.: Agreement spaces for counselor agents (short paper). In: Proc. of 8th Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS 2009). Decker, Sichman, Sierra and Castelfranchi (eds.), May, 10-15, 2009, Budapest, Hungary (2009)
4. Fogués, R.L., Alberola, J.M., Such, J.M., Espinosa, A., García-Fornes, A.: Towards Dynamic Agent Interaction Support in Open Multiagent Systems
5. Luck, M., McBurney, P., Shehory, O., Willmott, S.: Agent technology: Computing as interaction (a roadmap for agent based computing)
6. Mailler, R., Lesser, V.: Solving distributed constraint optimization problems using cooperative mediation. In: Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems - Volume 1. pp. 438–445. AAMAS '04, IEEE Computer Society, Washington, DC, USA (2004), http://dx.doi.org/10.1109/AAMAS.2004.249
7. Olfati-Saber, R., Fax, J.A., Murray, R.M.: Consensus and cooperation in networked multiagent systems. Proceedings of the IEEE 95(1), 215–233 (2007)
8. Pereira, S.S., Zamora, A.P.: Mean Square Convergence of Consensus Algorithms in Random WSNs. IEEE Transactions on Signal Processing 58, 2866 – 2874 (2010)

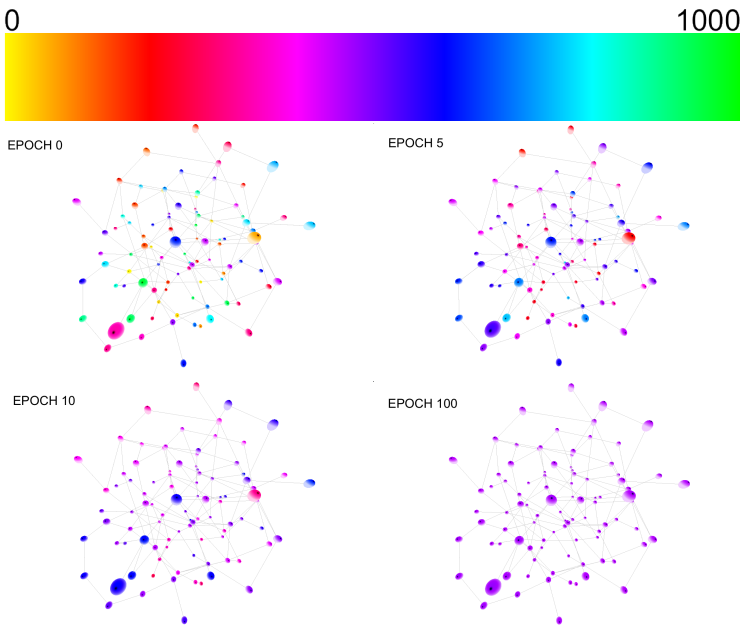

Fig. 5: Evolution of the consensus network in the MAGENTIX2 implementation.

9. Pujol-Gonzalez, M.: Multi-agent coordination: Dcops and beyond. In: Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence. pp. 2838–2839 (2011), http://aaai.org/ocs/index.php/IJCAI/IJCAI11/paper/view/3106
10. Such, J., García-Fornes, A., Espinosa, A., Bellver, J.: Magentix2: A privacy-enhancing agent platform. Eng. Appl. Artif. Intel. 26(1), 96–109 (2013)
11. Vinyals, M., Rodriguez-Aguilar, J., Cerquides, J.: Constructing a unifying theory of dynamic programming dcop algorithms via the generalized distributive law. Autonomous Agents and Multi-Agent Systems 22, 439–464 (2011), http://dx.doi.org/10.1007/s10458-010-9132-7, 10.1007/s10458-010-9132-7