DNVA: a tool for visualizing and analyzing multi-agent learning in networks

Sherief Abdallah, Sima Sadleh
Faculty of Engineering and IT
British University in Dubai
Dubai, United Arab Emirates
Email: shario@ieee.org

Iyad Rahwan, Aamena Al Shamsi
Masdar Institute
Abu Dhabi, United Arab Emirates

Victor Lesser
Computer Science Dept.
University of Massachusetts Amherst
Amherst, United States

***Abstract*—Networks are seen everywhere in our modern life, including the Internet, the Grid, P2P file sharing, and sensor networks. Consequently, researchers in Artificial Intelligence (and Multi-Agent Systems in particular) have been actively seeking methods for optimizing the performance of these networks. A promising yet challenging optimization direction is multi-agent learning: allowing agents to adapt their behavior through interaction with one another. However, understanding the dynamics of an adaptive agent network is complicated due to the large number of system parameters, the concurrency by which the system parameters change, and the delay in the effect/consequence of parameter changes. All these factors make it hard to understand why an adaptive network of agents performed well at some time and poorly at another.**

In this paper we present a software tool that enables researchers in the multi-agent systems field to visualize and analyze the evolution of adaptive networks. The proposed software customizes and implements techniques from data mining and social network analysis research and augment these techniques in order to analyze local agent behaviors. We use our tool to analyze two domains. In both domains we are able to report and explain interesting observations using our tool.

***Keywords*-multi-agent learning; visualization; network analysis; dynamics**

I. Introduction

Networks are seen everywhere in our modern life, including the Internet, the Grid, P2P file sharing, and sensor networks. Consequently, researchers in Artificial Intelligence (and Multi-Agent Systems in particular) have been actively seeking methods for optimizing the performance of these networks. A promising yet challenging optimization direction is multi-agent learning: allowing agents to adapt their behavior through interaction with one another. However, understanding the dynamics of an adaptive agent network is complicated due to the large number of system parameters, the concurrency by which the system parameters change, and the delay in the effect/consequence of parameter changes. All these factors make it hard to understand why an adaptive network of agent performed well at some time and poorly at another.

This material is based in part upon work supported by the British University in Dubai (BUiD) under Grant Number INF009 and the National Science Foundation (NSF) under Award Number IIS-1116078. Any opinions, findings, conclusions or recommendations expressed in this publication are those of the authors and do not necessarily reflect the views of BUiD or NSF.

In this paper we present a tool that enables researchers in the multi-agent systems field to visualize and analyze the evolution of adaptive networks. Such tool facilitates better understanding of multiagent learning dynamics in networks; and is therefore an important step toward developing more advanced and robust learning techniques for agent networks. While there has been some previous work that used data mining techniques to analyze communication messages (between agents) [15], the previous work focused on debugging multi-agent systems and inspecting individual messages exchanged between the agents. Such analysis is not helpful in understanding learning dynamics that require higher level of analysis and looking at agent strategies rather than individual messages. Also some of the previous work provided tools for analyzing multi-agent systems visually, but these tools did not rely on a methodology based on data mining as we do here [12], [8]. Most of the work in analyzing (communication) networks and distributed systems relied on simple heuristics that are intuitive, easy to understand, and experimentally verified to work adequately. Even when large-scale and in-depth analysis of the network behavior was conducted, the underlying nodes were assumed to have fixed behavior (not learning) [13]. Here we are interested in analyzing networks of learning agents, where the dynamics of the network change over time even if the outside world remains unchanged. As Section III illustrates, the software tool helped us analyze and explain interesting observations that were reported in earlier work for two domains: social learning [17], [2] and the taxi dispatch system [4].

To summarize, our contributions in this paper are:

- Presenting a new tool for analyzing the learning dynamics of a network of adaptive agents.
- Two case studies that illustrate the use of our tool in explaining observed phenomena.

II. Dynamic Network Visualization Analysis (DNVA) Tool

Our DNVA tool is an open source software that builds on the Social Network Image animator (SoNIA) open source

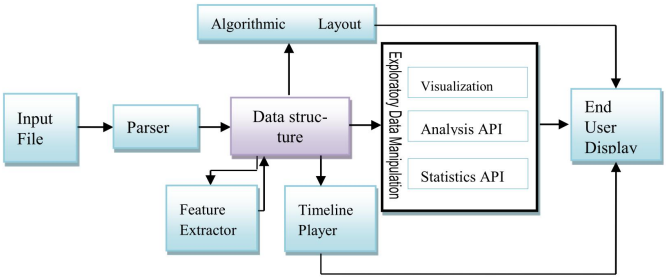


Figure 1. DNVA Tool Block Diagram.

package [6]. SoNIA is a java based package that helps in exploring dynamic network evolving over time and facilitates studying the relations of networks' entities (nodes). SoNIA focused on visualizing dynamic networks over different slices of time (via different algorithmic layouts). Our tool extends SoNIA's capabilities in the following aspects:

- New XML format and a corresponding parser and data structure that is more suitable for large-scale networks and online analysis.
- Advanced detailed network visualization with corresponding data mapper.
- Novel Network Analysis Visualization

Figure 1 shows the block diagram of the DNVA tool. The remainder of this section describes the main elements of DNVA.

A. New Input Format

Several software tools support XML as an input format, with considerable variations. Only few formats, however, support *dynamic* networks. Figure 2 compares the logical structure of the input file for the (original) SONIA software against the format of our tool. The file format for SONIA has two crucial limitation: time scalability and online visualization. For SONIA, changes in the network are grouped per network entity (i.e. per node or edge). So for example, all the changes in node X's features should be grouped together in one block in SONIA's input file format. Such format works for small time scale, because it requires processing the complete input file before being able to visualize anything. This should be contrasted to our file format where changes are grouped per time instance, which allows immediate and online visualization of changes. Further features of the new input file format include:

- Nodes and edges' addition and removal is allowed.
- Modification in nodes and edges attributes over time is allowed.
- Default values provided for nodes with missing edges/nodes attributes.

The structure of the proposed DNVA input file is illustrated below.

```
<DynamicNetwork>
```

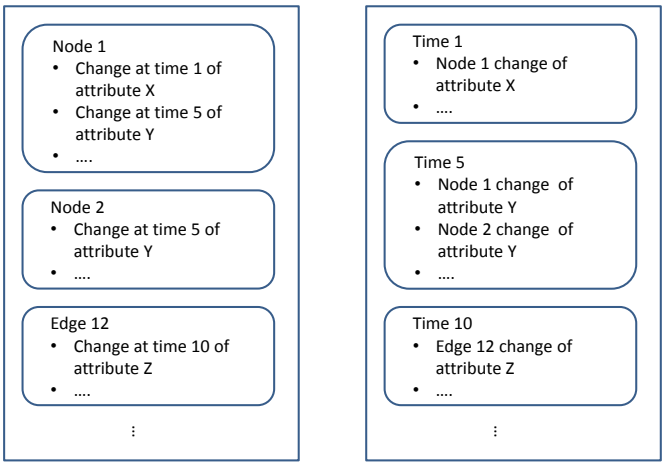


Figure 2. Comparing the file formats of SONIA (left) and DNVA (right).

```
<!-- Nodes Attributes definition -->
<Nodeattributes>
<attribute name='Q_a1' type='Double'/>
<attribute name='Q_a2' type='Double'/>
</Nodeattributes>

<!-- Edges Attributes definition -->
<Edgeattributes>
<attribute name='Num_of_Interactions'
        type='Integer'/>
</Edgeattributes>

<!--Initial nodes, edges and attributes at
        time 0 -->
<change time='0'>
<!-- List of nodes -->
<node id='0'>
<attributeValue name='Q_a1' value='0'/>
<attributeValue name='Q_a2' value='0'/>
</node>

<node id='1'>
<attributeValue name='Q_a1' value='0'/>
<attributeValue name='Q_a2' value='0'/>
</node>

<!-- List of edges -->
<edge edgeID='0-1' source='0' target='1'
        type='add'>
<attributeValue name='Num_of_Interactions'
        value='0'/>
</edge>
</change>

<!-- Once any change occurs it will be added
        (node/edges  addition, removal, attribute
        values modification-->
<change time='5'>
<node id='1'>
<attributeValue name='Q_a1' value='0.1'/>
</node>
</change>
</DynamicNetwork>
```

A parser is implemented to represent the network data structure including all nodes, edges and their attributes. Us-

ing the data structure, the tool can extract network features such as the node degree over time.

B. Advanced Detailed Network Visualization

Visualization was and remains the crux of exploratory data analysis. Our tool facilitates visualizing dynamic networks by defining three components:

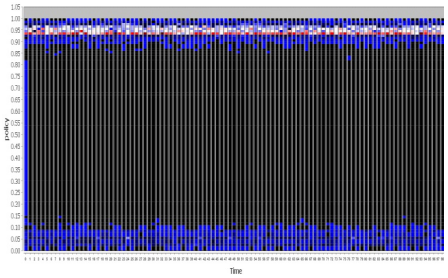
- a set of **visualizable features** for nodes and edges that is *defined by the tool*. The visualizable node features include the node size (integer: 1-100), shape (nominal: square, circle, diamond, etc), and three color components (3 integers reflecting the red, green, and blue color components). The visualizable edge features include the edge width, style (dashed, solid, dotted, etc), and three color components.
- a set of **logical attributes** for nodes and edges that is *defined in the input file*. The set can include for example the probability of choosing an action by an agent in general (node attribute) or for a particular neighbor (edge attribute).
- a mapping between logical attributes and visualizable features which specify which logical attributes are to be visualized and how. Figure 3 shows a snapshot of such mapping.

For effective visualization of large networks, our tool allows filtering and highlighting a subnetwork. The filtering feature enables the user to formulate simple queries (specifying values for logical attributes) or complex queries (combination of simple queries). Based on the formulated query the network representation will be updated to show only the nodes that satisfy the query. The highlighting feature is similar to filtering feature, where the subnetwork satisfying the query is highlighted, without deleting the rest of the network. We allow more than one subnetwork to be highlighted with different colors (reflecting multiple logical subnetworks). To visualize the changes in a network over time, play/stop and resume time line is provided through the SoNIA package.

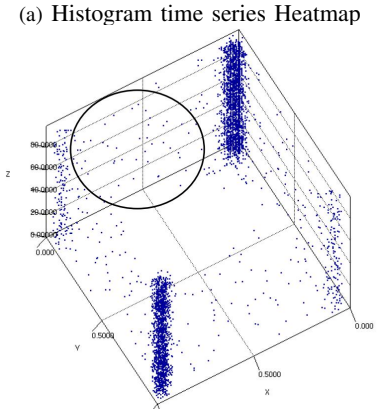
In addition to the logical attributes that are defined in the input file, our tool can compute network-based attributes (i.e. the attributes that depend on the network structure) such as the node degree. These computed attributes are then treated as normal logical attributes (as if the network-based attributes where read from the input file) and can be visualized accordingly.

C. Novel Network Analysis Visualizations

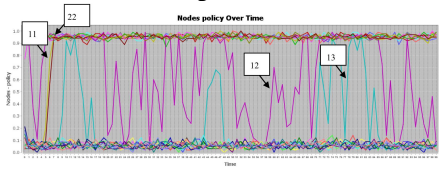
Visualizing the detailed network (with individual nodes and edges), even with filtering and highlighting, is difficult to use as the first step when analyzing a large network. Instead, one would prefer a summarized view of the network dynamics to start with and then go gradually into more *focused* and detailed view. Different novel summarizing graphs are supported by our tool, including:



(a) Histogram time series Heatmap



(b) 3-D edge visualization



(c) collective time series

Figure 4. Different Visualizations of DNVA. In the heatmap visualization the horizontal axis represents time, the vertical axis represents (logical) attribute value, and the color represents the frequency of a particular attribute value. The 3D-edge visualization and the collective time-series visualizations are described in Section III.

- Histogram time series heat-map
- 3-D edge visualization
- Collective time series

Figure 4 illustrates these visualizations, which are explained in more detail later when we discuss the case studies.

III. Case Studies

To evaluate the effectiveness of our tool, this section presents our analysis of two domains: social learning and taxi dispatch systems. The significant difference between the two domains stresses the generality of our tool.

A. Social Learning

Social learning studies how a group of agents interact and learn from one another to reach a *norm* [14]. A norm or a convention is an unwritten law that a society of agents agree on. Social norms are used by humans all the time. Choosing on which side of the road to drive a car and the right-of-way at an intersection are well-known examples. In a multi-agent

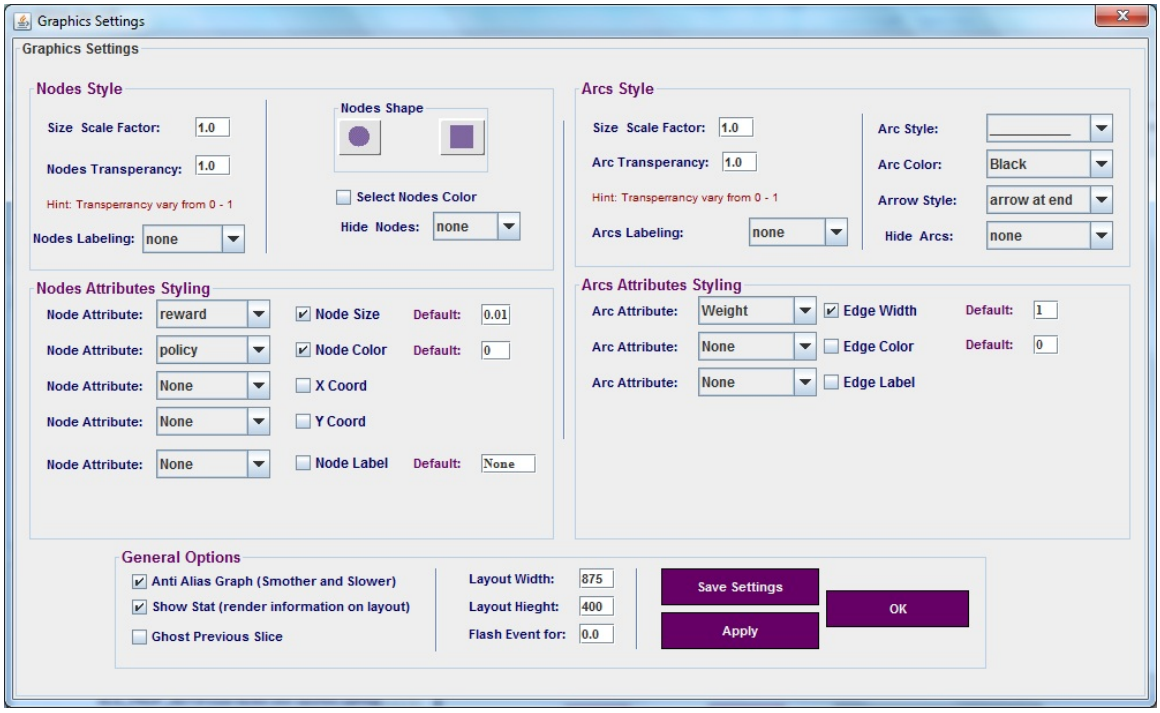


Figure 3. Snapshot of the tool showing the mapping between logical attributes and visualizable features.

setting, a convention may refer to a dominant coordination strategy, a common communication language, or the right of way among a group of robots. Upon establishing a norm, the overhead of coordination drops and the reliability of the multi-agent system increases [16]. When studying the emergence of norms and conventions, researchers typically assumed the interaction between agents to be random: a pair of agents were selected randomly to interact with one another. The process is then repeated until convergence. When agents are adaptive, the process is referred to as social learning. The coordination game is perhaps the most widely used game for studying social learning as it presents an agent community with two equally plausible norms to choose from [14]. Recently social leaning was studied in networks [17] where the underlying network restricted the interactions between agents. In such a setting, convergence to a global norm was no longer guaranteed as more than one (sub)convention might emerge concurrently and remain stable.[1] The reason for the emergence of multiple stable sub-conventions was the existence of a stable barrier that separated the sub-conventions from one another (or equivalently, prevented each convention from invading the other). Such a barrier or a frontier created a suboptimal equilibrium. The frontier effect was reported to either prevent or significantly slowdown the convergence to a global norm across variety of network types, particularly for scale-free networks [17].

[1]A sub-convention is a convention that is not adopted by the vast majority of the agents.

Here we study social leaning in scale-free networks, the most challenging type of networks for reaching a global consensus. Figure 5 shows the 3D visualization of the dynamics of this case (the z-axis represents time, from bottom to top). Each point in the x-y plane represents a network edge: the probability of choosing the first convention for the two agents across the edge. There is no single norm that agents converge to. The vast majority of edges connect two agents that adopt the the same norm (the bottom and the top corners of the 3D cube). There are few edges on the boundary of the two norms (the left and the right corners of the 3D cube). The unexpected part here is the nodes that keep appearing between the four corners (e.g. those enclosed by the black ellipse). With the simple coordination game, one would expect all agents to converge to one of the two norms and therefore all edges to converge to one of the four corners. This observation required further investigation to determine whether this was just a simulator bug or a genuine observation.

Figure 6 visualizes individual node policies as time series. A policy of a node is simply the probability of choosing the first convention. The vast majority of nodes settle to one of the two norms, as expected from Figure 5. However, there are few nodes that never settle and keep oscillating: Node 12 and Node 13. This phenomena was not reported by the previous research in social learning [17], [2].

To understand why such nodes behaved this way, we needed to visualize the network in detail. We used the

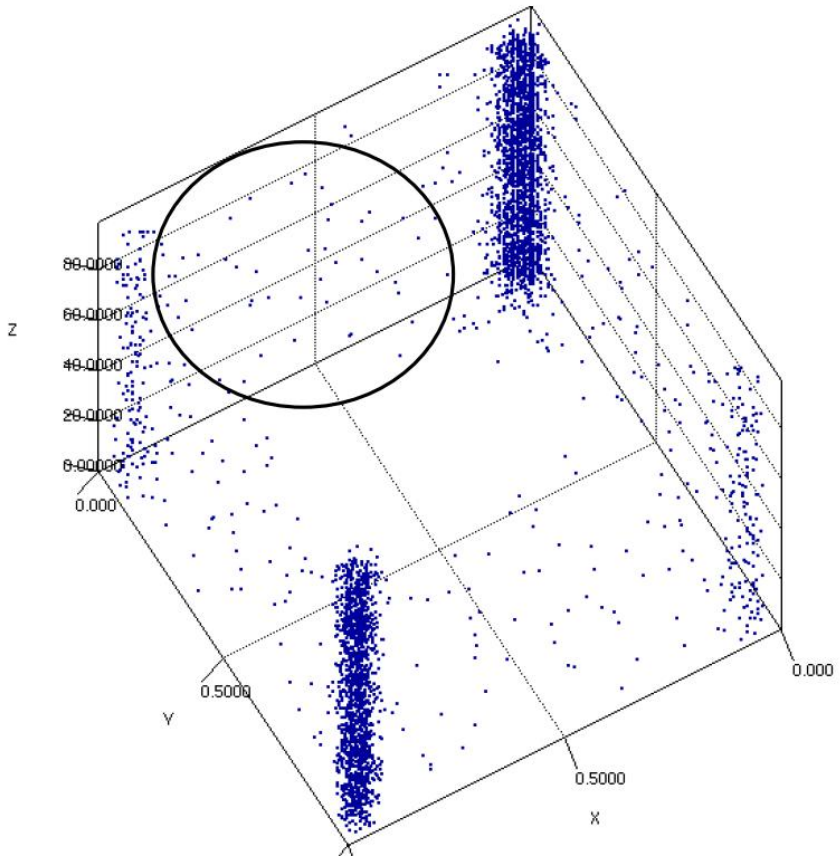


Figure 5. 3D visualization of the dynamics (the z-axis represents time, from bottom to top). Each point in the x-y plane represents a network edge: the probability of choosing the first convention for the two agents across the edge. There is no single norm that agents converge to. The vast majority of edges connect two agents that adopt the the same norm (the bottom and the top corners of the 3D cube). There are few edges on the boundary of the two norms (the left and the right corners of the 3D cube).

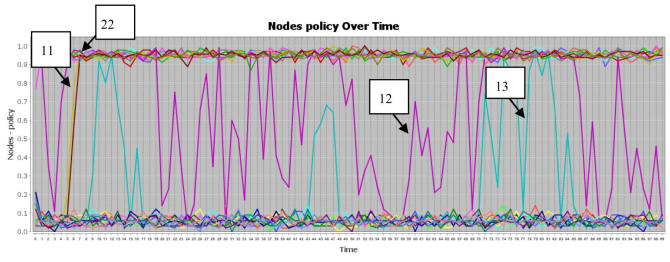


Figure 6. Visualizing individual node policies as time series. The vast majority of nodes settle to one of the two norms. However, there are two nodes that never settle and keep oscillating: Node 12 and Node 13. This phenomena was not reported by any of the previous research in social learning.

simple mapping from a node's policy (logical attribute) to the node's red color component (visual feature). Figure 7 visualizes the underlying network. Notice here that Node 12 and Node 13 (highlighted) are sandwiched between two stable subconventions. As a result, the two nodes are susceptible to random walks: there is no strict preference to one convention as opposed to the other and the two nodes

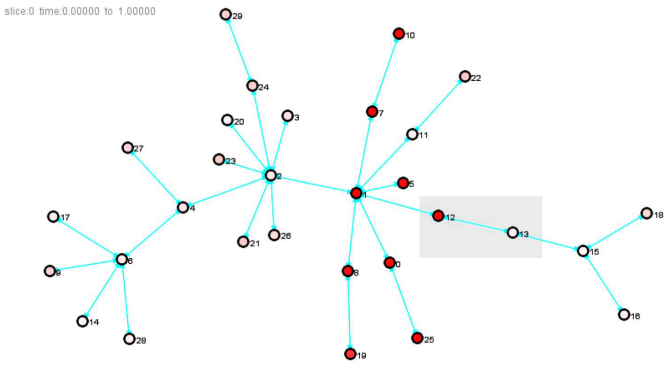


Figure 7. Visualization of the underlying network. Notice here that Node 12 and Node 13 (highlighted) are sandwiched between two stable subconventions and as a result they are susceptible to random walks.

will never converge.

B. Taxi Dispatch System

The second domain we analyzed is the taxi dispatch system. A taxi dispatch system is used to assign vacant

taxi to a customer in different locations. In a recent paper, multi-agent technologies were used to improve the performance of such system [4]. The system divided the city into regional dispatch area and adjacent areas are assigned to each dispatch area, therefore creating a network. An adjacent dispatch area is used when there is no vacant cabs in the local dispatch area. The underlying network was initially created by a human expert, and then later was restructured automatically through a self-organizing mechanism [4]. Here we analyze the data set representing the evolution of the taxi dispatch network and observe the change in the network properties. The original paper used the degree distribution to assess the change in the network due to self-organization. The degree distribution, however, showed very little change and that does not fully explain the significant improvement in performance.

We visualize the initial network (before self-organization) in Figure 8, where node color visualizes the node degree. We can see more than 27 isolated areas (indicated by red rectangles) that have no adjacent area. The isolated areas either suffer overload or under-utilization of cabs. Figure 9 visualizes the network at the end of the simulation and after applying the multi-agent self-organization technique. The orange edges represent new connections that were added through self-organization. Notice here although the change in the degree distribution was minimal, the change in some individual nodes was significant. For example Node 11 (highlighted) was initially isolated, but toward the end it got (in)degree of 7. Node 87 (highlighted) has degree of 1, but toward the end became one of the hubs. In fact, when the nodes with dark red color in the final network (hubs) were not hubs initially (Figure 8). This shows that the self-organizing mechanism used in the original work did not follow preferential attachment property.

IV. Related Work

Some researchers analyzed how individual agent policies co-evolved over time, either theoretically [9], [18], [3] or experimentally [11], as the number of agents increases, inspecting individual agent policies does not scale well with the size of the network. A more recent work used data mining techniques to extract patterns from log files of communicated messages between agents [15]. Inspecting individual messages exchanged between the agents becomes less practical and less useful as the network gets larger. Some of the previous work provided generic frameworks for analyzing multi-agent systems [12], [8], but these framework limited analysis to few agents and ignored the underlying network structure, unlike the work presented here. Most of the work in analyzing (communication) networks and distributed systems relied on simple heuristics that are intuitive, easy to understand, and experimentally verified to work adequately. Large-scale and in-depth analysis of the network behavior assumed the underlying nodes are relatively simple with fixed behavior [13]. Here we are interested in analyzing networks of learning agents, where the dynamics of the network change over time even if the outside world remains unchanged.

There has been several tools that were developed for analyzing social networks. The network analysis approach relied on summarizing the quantitative characteristics of a network using simple measures such as the degree distribution [1]. Example of such tools include the Social Network Analysis (SNA) tool [7], the open source tool Java Universal Net-work/Graph Framework (JUNG) [10], Gephi [5], and SoNIA [6]. These tools were used successfully for mining, visualizing and analyzing snapshot of a network or a graph. However, except for SoNIA that we extend in this work, none of these tools were specifically designed to handle dynamic networks.

V. Conclusion

In this paper we presented our tool for analyzing networks of adaptive agents. Unlike most of the similar tools, which either focused on analyzing a snapshot of a network or ignored the underlying network, our tool analyzes the learning dynamics over time in a network of agents at different levels of details. We showed how our tool helped us explain some interesting observations in two case studies: the social learning and the taxi dispatch domains.

One of the future directions we plan to pursue is to extend our tool and implement more advanced techniques, such as frequent sequence mining as well as novel visualization techniques that are suitable for multi-state learning dynamics.

References

[1] S. Abdallah. Generalizing unweighted network measures to capture the focus in interactions. *Social Network Analysis and Mining*, 1(4):255–269, 2011.

[2] S. Abdallah. Using a hierarchy of coordinators to overcome the frontier effect in social learning. In *Proceedings of the 8th International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, 2012.

[3] S. Abdallah and V. Lesser. A multiagent reinforcement learning algorithm with non-linear dynamics. *Journal of Artificial Intelligence Research*, 33:521–549, 2008.

[4] A. Alshamsi, S. Abdallah, and I. Rahwan. Multiagent self-organization for a taxi dispatch system. In *The International Joint Conference on Autonomous Agents and Multiagent Systems*, page (to appear), May 2009.

[5] M. Bastian, S. Heymann, and M. Jacomy. Gephi: An open source software for exploring and manipulating networks, 2009.

[6] S. Bender-deMoll and McFarland. The art and science of dynamic network visualization. *Journal of Social Structure*, 7(2), 2006.

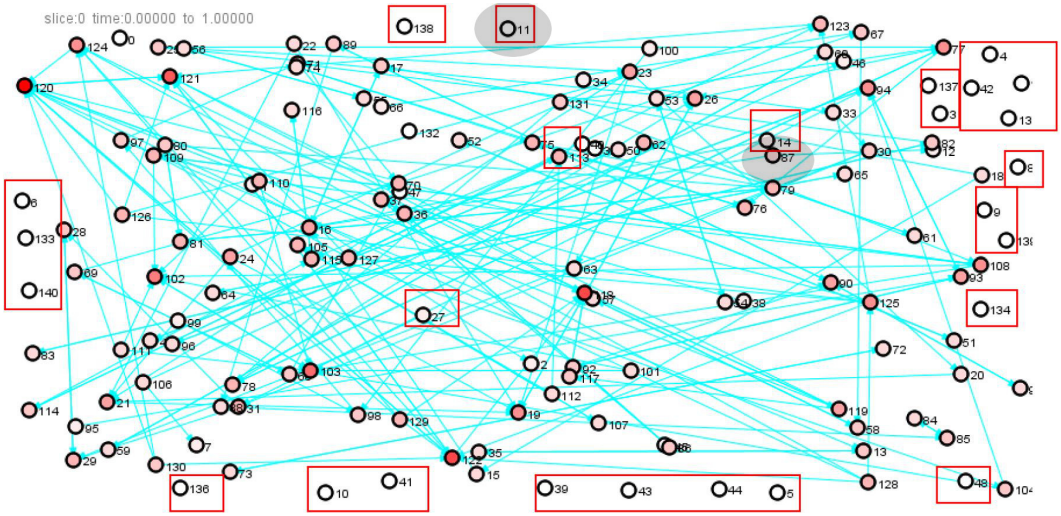


Figure 8. Visualizaition of the taxi network before reorganization.

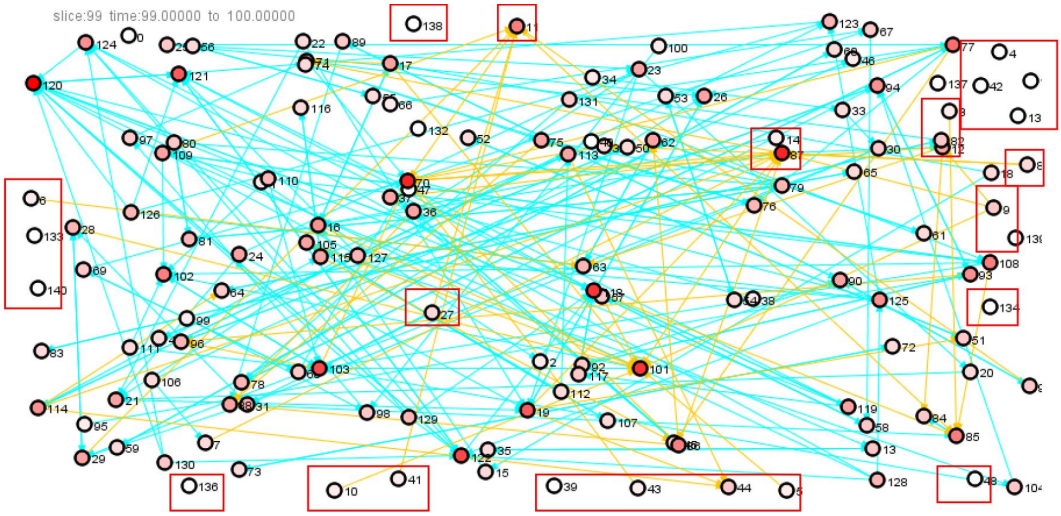


Figure 9. Visualization of the taxi network after the reorganization.

[7] A. Bohn, I. Feinerer, K. Hornik, and P. Mair. Content-Based Social Network Analysis of Mailing Lists. *The R Journal*, 3(1):11–18, June 2011.

[8] T. Bosse, D. N. Lam, and K. S. Barber. Tools for analyzing intelligent agent systems. *Web Intelligence and Agent Systems*, 6(4):355–371, 2008.

[9] M. Bowling and M. Veloso. Multiagent learning using a variable learning rate. *Artificial Intelligence*, 136(2):215–250, 2002.

[10] D. Fisher, J. O'madadhain, P. Smyth, S. White, and Y.-B. Boey. Analysis and Visualization of Network Data using JUNG. *Journal of Statistical Software*.

[11] M. Ghavamzadeh, S. Mahadevan, and R. Makar. Hierarchical multi-agent reinforcement learning. *Autonomous Agents and Multi-Agent Systems*, 13(2):197–229, 2006.

[12] J. Jin, R. T. Maheswaran, R. Sanchez, and P. Szekely. Vizscript: visualizing complex interactions in multi-agent systems. In *Proceedings of the international conference on Intelligent user interfaces*, pages 369–372, 2007.

[13] V. Paxson. End-to-end routing behavior in the internet. *SIGCOMM Comput. Commun. Rev.*, 36(5):41–56, 2006.

[14] S. Sen and S. Airiau. Emergence of norms through social learning. In *Proceedings of the international joint conference on Artifical intelligence*, IJCAI, pages 1507–1512, San Francisco, CA, USA, 2007. Morgan Kaufmann Publishers Inc.

[15] E. Serrano, J. J. Gómez-Sanz, J. A. Botía, and J. Pavón. Intelligent data analysis applied to debug complex software systems. *Neurocomputing*, 72(13-15):2785–2795, 2009.

[16] T. Sugawara. Emergence and stability of social conventions in conflict situations. In *Proceedings of the international joint conference on Artifical intelligence*, pages 371–378, 2011.

[17] D. Villatoro, J. Sabater-Mir, and S. Sen. Social instruments for robust convention emergence. In *IJCAI*, pages 420–425, 2011.

[18] P. Vrancx, K. Tuyls, and R. Westra. Switching dynamics of multi-agent learning. In *Proceedings of the 7th international joint conference on Autonomous agents and multiagent systems*, pages 307–313, Richland, SC, 2008. International Foundation for Autonomous Agents and Multiagent Systems.