Using Graph Analysis to Study Networks of Adaptive Agent

Sherief Abdallah
British University in Dubai, United Arab Emirates
University of Edinburgh, United Kingdom
shario@ieee.org

ABSTRACT

Experimental analysis of networks of cooperative learning agents (to verify certain properties such as the system's stability) has been commonly used due to the complexity of theoretical analysis in such cases. Due to the large number of parameters to analyze, researchers used metrics that summarize the system in few parameters. Since in cooperative system the ultimate goal is to optimize some global metric, researchers typically analyzed the evolution of the global performance metric over time to verify system properties. For example, if the global metric improves and eventually stabilizes, it is considered a reasonable verification of the system's stability.

The global performance metric, however, overlooks an important aspect of the system: the network structure. We show an experimental case study where the convergence of the global performance metric is deceiving, hiding an underlying instability in the system that later leads to a significant drop in performance. To expose such instability, we propose the use of the graph analysis methodology, where the network structure is summarized using some network measures. We develop a new network measure that summarizes an agent's interaction with its neighbors and takes the *disparity* of these interactions into account. The new measure is applied to our case study, clearly exposing the instability that was previously hidden by the global performance metric.

Categories and Subject Descriptors

I.2.6 [**Artificial Intelligence**]: Learning; I.2.11 [**Artificial Intelligence**]: Distributed Artificial Intelligence

General Terms

Experimentation

Keywords

Simulation and Experimental Verification, Agent Networks, Multi-agent Learning, Network Analysis

1. INTRODUCTION

Cite as: Using Graph Analysis to Study Networks of Adaptive Agent, Sherief Abdallah, *Proc. of 9th Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS 2010)*, van der Hoek, Kaminka, Lespérance, Luck and Sen (eds.), May, 10–14, 2010, Toronto, Canada, pp. XXX-XXX.

Organizing agents in a network is a common approach to achieve scalability in multi-agent systems [1]. Restricting an agent to interact only with its neighbors simplifies the agent's decision making problem (the problem becomes independent of the overall system size). To cope with an environment that keeps changing or that is not known a prior, learning algorithms have been used to optimize performance in agent networks [2, 3, 1]. Multi-agent learning algorithms allow agents to optimize their decisions based on their interactions with both the environment and their neighbors in the network. Analyzing the dynamics of such a system (consisting of networked adaptive agents) over time is a non-trivial task, due to the large number of system parameters (each agent maintains a set of local parameters controlling its behavior), the concurrency by which these parameters change (agents acting independently), and the delay in the effect/consequence of parameter changes (because of communication delay between agents and the time it takes for learning algorithms to adapt).

A researcher studying a large-scale system of adaptive agents needed some metrics to summarize this system (with its large number of parameters) into a few numbers that are more manageable. In cooperative MAS, the natural choice was the global performance metric(s) the system is trying to optimize (e.g. payoff). Researchers inspected the evolution of some global performance metrics as a rough approximation to the internal dynamics of the adaptive agent network. [2, 3, 1]. For example, if the global metric improved over time and eventually *appeared* to stabilize, it was usually considered a reasonable verification of convergence. Examples of global performance metrics include the percentage of total number of delivered packets in routing problems [4], the average turn around time of tasks in task allocation problems [5], or the average reward (received by agents) in general [6].

The global performance metric, however, overlooks an important aspect of the system: the network structure. We present in this paper an experimental case study where the stability of the global performance metric can hide an underlying instability in the system. The hidden instability later leads to a significant drop in the global performance metric itself, but after some period of (fake) stability.

We propose the use of graph analysis to study the dynamics of networked adaptive agents. Graph analysis is an interdisciplinary research field that studies interesting graph properties in real-world networks. In recent years graph analysis received significant attention due to the explosive growth of social networks and the discovery of common patterns that govern a wide-range of real world networks [7, 8].

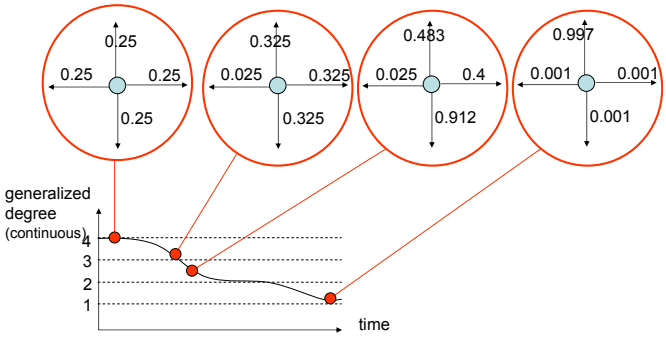

Figure 1: Example showing the evolution of a node's interaction with its neighbors over time (top) and the corresponding continuous degree of that node over time.

The core of graph analysis is *network measures*: functions that summarize a graph to simpler numeric values which are then more manageable to analyze.

One can abstract a networked MAS at any point in time as a weighted graph, where an edge weight reflects the amount of interaction over that edge (e.g. the number of messages exchanged). Figure 4(a) shows an example network of 4 agents. We can then apply some network measures that were developed in graph analysis to summarize these weighted graphs into meaningful more manageable parameters. However, the bulk of the measures previously developed in graph analysis focused on unweighted graphs [9]. The lack of weighted network measures is due, at least in part, to the difficulty of quantifying such weights in real-world networks (e.g. social networks). Even if quantified, the accuracy of such weights are at best debatable. In our case here networks are artificial and weights can be quantified precisely. In fact, weights can not be ignored, because they provide crucial insights into the learning process going on inside the network of agents.

We propose a new network measure, which we call the *continuous degree*. The new measure takes weights into account to determine how many neighbors are effectively being used by an agent in the network. Figure 1 illustrates how our new measure summarizes the evolution of a node's interaction over time. When initially the agent interacts uniformly with all its neighbors, the continuous degree equals 4, the number of neighbors. At the other extreme when the agent interacts mostly with one neighbor, the continuous degree equals 1. We prove interesting properties of our proposed measure, including its connection to the traditional node degree and its ability to capture the disparity of interaction among neighbors. We use our measure to expose and explain the instability reported in the case study we mentioned earlier.

In summary, this paper makes the following contributions:

- presents a case study illustrating the risk of relying on global performance metrics to analyze networks of adaptive agents.
- proposes the use of graph analysis to analyze networks of adaptive agents, and develops a new network measure that takes the disparity of interaction among agents into account
- illustrates how graph analysis can be used to explain the observed behavior in our case study.

The paper is organized as follows. Section 2 describes the case study we will be using throughout the paper. Section 3 presents our proposed measure, the continuous degree. Section 4 revisits the case study, illustrating how the continuous degree exposes the instability in the system. Section 5 analyzes our proposed measure and proves some of its properties. Section 6 discusses the related work. We conclude in Section 7.

2. CASE STUDY: DISTRIBUTED TASK ALLOCATION PROBLEM (DTAP)

We use a simplified version of the distributed task allocation domain (DTAP) [5], where the goal of the system is to assign tasks to agents such that the service time of each task is minimized. For illustration, consider the example scenario depicted in Figure 2. Agent A0 receives task T1, which can be executed by any of the agents A0, A1, A2, A3, and A4. All agents other than agent A4 are overloaded, and therefore the best option for agent A0 is to forward task T1 to agent A2 which in turn forwards the task to its left neighbor (A5) until task T1 reaches agent A4. Although agent A0 does not know that A4 is under-loaded (because agent A0 interacts only with its immediate neighbors), agent A0 should eventually learn (through experience and interaction with its neighbors) that sending task T1 to agent A2 is the best action without even knowing that agent A4 exists.

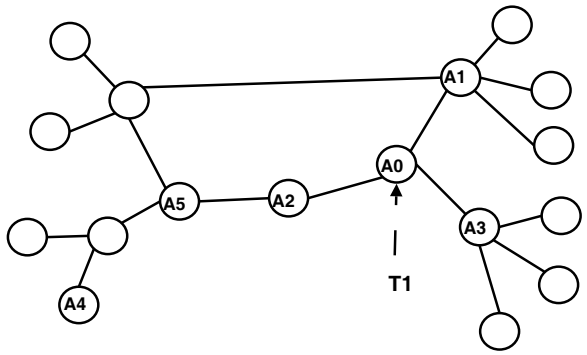

Figure 2: Task allocation using a network of agents.

Each time unit, agents make decisions regarding all task requests received during this time unit. For each task, the agent can either execute the task locally or send the task to a neighboring agent. If an agent decides to execute the task locally, the agent adds the task to its local queue, where tasks are executed on a first come first serve basis, with unlimited queue length.

The main goal of DTAP is to reduce the total service time, averaged over tasks, $ATST = \frac{\sum_{T \in \overline{T}_\tau} TST(T)}{|\overline{T}_\tau|}$, where $\overline{T}_\tau$ is the set of task requests received during a time period τ and $TST(T)$ is the total time a task T spends in the system. The $TST(T)$ time consists of the time for routing a task request through the network, the time the task request spends in the local queue, and the time of actually executing the task.

Agents interact via two types of messages. A REQUEST message $\langle i, j, T \rangle$ indicates a request sent from agent i to agent j requesting the execution of task T. An UPDATE

message $\langle i, j, T, R\rangle$ indicates a feedback (reward signal) from agent i to agent j that task T took R time steps to complete (the time steps are computed from the time agent i received T's request).

Communication delay is an important property of our version of DTAP. Each agent has a physical location and the communication delay between two agents is proportional to the Euclidean distance between them (one time unit per distance unit). Due to communication delay, the effect of an action does not appear immediately because it is communicated via messages and messages take time to route. Not only is the reward delayed but so is any change in the system's state. A consequence of communication delay is partial observability: an agent can not observe the full system state (the queues at every other agent, messages on links and in queues, etc.).

Although the underlying simulator has different underlying states (tasks at local agent queues, messages in transit over communication links, etc.), we made agents oblivious to these states. The only feedback an agent gets is its own reward. This simplifies the agent's decision problem and reemphasizes partial observability: agents collectively learn a joint policy that makes a good compromise over the different unobserved states. The following section gives a brief overview of the multi-agent learning (MAL) algorithms that we evaluate in our case study and discusses the issue of convergence with respect to MAL.

2.1 Multiagent Learning

A large number of multi-agent learning algorithms were proposed that vary in their underlying assumptions and target domains [10]. The experimental results and the analysis we report here focus on two (gradient-ascent) multi-agent learning algorithms: GIGA-WoLF [11] and WPL [5], but our methodology can be used with other algorithms as well. These two learning algorithms allow agents to learn stochastic policies, which are better suited for partially-observable domains [1]. The specifics of WPL and GIGA-WoLF (such as their update equations, the underlying intuition, their differences and similarities) are neither relevant to the purpose of this paper nor needed to follow our analysis in Section 4. Nevertheless, and for completeness, we mention below (very briefly) the equations for updating the policy for the two algorithms. Further details regarding the two algorithms can be found elsewhere [11, 5].

The term *convergence*, in the reinforcement learning context, refers to the stability of the learning process (and the underlying model) over time. Similar to single agent reinforcement learning algorithms (such as Q-learning [12]), the convergence of a multi-agent reinforcement learning (MARL) algorithm is an important property that received considerable attention [13, 11, 14, 15, 16, 5]. However, proving the convergence of a MARL algorithm via theoretical analysis is significantly more challenging than proving the convergence in the single agent case. The presence of other agents that are also learning deem the environment non-stationary, therefore violating a foundational assumption in single agent learning. In fact, proving the convergence of a MARL algorithm even in 2-player-2-action single-stage games (arguably the simplest class of multi-agent systems domains) has been challenging [13, 14, 5].

An agent i using WPL updates its policy π_i according to the following equations:

$$\forall j \in neighbors(i) : \Delta\pi_i^{t+1}(j) \leftarrow \frac{\partial V_i^t(\pi)}{\partial \pi_i^t(j)} \cdot \eta \cdot \begin{cases} \pi_i^t(j) & \text{if } \frac{\partial V_i^t(\pi)}{\partial \pi_i^t(j)} < 0 \\ 1 - \pi_i^t(j) & \text{otherwise} \end{cases}$$

$$\pi_i^{t+1} \leftarrow projection(\pi_i^t + \Delta\pi_i^{t+1})$$

where η is a small learning constant and $V_i(\pi)$ is the expected reward agent i would get if all agents (including agent i) follow the joint policy π (agent i's part of π is its own policy π_i). The *projection* function ensures that after adding the gradient $\Delta\pi_i$ to the policy, the resulting policy is still valid.

An agent i using GIGA-WoLF updates its policy π_i according to the following equations:

$$\hat{\pi}_i^{t+1} = projection(\pi_i^t + \eta V_i^t(\pi)^t)$$
$$z_i^{t+1} = projection(\pi_i^t + \eta V_i^t(\pi)/3)$$
$$\delta_i^{t+1} = \min\left(1, \frac{||z_i^{t+1} - z_i^t||}{z_i^{t+1} - \hat{\pi}_i^t}\right)$$
$$\pi_i^{t+1} = \hat{\pi}_i^{t+1} + \delta_i^{t+1}(z_i^{t+1} - \hat{\pi}_i^{t+1})$$

The following section evaluates both WPL and GIGA-WoLF in DTAP using ATST as the metric of evaluation.

2.2 Stability Under the Global Metric Results

We have evaluated the performance of WPL and GIGA WoLF using the following setting.100 agents are organized in a 10x10 grid. Communication delay between two adjacent agents is two time units. Tasks arrive at the 4x4 sub-grid (i.e. the 16 agents at the center) at the center at rate 0.5 tasks per agent per time unit. All agents can execute a task with a rate of 0.1 task/time unit (both task arrival and service durations follow an exponential distribution).

Figure 3 plots the global performance (measured in terms of ATST) of the two multi-agent learning algorithms in the DTAP domain. Just by looking at the ATST plot, it is relatively safe to conclude that WPL converges quickly while GIGA-WoLF converges after about 75,000 time steps. This is consistent with previously reported results that were conducted on a much simpler version of DTAP (only 5 agents with no communication delay) [17]. The following section presents a new network measure that we later use to discover that the apparent stability of GIGA-WoLF is actually misleading.

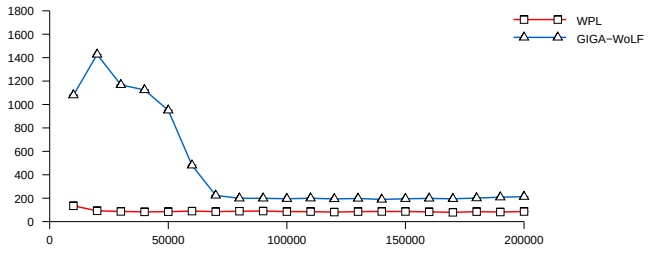

Figure 3: Comparing the average total service time for 200,000 time steps of the DTAP problem for WPL and GIGA-WoLF.

3. THE CONTINUOUS DEGREE (C-DEGREE)

A key measurement that has been used extensively in analyzing networks is the *degree* of a node. A node's degree is the number of edges incident to that node. Intuitively, a node's degree reflects how connected the node is. This simple measure (along with other network measures) allowed the discovery of patterns common in many real world networks, such as the power law of the degree distribution [18].

To put it more formally, a network (graph) is defined as $N = \langle V, E \rangle$, where V is the set of network nodes (vertices) and E is the set of edges (links) connecting these nodes. The degree of a node $v \in V$ is $k(v) = |E(v)|$, where $E(v)$ is the set of edges incident to node v. For a weighted network, we define the function $w(e)$ which returns the weight of edge $e \in E$. Also for convenience, we define the function $W(v) = \{W(e) : e \in E(v)\}$, i.e. the multiset of weights incident to node v. The degree distribution $P(k)$ measures the frequency of a particular degree k in a network ($P(k = u) = |\{v : v \in V \wedge k(v) = u\}|$) and serves as a common method for summarizing and characterizing networks.

One of the limitations of the degree measure is that it ignores any disparity in the interaction between a node and its neighbors. In other words, the degree measure assumes uniform interaction across each node's neighbors. This can result in giving an incorrect perception of the *effective* node degree. For example, an agent may have 10 or more neighbors but mainly interacts with only two of them. Should that agent be considered 2 times more connected than an agent with only 5 neighbors but also interacting primarily with two of them?

We introduce in this paper a new measure for analyzing weighted networks: the *continuous degree*, or the *C-degree*.

DEFINITION 1. *The* C-degree *of a node v in a network is $r(v)$, where*

$$r(v) = \begin{cases} 0 & \text{if } v \text{ is disconnected} \\ 2^{\left(\sum_{e \in E(v)} \frac{w(e)}{s(v)} \log_2 \frac{s(v)}{w(e)}\right)} & \text{otherwise} \end{cases}$$

Where $s(v) = \sum_{w \in W(v)} w$ is called the strength of node v. Intuitively, the quantity $\frac{w(e)}{s(v)}$ represents the probability of an interaction over an edge e. The set $\left\{(e, \frac{w(e)}{s(v)}) : e \in E(v)\right\}$ is the interaction probability distribution for node v. The quantity $H(v) = \sum_{e \in E(v)} \left[\frac{w(e)}{s(v)} \log_2 \frac{s(v)}{w(e)}\right]$ is then the entropy of the interaction probability distribution, or how many bits are needed to encode the interaction probability distribution. The entropy quantifies the disparity in the interaction distribution: the more uniform the interaction distribution is, the higher the entropy and vice versa.[1] The purpose of the power 2 is to convert the entropy back to the number of neighbors that are effectively being used.

Figure 4 compares the continuous degree distribution to the (discrete) degree distribution in a simple weighted network of four nodes. A node on the boundary has an out degree of 1, while an internal node has an out degree of 2. Intuitively, however, only one of the internal nodes is fully utilizing its degree of 2 (the one to the left), while the other node (to the right) is mostly using one neighbor only. The C-degree measure captures this and shows that only one internal node has a C-degree of 2 while the other internal node has a C-degree of 1.38.

[1]This is in contrast to the Y measure, which decreases if the interaction distribution becomes more uniform. A brief discussion of the Y-measure is given in Section 6

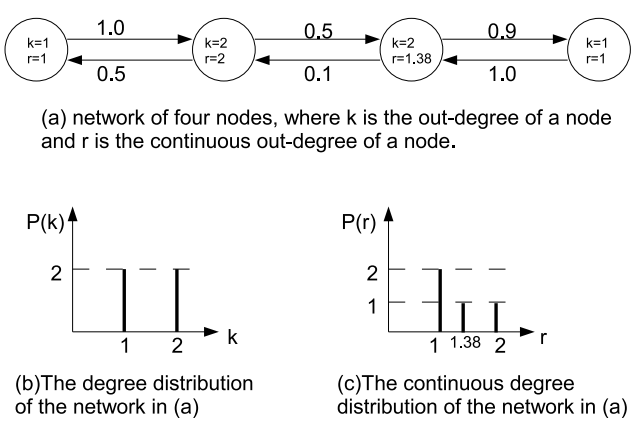

Figure 4: Continuous vs discrete degree distributions.

What sets our measure apart from previous work in graph analysis is that it is a continuous generalization of the degree measure that captures the disparity of interaction (the difference in weights) among the neighbors of a node. In particular, if every node interacts with all its neighbors equally, then the C-degree becomes identical to traditional (discrete) degree measure of the same node. However, if there is a disparity in a node's interaction with its neighbors, then the C-degree will capture such disparity, unlike the traditional degree measure. We prove these properties of the C-degree in Section 5, but before doing so, let us illustrate how the C-degree can help in verifying the stability of a network of adaptive agents.

4. VERIFYING STABILITY USING C-DEGREE

Returning to our case study (refer to Section 2), Figure 5 plots the average C-degree over the 100 agents against time. When agents are using WPL, the C-degree do stabilize. When agents use GIGA-WoLF, however, the C-degree does not converge and continues to decrease. This observation suggests that we need to run the simulation for a longer time period in order to verify GIGA-WoLF's stability.

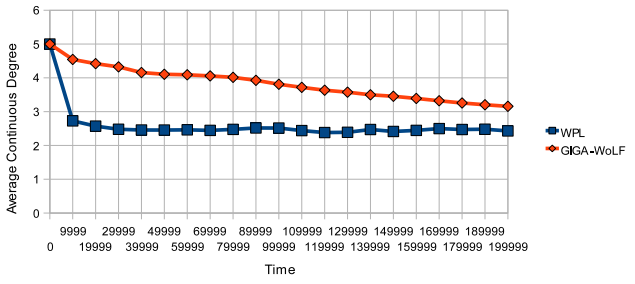

Figure 5: The average C-degree of WPL and GIGA-WoLF for 200,000 time steps.

When the simulator is allowed to run for 600,000 time

steps, the global performance metric (the ATST in the case of DTAP) of GIGA-WoLF starts slowly to diverge after 250,000 time steps and the corresponding C-degree continues to decrease. WPL's C-degree remains stable, consistent with WPL's the global performance metric.

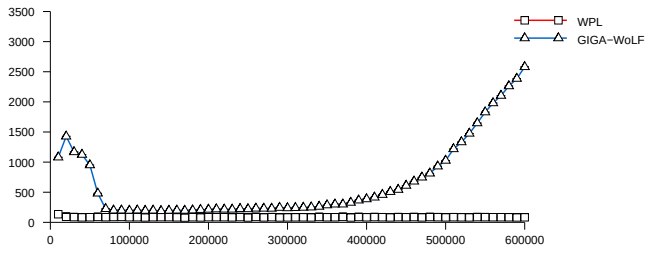

Figure 6: Comparing the ATST for 600,000 time steps of the DTAP problem for WPL and GIGA-WoLF.

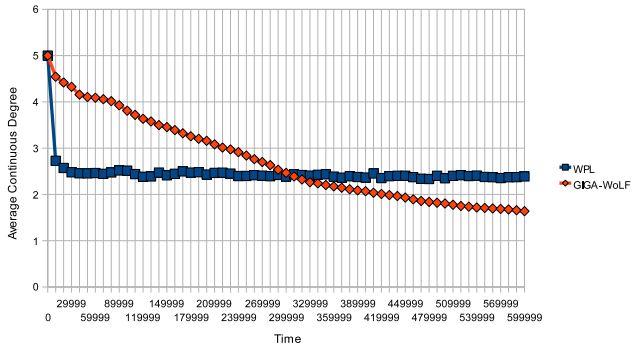

Figure 7: The average C-degree of WPL and GIGA-WoLF for 600,000 time steps.

The plot of the C-degree tells us that GIGA-WoLF learns as a slower pace than WPL. Furthermore, while agents using WPL continue to effectively use (at least) two neighbors, agents using GIGA-WoLF almost learn a deterministic policy (interacting primarily with only one neighbor). It is interesting to note that the average payoff for GIGA-WoLF starts to diverge when the average C-degree of GIGA-WoLF falls below the average C-degree of WPL. GIGA-WoLF's divergence in this case study is not surprising in itself, since GIGA-WoLF's convergence was proved only for 2-player-2-action games and GIGA-WoLF was reported to diverge in some games with larger number of either players or actions [11, 19]. A more in-depth analysis that take the specifics of GIGA-WoLF into account is needed to fully explain the root of GIGA-WoLF's divergence in the DTAP domain. Such an analysis is beyond the scope of this paper and should not distract from the main point we are trying to make: the common practice of using a global performance metric to verify the stability of an adaptive MAS is not reliable and can hide threatening instability. Graph analysis can provide useful insights in that respect when equipped with informative network measures, such as the measure we have presented.

5. PROPERTIES OF THE C-DEGREE

The C-degree satisfies three properties that establish its connection to the original degree metric (first two properties) and its ability to capture the disparity of interactions among the neighbors of a node (the third property).

1. **Preserving maximum degree:** $\forall v \in V : r(v) \leq k(v)$. Furthermore, $r(v) = k(v)$ iff $\forall e \in E(v) : w(e) = C$, where C is some constant. In other words, the C-degree is maximum and equals the original degree when there is no disparity between weights.

2. **Preserving minimum degree:** $\forall v \in V : r(v) = 0$ iff v is disconnected. Furthermore, $r(v) = 1$ iff $\exists e \in E(v) : w(e) > 0$ and $\forall e' \in E(v) \wedge e' \neq e : w(e') = 0$. In other words, the C-degree equals one when all edges, except one edge, have zero weight.

3. **Consistent partial order over nodes:** The C-degree imposes a simple partial order that is consistent with the above two properties. If the two nodes have the same number of neighbors, the same summation of weights (strength), and the individual edge weights are the same except for two edges, then the node with more uniform weights has higher C-degree. A formal definition of this property is given in Lemma 4.

The intuition of the three properties can be clarified through the following numeric example. Let $v1, v2, v3$, and $v4$ be four nodes where $W(v1) = \{5, 5, 5, 5\}, W(v2) = \{9, 5, 5, 1\}, W(v3) = \{9, 8, 2, 1\}$ and $W(v4) = \{20, 0, 0, 0\}$. The third property then guarantees that the C-degree imposes the ordering: $r(v1) > r(v2) > r(v3)$. All three properties guarantee that $k(v1) = r(v1) > r(v2) > r(v3) > r(v4) = 1$. Below we prove the three properties.

LEMMA 2. *The C-degree satisfies the minimum degree axiom.*

PROOF. When all weights are zero except only one weight that is greater than zero, then the entropy (the exponent of the C-degree) is zero, and therefore the C-degree is 1. □

LEMMA 3. *The C-degree satisfies the maximum degree axiom.*

PROOF. Under uniform interaction, all the weights incident to a node v are equal to a constant W_v. Therefore

$$\forall v \in V, e \in E(v) : \frac{w(e)}{s(v)} = \frac{W_v}{\sum_{e \in E(v)} W_v} = \frac{1}{k(v)}$$

We then have

$$\begin{aligned} \forall v : r(v) &= 2^{\sum_{e \in E(v)} \frac{w(e)}{s(v)} \log_2 \frac{s(v)}{w(e)}} \\ &= 2^{\sum_{k(v)} \frac{1}{k(v)} \log_2 (k(v))} \\ &= 2^{\log_2 (k(v))} \\ &= k(v) \end{aligned}$$

In other words, both the degree and the C-degree of a node become equivalent under uniform interaction. The C-degree is also maximum in this case, because the exponent is the entropy of the interaction distribution, which is maximum when the interaction is uniform over edges. □

LEMMA 4. *The C-degree satisfies the consistent partial order property.*

PROOF. Let i, j be two nodes such that $k(i) = k(j) = n, s(i) = s(j) = s, |W(i) \bigcap W(j)| = n-2, \{w_{i1}, w_{i2}\} = W(i) - W(j), \{w_{j1}, w_{j2}\} = W(j) - W(i)$, and $|w_{i1} - w_{i2}| < |w_{j1} - w_{j2}|$. Without loss of generality, we can assume that $w_{i1} \geq w_{i2}$ and $w_{j1} \geq w_{j2}$, therefore $w_{i1} - w_{i2} < w_{j1} - w_{j2}$. We also have

$$\frac{w_{i1} + w_{i2}}{s} = 1 - \sum_{w \in W(i) \bigcap W(j)} \frac{w}{s} = \frac{w_{j1} + w_{j2}}{s} = c$$

, therefore

$$\frac{w_{j1}}{s} > \frac{w_{i1}}{s} \geq \frac{c}{2} \geq c - \frac{w_{i1}}{s} > c - \frac{w_{j1}}{s}$$

Then from Lemma 5 we have $h(c, \frac{w_{i1}}{s}) > h(c, \frac{w_{j1}}{s})$, or

$$-\frac{w_{i1}}{s}\log_2(\frac{w_{i1}}{s}) - (c - \frac{w_{i1}}{s})\log_2(c - \frac{w_{i1}}{s}) >$$
$$-\frac{w_{j1}}{s}\log_2(\frac{w_{j1}}{s}) - (c - \frac{w_{j1}}{s})\log_2(c - \frac{w_{j1}}{s})$$

Therefore $H(i) > H(j)$, because the rest of the entropy terms (corresponding to $W(i) \bigcap W(j)$) are equal, and consequently $r(i) > r(j)$. □

LEMMA 5. *The quantity $h(C, x) = -x\log_2(x) - (C - x)\log_2(C - x)$ is symmetric around and maximized at $x = \frac{C}{2}$ for $C \geq x \geq 0$.*

PROOF.

$$h(C, \frac{C}{2} + \delta) = -(\frac{C}{2} + \delta)\log_2(\frac{C}{2} + \delta) - (\frac{C}{2} - \delta)\log_2(\frac{C}{2} - \delta)$$

$$= h(C, \frac{C}{2} - \delta)$$

Therefore $h(C, x)$ is symmetric around $c/2$. Furthermore, $h(C, x)$ is maximized when

$$\frac{\partial h(C, x)}{\partial x} = 0 = -1 - \log_2 x + 1 + \log_2(C - x)$$

or

$$\log_2 x = \log_2(C - x)$$

Therefore $h(C, x)$ is maximized at $x = C - x = \frac{C}{2}$. □

The following section discusses the similarities and differences between the C-degree and previously developed network measures in the field of graph analysis.

6. RELATED WORK

Some researchers analyzed how individual agent policies co-evolved over time, either theoretically [13, 20, 5] or experimentally [6], as the number of agents increases, but inspecting individual agent policies does not scale well with the size of the network. A recent work used data mining techniques to extract patterns from log files of communicated messages between agents [21]. 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, but these framework limited analysis to a few agents and ignored the underlying network structure, unlike the work presented here [22, 23]. 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 received attention recently, assuming the underlying nodes are relatively simple with fixed behavior [24]. 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.

Surveying all network measures that were proposed to analyze weighted graphs is beyond the scope of this paper. Instead, we focus on a sample of these measures that are mostly related to our proposed measure (the interested reader may refer to survey papers on the subject such as [9]).

The strength of a node (the summation of weights incident to the node) becomes identical to the node's degree if all weights are equal to 1. The strength measure, however, fails to capture the disparity of interaction between an individual node and its neighbors (the consistent partial order property). For example, suppose that a node v_1 has multiset of incident weights $\{1, 9\}$, while another node v_2 has a multiset of incident weights $\{5, 5\}$. Both nodes will have the same strength, despite the fact that node v_2 is interacting with its neighbors more uniformly than node v_1. A more recent work [25] analyzed a graph's total weight, $\sum_{e \in E} w(e)$, against the graph's total number of edges, $|E|$, over time. That work also analyzed the degree of a node, $k(v)$, against the node's strength, $s(v)$. These measures again fail to capture the disparity in interaction between a node and its neighbors.

The network measure $Y(v) = \sum_{e \in E(v)} \left(\frac{w(e)}{s(v)}\right)^2$ successfully captures the disparity of interaction within a node v [26]. However, the Y measure is not a generalization of the degree measure as it fails to satisfy the first two properties of the C-degree, therefore the Y-measure is less intuitive than the C-degree. An interesting method for generalizing the node degree is generating an ensemble of unweighted networks that are sampled from the original weighted network [27] (the weight of an edge reflects the probability of generating the edge in a sample network). The effective node degree is then the average over the samples. While the ensemble approach satisfies the first two properties of the C-degree, it still fails to satisfy the third property, the consistency in handling disparity.

7. CONCLUSION AND FUTURE WORK

In this paper we presented a case study of 100 networked adaptive agents where the global performance metric can hide an underlying instability in the system, and show that this instability lead to a significant drop in performance later on. We proposed the use of graph analysis to analyze agent interactions over time. We also developed a new network measure, the C-degree, that generalized the degree of a node to take weights into account. Our methodology successfully exposed the hidden instability in the case study. We finally proved that the C-degree of a node reduces to the original (discrete) degree when the node interacts with its neighbors uniformly. Otherwise the C-degree captures the disparity in a node's interaction with its neighbors.

The analysis presented here, as mentioned earlier, is a preliminary step that we hope to trigger more research that applies network analysis and mining techniques to multi-agent

systems. The C-degree, which we proposed here, focuses on capturing the *disparity of interaction* between a node and its neighbors. In domains where the (absolute) quantity of interaction is crucial in understanding the dynamics, alternatives to the C-degree may be more effective.

8. REFERENCES

[1] S. Abdallah, V. Lesser, Multiagent reinforcement learning and self-organization in a network of agents, in: Proceedings of the International Joint Conference on Autonomous Agents and Multiagent Systems, 2007, pp. 1–8.

[2] J. A. Boyan, M. L. Littman, Packet routing in dynamically changing networks: A reinforcement learning approach, in: Proceedings of the Annual Conference on Advances in Neural Information Processing Systems, 1994, pp. 671–678.

[3] L. Peshkin, V. Savova, Reinforcement learning for adaptive routing, in: Proceedings of the International Joint Conference on Neural Networks, 2002, pp. 1825–1830.

[4] Y.-H. Chang, T. Ho, Mobilized ad-hoc networks: A reinforcement learning approach, in: Proceedings of the First International Conference on Autonomic Computing, 2004, pp. 240–247.

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

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

[7] M. E. J. Newman, Finding community structure in networks using the eigenvectors of matrices, Physical Review E 74 (2006) 036104.

[8] J. Park, A.-L. Barabasi, Distribution of node characteristics in complex networks, Proceedings of the National Academy of Science 104 (2007) 17916–17920.

[9] S. Boccaletti, V. Latora, Y. Moreno, M. Chavez, D.-U. Hwang, Complex networks: Structure and dynamics, Physics Reports 424 (2006) 175–308.

[10] L. Panait, S. Luke, Cooperative multi-agent learning: The state of the art, Autonomous Agents and Multi-Agent Systems 11 (3) (2005) 387–434.

[11] M. Bowling, Convergence and no-regret in multiagent learning, in: Proceedings of the Annual Conference on Advances in Neural Information Processing Systems, 2005, pp. 209–216.

[12] R. Sutton, A. Barto, Reinforcement Learning: An Introduction, MIT Press, 1999.

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

[14] V. Conitzer, T. Sandholm, AWESOME: A general multiagent learning algorithm that converges in self-play and learns a best response against stationary opponents, Machine Learning 67 (1-2) (2007) 23–43.

[15] Y. Shoham, R. Powers, T. Grenager, If multi-agent learning is the answer, what is the question?, Artificial Intelligence 171 (7) (2007) 365–377.

[16] B. Banerjee, J. Peng, Generalized multiagent learning with performance bound, Autonomous Agents and Multiagent Systems 15 (3) (2007) 281–312.

[17] S. Abdallah, V. Lesser, Learning the task allocation game, in: Proceedings of the International Joint Conference on Autonomous Agents and Multiagent Systems, 2006, pp. 850–857.

[18] A. Clauset, C. Rohilla Shalizi, M. E. J. Newman, Power-law distributions in empirical data, ArXiv e-printsarXiv:0706.1062.

[19] M. Bowling, Convergence and no-regret in multiagent learning, Technical Report TR04-11, University of Alberta (2004).

[20] P. Vrancx, K. Tuyls, R. Westra, Switching dynamics of multi-agent learning, in: Proceedings of the 7th International Joint conference on Autonomous Agents and Multiagent Systems, 2008, pp. 307–313.

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

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

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

[24] V. Paxson, End-to-end routing behavior in the internet, SIGCOMM Computer Communication Review 36 (5) (2006) 41–56.

[25] M. McGlohon, L. Akoglu, C. Faloutsos, Weighted graphs and disconnected components: patterns and a generator, in: ACM SIGKDD Conference on Knowledge Discovery and Data Mining, 2008, pp. 524–532.

[26] E. Almaas, B. Kovacs, T. Vicsek, Z. N. Oltvai, A. L. Barabasi, Global organization of metabolic fluxes in the bacterium, escherichia coli, Nature 427 (2004) 839.

[27] S. E. Ahnert, D. Garlaschelli, T. M. A. Fink, G. Caldarelli, Ensemble approach to the analysis of weighted networks, Physics Review E 76 (1) (2007) 016101.