A Visualization Technique for Multiagent Systems

Manal Rayes

Master of Science in Information Technology
Faculty of Informatics
The British University in Dubai
March 2011

Abstract

In this dissertation we consider the problem of monitoring and visualizing the performance of multi-agent systems, i.e. systems of adaptive agents that change their behaviour as a result of learning and experiencing. Instead of relying on global performance measures, as have been done by current visualization techniques, we use a different technique which is a combination of dimensionality reduction and social network measures. The advantage of this technique, we claim, is that it does not only capture the performance of the multiagent system on its macro level, as is the case with the global performance metrics methods, but it can also capture the performance on the micro level, i.e. by being sensitive to the performance of individual agents.

To test our technique, we first conduct several experiments to compare different combinations of dimensionality reduction techniques and network measures. Then we apply one such combination on networks of adaptive agents that play games (we use the term "game" as in the Game theory).

Our findings confirm that using dimensionality-reduced weighted network measures in visualizing the performance of multi-agent systems is informative in the sense that they proved to be sensitive to changes in the global as well as local system dynamics (for example the global network structure and the local learning of individual agents).

Acknowledgement

First to my parents who have always been there for us, and to my sister and brother. To my father who has encouraged us to constantly seek higher standards, pursue high goals, and face any challenges with faith and hard work; and to my mother for her unconditional love, prayers, and support, especially during this work when I most needed her for taking care of my baby.

To my husband, thank you for the valuable support and encouragement.

To my supervisor for his guidance (despite the many other work overloads).

Lastly, to all friends who wished me good luck.

Declaration

I declare that this thesis was composed by myself, that the work contained herein is my own except where explicitly stated otherwise in the text, and that this work has not been submitted for any other degree or professional qualification except as specified.

(*Manal Rayes*)

Contents

List of Figures

Chapter 1

Introduction

Multiagent systems (MAS) are systems composed of multiple interacting intelligent agents, where an intelligent agent is a computational element that is capable of interacting with its environment (as well as with other agents) and taking intelligent actions. The most important aspects of an intelligent agent are that its rationality, in the sense that it performs the action(s) that would result in the outcome that best serves its goal, and its capability of learning or using its knowledge in choosing such actions.

Although the subject of multiagent systems has its roots in Game theory (which analyses situations where the action of one agent depends on the others' choices) and distributed artificial intelligence (which studies the development of distributed solutions for complex problems that require intelligence), multiagent systems have been recognized as a field in its own right since about 1980, and has gained widespread interest in the mid 1990s. This acknowledgement of the individuality of the field was largely driven by having the researchers recognizing that placing multiple intelligent agents in one collective system is a non-trivial problem that has its own considerations and issues. These difficulties stem from the fact that MAS involves two levels: the micro level, with individual agents each with its local goals, knowledge, and learning, and the macro level, in that all the elements of the system have to achieve one desired global behaviour. This particular property of MAS gives rise to an important challenge (the one we are addressing in this study): the difficulty of visualizing the performance and analysing the dynamics of multiagent systems on both levels: the micro level (that of the individual agents each with its own set of local parameters) and the macro level (that of the whole system).

While traditional techniques have relied heavily on using global performance metrics the system is trying to optimize, such as the social welfare, or other summarizing statistics of the local performance parameters such as the average number of wins, in visualizing the performance of multiagent systems, but these techniques can overlook important information pertinent to the performance on the micro level such as the malfunction of some agent (in its behaviour due to a disruption in its learning functionality for example). Moreover, experiments have shown that depending on the global performance alone can not reveal hidden instability [1].

Thus, we define the problem we are tackling in this research as follows:

to find a technique for visualizing the performance of multiagent systems in a way that is capable of summarizing the global performance of the whole system (on its macro level) by as few parameters as possible, while being able to remain sensitive to the dynamics of the individual agents (the micro level).

In our search for a solution, we were inspired by two main subjects: a) dimensionality reduction which is defined as the process of reducing the number of parameters of a dataset consisting of a large number of parameters into a dataset with fewer parameters while retaining as much as possible of the features (i.e. the variation in parameters); and b) organizing agents in a network and looking at their interaction through the spectacle of social network analysis, where a number of metrics have been devised to summarize characteristics pertinent to the

local centrality of individual nodes as well as the global structure of the network.

We relate the first of these two subjects (the dimensionality reduction) to the first criteria we are looking for in the visualization technique (that is of being able to summarize the macro level of the system), and the second subject (social network analysis) to the second criteria (that is of being able to embed the system performance on the micro level). Also we relate the second subject to a recent work [1] that suggests the use of graph analysis to study networks of adaptive agents and, moreover, extends one most-commonly used social network metric (the node degree centrality) into a new measure (called the C-degree) that accounts for the disparity in interaction between a node and its neighbours.

We combine these two subjects, namely the dimensionality reduction and the social network analysis, into one technique, the dimensionality-reduced network metrics, for visualizing the performance of multiagent systems in a manner that would preserve characteristics of the local as well as the global level.

To achieve this, we outline two main tasks to be achieved:

Task-1 carry out a comparative study of different combinations of dimensionality reduction techniques and of social network metrics in order to come up with one combination that would achieve good visualization results, and,
Task-2 test this dimensionality-reduced network metrics combination on networks of adaptive agents.

Through the first task, of conducting a comparative study, we aim at finding a combination that is capable of demonstrating several properties, which we are going to evaluate later in our second task. These desired properties are addressed in the following research questions.

RQ1. Can this visualization technique differentiate between different macro-level settings of multiagent systems by being able to distinguish between different learning algorithms applied to the system?
RQ2. Can this visualization technique capture the performance of nodes (the micro-level) by being sensitive to disruptions in the learning of agents as well as in the network structure?
RQ3. Can this visualization technique explain the performance of the system, for example by identifying the type and source of any disruption or malfunctioning in the system?

In this thesis we show how we approach the problem of visualizing multiagent systems (as defined above), by achieving both tasks (of conducting initial experiments to find a suitable combination and then by applying this combination on a network of learning agents). We discuss how the experimental findings succeed at positively answering the first two research questions, while not being able to provide a clear answer to the last question, hence it remains as a future work.

The remaining of this thesis in organized as follows. Chapter 2 provides the necessary background. In it we introduce the multiagent problem formulation, a brief survey of the most common multiagent learning approaches, as well as existing visualization techniques for multiagent systems. Chapter 3 describes our methodology we follow for testing our visualization method by describing the techniques we are employing, namely dimensionality reduction and the social network measures, the tools we are using, and the experimental settings we are applying. Chapter 4 demonstrates the results of the various experiments we conducted in attempt to answer the research questions we posed above, and discusses these results. We conclude in Chapter 5 by summarizing the work done and suggesting future work.

Chapter 2

Background

2.1 Multiagent Problem Formulation

"The goal of multiagent systems' research is to find methods that allow us to build complex systems composed of autonomous agents who, while operating on local knowledge and possessing only limited abilities, are capable of enacting the desired global behaviors" [22].

To provide some intuition about the problem of multiagent learning, consider the game described by the payoff matrix below.

	Left	Right
Up	1,0	3,2
Down	2,1	4,0

The row player would get a higher payoff by playing *Down* whether the column player plays *Left* or *Right*, hence, *Down* is a strictly dominant strategy. If, however, we assume that this game is played repeatedly with the row player consistently playing *Down*. After a while, the column player will respond by playing *Left*. By repeatedly playing its dominant strategy, the row player had caused the column player to *adapt*.

2.1.1 Preliminary Concepts

Agents are typically modelled as utility maximizers who inhabit some kind of Markov decision process and whose actions can affect each other's utilities. A **utility** function can be defined as follows given that S is the set of states in the world.

$$u_i : S \rightarrow \Re$$

In a non-deterministic environment the **expected utility** is computed instead based on the probability of reaching the next state $\acute{s}$, given state s and action a. This probability is given by a transition function $T(s,a,\acute{s})$ and the expected utility is computed as follows.

$$E[u_i, s, a] = \Sigma_{\acute{s} \in S} T(s, a, \acute{s}) u_i(\acute{s})$$

A **policy** (π) is a mapping from states to actions. The agent's goal thus becomes that of finding the **optimal policy**, i.e. the policy that maximizes its expected utility. Optimal policy can thus be defined as

$$\pi_i^*(s) = argmax_{a \in A} E[u_i, s, a]$$

.

The idea that an agent inhabits an environment, takes an action that changes the state of the environment (the state of the environment can also change due to some external event), and receives a reward accordingly, is captured by a **Marcov decision process (MDP)**.

Definition 1. *(Marcov Decision Process). An MDP consists of an initial state $s_1 \in S$ (set of states), a transition function T(s,a,s$'$) and a reward function r: $S \rightarrow \Re$.*

2.1.2 Game Theory

Game theory is the branch of Mathematics for analyzing strategic decisions amongst multiple players, where the actions of one party impact all other parties. Game theory was first introduced in the book "*The Theory of Games and Economic Behavior*" by von Neumann and Morgenstern [23] and gained prominence with the work of John Nash [10].

In this dissertation we consider games in *normal form* (also known as standard or strategic form and it is the simplest and most familiar form of games). In the simplest type of these games where we have two players each of which must take one of two actions and receive a payoff (here we use the words payoff, utility, and reward interchangeably) based on their joint actions, the game can be represented as an n-dimensional payoff matrix.

A sample payoff matrix for two players is shown in the the following payoff matrix. Each player can take one of two actions and receive a payoff accordingly. The first value in any cell (i, j) is the payoff of the row player (r_i), whereas the second value is the payoff of the column player (c_j), when the row player takes action i and the column player takes action j simultaneously.

	c	d
a	1,2	4,3
b	3,2	2,4

In normal-form games we always assume that the players play simultaneous actions. Another assumption is that the players have common knowledge of the utilities that all players can receive.

When an agent deterministically take a specific action, then it is playing a *pure strategy*. In contrast, if the player takes different actions according to a probability distribution σ then it is playing a *mixed strategy*.

A game with non-deterministic actions falls under the general class of *stochastic games* where the non-deterministic property is captured with probabilistic transitions. In stochastic games, a game is played in a sequence of stages. In each stage when all agents take actions, the game moves to a new random state whose distribution depends on the previous state and the actions taken by the agents. A special case of stochastic games that concern us in this study is the *repeated game* which is basically a stochastic game with only one stage game.

Stochastic games share some considerations that need to be specified a priori. For example, we need to specify whether the agents can observe the stage games, the actions available to them, and the transition probabilities. The simplest setting is to make the game fully observable. Another consideration is whether agents know about the other agents, their actions, strategies, or the rewards they receive after the game has been played. In many models, agents can learn about other agents' strategies so that they can devise a best response.

The most important question about games, and the one that flows naturally and receives most consideration from researchers of the field, is how to reason about such games. While the problem can be reduced in single-agent settings to that of finding the optimal strategy (i.e. the strategy that is strictly better than all other possible strategies) which is equivalent to choosing a utility-maximizing action in multi-agent settings, but in multiagent settings it is more complex and even not meaningful. This is mainly because of the presence of multiple agents all of whom are trying to maximize their payoffs, and each of whom interests are equally

as important as every one else's interests. Instead, game theorists have identified a number of "interesting" *solutions concepts*, where a solution concept is a subset of outcomes that satisfies certain conditions of acceptability.

Probably the simplest solution concept is the *maximum social welfare* which prefers the strategy profile that maximizes the sum of individual rewards. A more sophisticated solution concept is the *Pareto optimality*. While this solution concept does not answer the question of identifying a single "best" strategy, but it can identify a strategy that is *better* than another. It does so by preferring the strategy that can make some player better off without making any other player worse off.

Another solution concept is the *Nash equilibrium* which is considered the most influential solution concept in game theory. This solution concept identifies the "most interesting" strategy profile to be the one in which every agent is playing its best response to every other agent. This results in the stability of the system, since no agent would want to unilaterally change his strategy. Moreover, Nash proved that all game matrices have at least one equilibrium strategy, although this strategy might be mixed (i.e with probability assigned to each action).

According to the type of Nash equilibrium in the payoff matrix, we can subdivide the normal-form games into three categories [21]: (a) games with one pure equilibrium (one unique solution), (b) games with one mixed equilibrium and (c) games with two pure equilibria and one mixed equilibrium. The following three paragraphs present one example of each category.

The *Prisoner Dilemma* is the most classical example of the normal-form games of type (a). In this game each player may cooperate (C) of defect (D). The payoff matrix of this game is given below.

	C	D
C	3,3	0,5
D	5,0	1,1

The unique solution is (D,D) and it also satisfies the Pareto optimality (since cooperating is strictly dominated by defecting).

Matching Pennies is an example of category (b) and is represented by the following payoff matrix:

	H	T
H	1,-1	-1,1
T	-1,1	1,-1

Each player must choose among two actions H (for head) and T (for tail). There is no strategy in this game that is said to be a best response to a best response, thus it has no pure Nash equilibrium. Instead, the unique Nash equilibrium of this game is the mixed strategy reached by having each player plays head and tail with equal probabilities.

The *Battle of the Sexes* is the third example representing games of category (c) and is a sample of a coordination game. In this game each player has to choose among two actions F (for football) ,and I (for Ice-Hockey) and while each of the two players has its own preferences (the row player prefers F, whereas the column player prefers I), but both players like to be with the other, i.e. by choosing the same action. The game can be defined by the following payoff matrix:

	I	F
I	4,7	0,0
F	3,3	7,4

The pure equilibria is given by the two pairs of (F,F) and (I,I), and the mixed equilibrium is defined by the strategy $(\frac{2}{3}, \frac{1}{3})$ and $(\frac{1}{3}, \frac{2}{3})$ for the row player and column player respectively.

An example of a game with three actions is the *climbing game* (as shown in the payoff matrix below), which is an example of symmetric games, that are games with equal payoff for both players.

$$\begin{bmatrix} 11 & -10 & 0 \\ -10 & 7 & 6 \\ 0 & 0 & 5 \end{bmatrix}$$

The previously introduced games were originally devised for two players. To study systems with multiple players, symmetric games with n players and n actions (where n can be any finite integer) have been suggested. An example of such games is the *dispersion game* (also called the anti-coordination game). In this game each player i of the n players selects an action a_i simultaneously from an action set of size n. If all players guess different actions (i.e. a sole action is picked by each player) then the reward will be maximal; if a player chooses a duplicate action his reward will be minimal. To represent this game a payoff function is used, since it is more convenient than a payoff matrix, as follows. First the number of players selecting action j is defined as

$$S(j) = \sum_{i=1}^{n} id(a_i, j)$$

where

$$id(i,j) = \begin{cases} 1 & if i = j \\ 0 & Otherwise \end{cases}$$

The payoff function is defined as

$$r_i = \begin{cases} 1 & if S(a_i) = 1 \\ 0 & Otherwise \end{cases}$$

2.1.3 Formal Definitions

In this subsection we introduce the formal definitions of the major concepts in Game Theory for normal-form games and solution concepts [15].

Definition 2. *(Normal-form game) A finite, n-person normal-form game is a tuple (N, A, u), where:*

- *N is a finite set of n players, indexed by i;*
- *$A = A_1 \times .... \times A_n$, where A_i is a finite set of actions available to player i. Each vector $a = (a_1,...,a_n) \in A$ is called an action profile;*
- *$u = (u_1,...,u_n)$, where u_i: $A \mapsto \Re$ is the payoff of player i.*

Definition 3. *(Stochastic game) A stochastic game can be represented as a tuple: (N, S, $\overrightarrow{A}$, $\overrightarrow{R}$, T), where:*

- *N is a set of agents indexed 1,...,n;*
- *S is a set of n-agent stage games;*
- *$\overrightarrow{A} = A_1,..., A_n$, where A_i is the set of actions of agent i;*
- *$\overrightarrow{R} = R_1,..., R_n$, with R_i: $S \times \overrightarrow{A} \rightarrow \Re$ is the reward function of agent i for stage game S;*
- *T: $S \times \overrightarrow{A} \rightarrow \prod(S)$ is a stochastic transition function that represents the probability of getting into the next stage given the current stage and the actions played in it.*

Definition 4. *(Mixed strategy) Let (N, A, u) be a normal-form game, and let σ be the set of all probability distributions. Then the set of mixed strategies for player i is $S_i = \sigma(A_i)$.*

Definition 5. *(Mixed strategy profile) The set of mixed-strategy profiles for n agents is $S_1 \times \ldots \times S_n$.*

Definition 6. *(Maximum social welfare profile) Let the maximum social welfare be $max_\pi \omega(\pi) = max_\pi \Sigma_{i=1}^{n} R^i(\pi)$, the strategy profile π^* is $\pi^* = argmax_\pi \omega(\pi)$*

Definition 7. *(Pareto domination) Strategy profile s Pareto-dominates strategy profile ś if for all $i \in N$, $u_i(s) \geq u_i(ś)$, and there exists some $j \in N$, for which $u_j(s) > u_j(ś)$*

Definition 8. *(Pareto optimality) Strategy profile s is Pareto optimal if there does not exist another strategy profile $ś \in S$ that Pareto dominates s.*

Definition 9. *(Best response) Player i's best response to the strategy profile s_{-i} is a mixed strategy $s_i^* \in S_i$ such that $u_i(s_i^*, s_{-i}) \geq u_i(s_i, s_{-i})$ for all strategies $s_i \in S_i$.*

Definition 10. *(Nash equilibrium) A strategy profile $s = (s_1, ..., s_n)$ is a Nash equilibrium if, for all agents i, s_i is a best response to s_{-i}.*

2.2 Multiagent Learning Approaches

Algorithms for multiagent learning in repeated games can be classified under broader classes based on different criteria. One such criteria is whether the technique explicitly considers other agents' strategies for its own learning, as in the model-based approaches, or just learns using the rewards received by the possible actions as in the model-free approaches.

In the following couple subsections we briefly introduce each category.

2.2.1 Model-based approaches

The algorithms which fall under this category aim at learning explicit models of the other players, assuming that they adhere to a stationary policy so that the player can then play its best response. Typically, in these algorithms learning occurs in the policy space, hence they are sometimes also called policy-based algorithms.

While this type of algorithms is rational (guarantees best-response), but it offers no guarantee of convergence. This is because it only plays pure policies, and hence cannot converge in games with non-pure equilibria.

The abstract scheme of model-based algorithms is as follows.

1. Start with some model of the opponent's strategy.
2. Compute and play the best response.
3. Observe the opponent's play and update your model of its strategy.
4. Goto step 2.

The most common and earliest instance of this scheme is *fictitious play*. Assuming that the opponent is playing a fixed strategy, the agent builds a model of its opponent's strategy based on its past experience (i.e. previous plays) and uses this model to choose its best response. The computed model is simply the probability distribution for the opponent's expected strategy. For repeated games, when it is assumed that all players are using fictitious play, it has been shown that fictitious play converges to Nash for certain types of games (those are games that are iterated dominance solvable).

Extensions to the fictitious play in a reinforcement learning context include the Opponent Modelling or Joint-Action Learners (JALs). In opponent modelling Q-learning algorithm, the action selected in the learning step for some state is the action that maximizes Q for that state multiplied by the estimate the other players will select that joint action based on their previous plays.

Another well-known algorithm is the *gradient ascent* [16]. This algorithm deals with the multiagent problem as an optimization problem that aims at finding the strategy that maximizes its expected reward, where in the setting of a two-action game, a strategy (i.e. a probability distribution over the two possible actions) can be represented as a single numerical value.

After each iteration the player will adjust its strategy so as to increase its expected payoff by moving their strategy in the direction of the current gradient with some small step size η. The gradient represents the effect of changing the current strategy on the expected payoff. It is computed by finding the partial derivative of the expected payoff with respect to the current strategy. The details of these computations are as follows.

Let $\alpha \in [0, 1]$ be a strategy that corresponds to the probability of the row player selecting the first action and $1 - \alpha$ be the probability the row player selecting the second action. Similarly, let β be a strategy for the column player. Let $V_r(\alpha, \beta)$ be the expected payoff the row player will receive. Then,

$V_r(\alpha, \beta) = \alpha\beta r_{11} + \alpha(1 - \beta)r_{12} + (1 - \alpha)\beta r_{21} + (1 - \alpha)(1 - \beta)r_{22}$

$= u\alpha\beta + \alpha(r_{12} - r_{22}) + \beta(r_{21} - r_{22}) + r_{22}$

where,

$u = r_{11} - r_{12} - r_{21} + r22.$

Then, the player can compute partial derivative of its expected payoff with respect to its strategy as follows.

$\frac{\partial V_r(\alpha, \beta)}{\partial \alpha} = \beta u + (r_{12} - r_{22}).$

The new strategy will then be,

$\alpha_{k+1} = \alpha_k + \eta \frac{\partial V_r(\alpha, \beta)}{\partial \alpha}.$

In a similar manner the strategy of the column player, β, can be computed.

In the case of an infinitesimal step size, the algorithm is called *Infinitesimal Gradient Ascent (IGA)*. It has been shown that if both players follow IGA with an appropriately decreasing step size, then their strategies will converge to a Nash equilibrium or the average payoffs over time will converge in the limit to the expected payoffs of a Nash equilibrium [16].

2.2.2 Model-free approaches

Another class of techniques that stems mainly from the recent works in AI, mostly under the general heading of *reinforcement learning* [17], is the model-free class. In this approach the agent is not concerned with learning the opponent's strategy. Instead, it tries to learn overtime about the returns of its own possible actions. Consequently, it can make a choice about the action that will maximize its expected utility. This is formally captured by Bellman's equation:

$$u(s) = r(s) + \gamma max_a \Sigma T(s, a, \acute{s})u(\acute{s})$$

This equation is considered the root from which algorithms that belong to this approach stem. The most common algorithm in this category is the *Q-learning* [24] which is defined, in its simplest form, by

$$Q(s, a) \leftarrow (1 - \gamma_t)Q(s, a) + \alpha_t(r(s, a) + \gamma max_{\acute{a}} Q(\acute{s}, \acute{a}))$$

.

To extend the Q-learning algorithm to the multi-agent stochastic game each agent can assume that the environment is passive by updating the Q values without regard for the other agent's strategies. To implement this the Q-values is defined as a function of all the agents' actions (in the equation of Q above, a is replaced by the vector $\overrightarrow{a}$).

2.2.3 Metrics for Learning Algorithms

The performance of learning algorithms for agents are evaluated in terms of convergence (does it converge?) and rationality (does it best serve the interest of the agent?). Various metrics have been suggested as numerical measures to help making quantitative evaluation and comparisons between algorithms.

While it is possible in single agent settings to measure convergence to the optimal strategy (along with other metrics related to optimality) [7], in multi-agent settings alternative metrics are needed due to the difficulties in defining what an optimal solution would be.

Convergence to Nash equilibrium is considered the multi-agent version of convergence to optimal solution. Other metrics are proposed to evaluate the rationality considered in multi-agent learning systems such as: *reward maximization, number of wins, regret, and incentive to deviate* (to any mixed strategy). *Regret* is computed by finding the difference between the reward obtained with the agent's current strategy and the reward that could have been obtained under the best pure strategy [3]. *Incentive to deviate* compares the current stage play to what would have been the best response action.

2.3 Existing Testing and Visualizing Methodologies

In [20] four methods for visualizing the performance and understanding the learning dynamics of multiagent learning algorithms have been listed: a) policy trajectory plot, b) directional field plot, c) percentage-wise convergence table with confidence intervals, and d) cumulative reward plot.

The first two methods are visual means to plot the policy space. These two methods can be used to plot games in 2D (Figures 2.1 and 2.3) or 3D (Figure 2.2). However, to better study the performance in higher dimensional games, the results can be listed in a table that show the convergence percentage with confidence intervals. The last method (cumulative reward plot) is used to plot some descriptive statistics such as the cumulative or average reward or the social welfare (Figure 2.4).

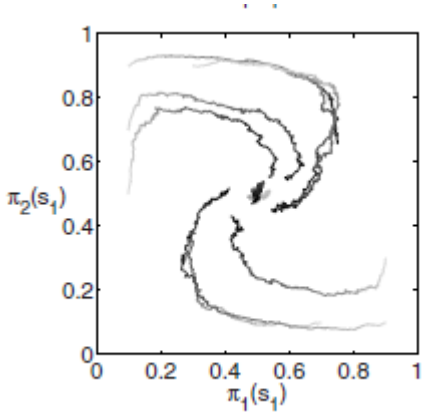

Figure 2.1: A sample of a trajectory plot for a Q-learner. Grayscales are used to indicate the direction of convergence [20].

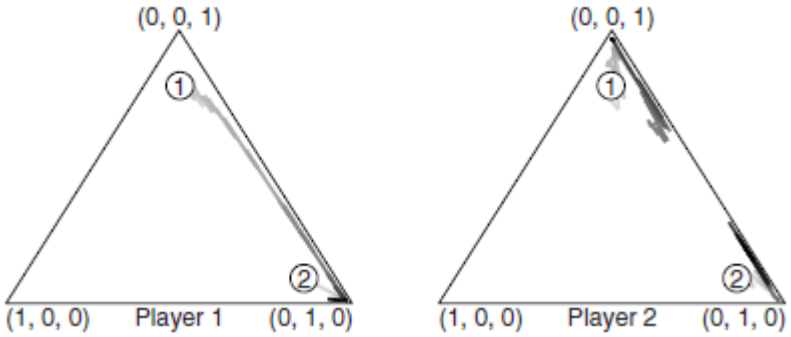

Figure 2.2: A sample of a simplex plot for the climbing game (with 3×3 game matrix) [20].

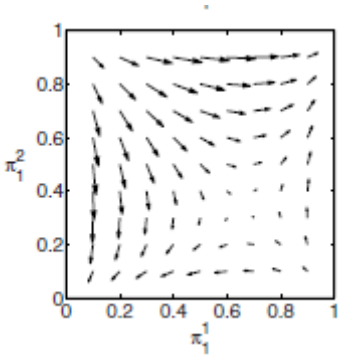

Figure 2.3: A sample of a directional field plot (FALA learning algorithm in the Battle of the Sexes game) [20].

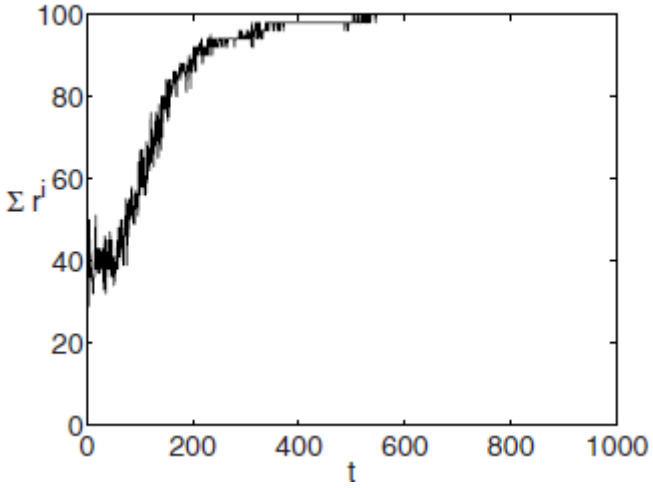

Figure 2.4: A sample of a cumulative reward plot (FALA learning algorithm in the dispersion game) [20].

Chapter 3

Methodology

In this dissertation we propose the use of dimensionality-reduced network metrics as a visualization technique for multiagent systems. In the Introduction we outlined two tasks to be achieved for that purpose. These tasks were: first, to test several combinations of dimensionality reduction techniques and social network metrics; and, second, to test the resulting combination on networks of adaptive agents.

In this chapter, we delve into these two tasks by first detailing the proposed techniques, namely the dimensionality reduction and the social network metrics. We then introduce the tools we used for applying these techniques and describe the settings we applied prior to conducting the experiments.

3.1 Techniques

In this section we provide details on the two techniques we are proposing for visualizing the performance of multiagent systems. We introduce dimensionality reduction by first defining it and then by providing a summary of its main techniques; the techniques we made use of in this research. We then move to social network metrics.

3.1.1 Dimensionality Reduction Techniques

Dimensionality reduction is the process of reducing the number of features or parameters of a data set consisting of a large number of interrelated variables, called latent variables, while retaining as much as possible of the variation. It is a key tool for analysing high-dimensional data. The number of latent variables is referred to as *intrinsic dimensions* or simply number of dimensions. For a random vector y, intrinsic dimension is defined as the minimal number of parameters needed to describe y [8].

The most common linear dimensionality reduction methods are the Principal Component Analysis (PCA) and the Multidimensional Scaling (MDS). Non-linear dimensionality reduction algorithms can be divided into two classes [8]: a) distance-preserving methods, such as Isomap and Semidefinite Embedding (SDE), and b) topology preserving methods, such as Locally Linear Embedding (LLE) and Isotop. The remainder of this section summarizes three of these algorithms (namely PCA, MDS, and Isomap) since they are used in this study.

Principal Component Analysis (PCA)

PCA is one of the oldest, simplest, and most commonly used mathematical techniques for reducing dimensionality in a large set of observations [14]. PCA works by transforming the

original data set into a new set of uncorrelated variables (PCs) which are ordered so that the first few retain most of the variation in all of the original variables [6].

Given x, a vector of p random variables (latent variables), PCA first looks for a linear function $\alpha_1 x$ of the elements of x having maximum variance, where α_1 is a vector of p constants $(\alpha_{11}, \alpha_{12}, .., \alpha_{1p})$. It then looks for a linear function $\alpha_2 x$ uncorrelated with $\alpha_1 x$ having maximum variance, and so on until p PCs could be found. The complexity is reduced by having the original variables transformed into p PCs. The linear function α_i is the eigenvector of the covariance matrix corresponding to its i^{th} largest eigenvalue.

Multidimensional Scaling (MDS)

MDS tries to find a set of vectors in p-dimensional space such that Euclidean distances among them corresponds as closely as possible to some function of the input matrix according to a criterion called stress [19]. It tries to preserve pairwise distances that are measured along straight lines.

The algorithm works as follows:

1. Arbitrarily assign points to coordinates in p-dimension space.
2. Compute Euclidean distances in $D^{'}$ matrix.
3. Compare $D^{'}$ matrix with the input D matrix and evaluate the stress function.
4. Adjust coordinates of each point in the direction that best maximizes stress.
5. Repeat 2-4 until stress wont get any better.

Isomap

Isomap extends MDS by incorporating geodesic distances imposed by weighted graphs [18]. It tries to preserve pairwise distances that are measured along shortest paths.

The steps taken by the algorithm are:

1. Determine a neighbourhood graph G.
2. Compute shortest paths in the graph for all pairs of data points. Each edge in the graph is weighted by its Euclidean length.
3. Apply MDS to the resulting shortest path matrix.

3.1.2 Social Network Metrics

Network measures are functions that summarize a graph into numeric values to simplify the analysis of the network. A key network metric is the centrality of a node which is concerned with measuring the extent to which a node is "central" in the network. A node is said to be more central than others if: it has more ties, it can reach all others more quickly, or it controls the flow between the others. These three properties form the three measures of node centrality: degree, closeness, and betweenness [5].

The degree of a node is the number of edges adjacent to that node. If the network is directed, we can differentiate in-degree (i.e. number of incoming links) and out-degree (i.e. number of outgoing links). Closeness is the degree an individual node is near all others. It is defined as the inverse of the sum of the shortest distances to all other nodes from this focal node. For this, closeness can not be applied if the network has disconnected components. Betweenness is the extent to which a node lies on the shortest path between each other couple of nodes. Degree is

a feature of the local structure of the network, while betweenness and closeness considers the global structure.

These measures of centrality have been devised originally for binary networks, i.e. networks where links have no numerical values (a link can either exist or not), however they were generalized by later works for weighted networks [2, 11, 13]. Degree was extended to weighted networks by defining it as the sum of weights of the adjacent links (of a focal node) [2]. This is also called the strength of the node. For computing closeness and betweenness the Dijkstra's shortest path was used since it takes into account the cost of the path [4].

To formally describe the different measures, we list the different notations and definitions as follows.

Given a network $N = \langle V, E \rangle$, where V is the set of nodes and E is the set of edges, 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.

The node's strength is defined as:

$$s_i = \sum_{j}^{N} w_{ij}$$

, where w is the adjacency matrix with values representing the weights of the links (note that in binary networks the adjacency matrix values can either be 0 or 1).

In [13] the number of ties, i.e. the degree, is also taken into consideration (besides the weights of the links) in computing weighted centrality. The relative importance between the number of links and their weights can be tuned by a variable parameter α. Thus, the weighted degree combines both the degree and the strength of the node as in the following formula.

$$C_D^{w\alpha} = k_i \times (\frac{s_i}{k_i})_\alpha = k_i^{(1-\alpha)} \times s_i^\alpha$$

where α is a positive parameter between 0 and 1. When α is 1 the measure's value equals the node's strength, and when it is set to 1 it gives the degree (as in binary networks).

The length of the shortest path, required for computing the closeness and betweenness, is found in binary networks by finding the minimum number of ties linking two nodes. Formally it is defined by

$$l = d(i, j) = min(x_{ih} + ... + x_{hj})$$

, where h corresponds to intermediary nodes on the linking path between nodes i and j. Dijkstra treated the weights as costs of transmitting [4]. To extend the computation of distance in weighted networks, the weights are inverted since in social networks the weights are operationalizations of strength of communications not the cost of them. Thus,

$$d^w(i, j) = min(\frac{1}{w_{ih}} + ... + \frac{1}{w_{hj}})$$

.

Closeness and betweenness are formally defined as

$$C_C(i) = [\sum_{j=1}^{N} d(i, j)]^{-1}$$

$$C_B(i) = \frac{g_{jk}(i)}{g_{jk}}$$

respectively, where g_{jk} is the number of shortest paths between nodes j and k, and $k_{jk}(i)$ is the number of paths passing through node i from j to k.

In [13] Dijkstra's shortest path algorithm is extended to include the number of intermediary

nodes. Formally, the length of the shortest path is defined as:

$$d^{w\alpha}(i,j) = min(\frac{1}{(w_{ih})^{\alpha}} + ... + \frac{1}{(w_{hj})^{\alpha}})$$

where α is a positive tuning parameter. Thus, closeness and betweenness are redefined in terms of this parametrized weighted distance function as follows:

$$C_C^{w\alpha}(i) = [\sum_{j=1}^{N} d^{w\alpha}(i,j)]^{-1}$$

$$C_B^{w\alpha}(i) = \frac{g_{jk}^{w\alpha}(i)}{g_{jk}^{w\alpha}}$$

.

Another measure, that is of importance to our work, is the *continuous degree*, of the *C-degree* [1]. This measure is formally defined as:

$$r(v) = 2^{(\sum_{e \in E(v)} \frac{w(e)}{s(v)} log_2 \frac{s(v)}{w(e)})}$$

. If node v is disconnected, then $r(v) = 0$. This measure uses the entropy of the interaction probability distribution, or how many bits are required to encode the interaction probability distribution, to quantify the disparity in the interaction in contrast to the traditional degree measure that assumes uniform interaction across the node's neighbors.

3.2 Tools

In order for applying the techniques we are making use of for our visualization method, we made use of several available tools. The tools we have used are: *MATLAB dimensionality reduction toolbox* for applying different dimensionality reduction techniques [9], *tnet* [12] for computing the weighted network measures, and *NetLogo* [25] for building a platform for testing the proposed visualization technique on networks of adaptive agents.

The following subsections briefly describe each of these tools.

3.2.1 MATLAB Dimensionality Reduction Toolbox: a Tool for Dimensionality Reduction

MATLAB is a high-level language and interactive environment for technical computing. Besides the built in tools for numerical, algebraic, and statistical analysis, it also provides many add-on toolboxes as extensions to solve particular classes of problems in different areas. In this research, we make use of the Matlab add-on toolbox of dimensionality reduction [9]. At the time of writing this thesis, this toolbox contained Matlab implementations of 33 techniques of dimensionality reduction. In addition, it also contains implementations of 6 techniques (as of the time of writing this thesis) for intrinsic dimensionality estimation. Access to any of the DR techniques can be done through the same command:

```
mapped_ data = compute_ mapping(data, method, no.  of dimensions, parameters).
```

In this thesis we use Matlab for dimensionality reduction using the following techniques: PCA, Isomap, and MDS (for MDS we use two implementations MDS-fast, and MDScale which is a built-in function in Matlab). In addition we use it for any additional computational task such as importing the data or performing any pre-processing on it. We also use it for coding and testing the C-degree measure (as defined in [1]).

3.2.2 tnet: a Tool for Computing Network Measures on Weighted Networks

tnet is a package written in *R*, a free software environment for statistical computing and graphics, for calculating weighted social network measures. The currently implemented network measures in *tnet* are centrality measures (degree, closeness, and betweenness), clustering coefficient, and the weighted rich-club effect framework.

In this thesis we use the following weighted measures: degree, betweenness, closeness node centralities as have been defined in [13] and implemented in tnet [12].

3.2.3 NetLogo: a Tool for Modeling Multiagent Systems

NetLogo is a multiagent programmable modeling environment. It uses a simple multiagent programming language (the Logo dialect that is extended to support agents) with a visual interface builder that makes it easy to create complex systems by programming mobile agents and specifying the parameters of the environment.

In this thesis we use NetLogo for the sake of running different learning algorithms on different networks of adaptive agents. For this sake, we developed a platform in which agents engage in playing some game and use their payoffs in learning and choosing their actions overtime.

In this platform, agents are situated in a network, with pre-specified number of nodes and average node degree, where each node represents an agent. Each tie links two players that (may) engage in playing the selected game. The network is weighted in a manner similar to weighted social networks where weights typically represent the amount of communication between the corresponding nodes. Moreover, the links are directed. Thus, a weight w_{ij} of a link that goes from node i to node j represents the valuation of node i to the communication with node j (based on the Q value the source node associates with the target node according to the payoff it receives via playing with it). The output of each run is the edge-list corresponding to the evolving weighted network.

A snapshot of the platform is illustrated in Figure 3.1.

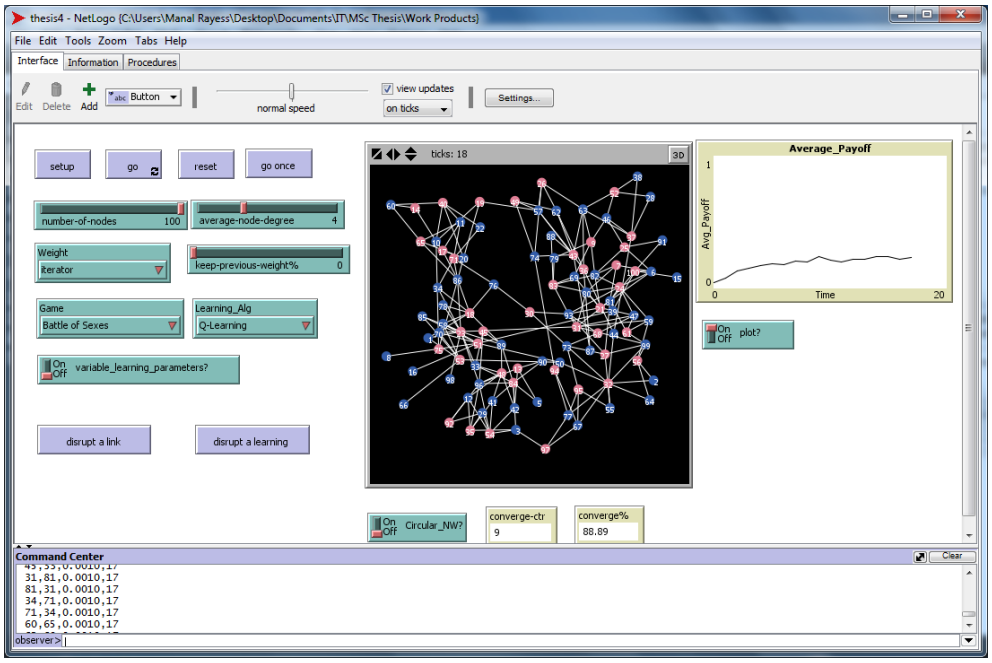

Figure 3.1: Platform developed with NetLogo for running learning algorithms on adaptive agents that are organized in a network and engage in repeatedly playing games.

3.3 Experimental Settings

In this section we detail the settings of the experiments we need to conduct for the accomplishment of tasks 1 and 2.

For the first task we have to compare different combinations of different dimensionality reduction techniques and different social network measures. For this, we conduct a couple of experiments. In the first experiment we test different dimensionality reduction techniques on a couple of existing datasets each of which correspond to some learning algorithm. The purpose of the second experiment is to identify the effect of different social network measures of the visualization of some evolving network. For this we use an existing dataset of a static network and then apply three different evolution mechanisms. Then we conduct a third set of experiments to test different values for the α parameter of the social network measures. We use the experimental findings of the first task to suggest an combination of dimensionality reduction technique(s) and social network measure(s).

For the second task we have to apply the resulting visualization method (from the first task) to different adaptive networks with different learning algorithms. For this task we use the platform as described in the previous section. The output of each run is an edgelist corresponding to a weighted network. We use this edgelist to compute the desired network measure(s) using *tnet*, then we fed the computed measures into Matlab to apply the desired dimensionality reduction technique(s).

It is worth-mentioning that prior to applying dimensionality reduction in MatLab we preprocess the dataset generated by tnet so that each row corresponds to a single snapshot of time. Thus, the number of dimensions to be reduced equals the number of computed network measures multiplied by the number of nodes.

The remainder of this section describes further details of experimental settings.

3.3.1 Settings for Task 1: Testing Different Combinations of Dimensionality Reduction Techniques and Social Network Measures

For accomplishing task 1, we conduct two experiments. In the first experiment we test different dimensionality reduction techniques on two dataset corresponding to implementations of two agent learning algorithms. To verify the effect of social network measures, we apply the tests first directly to the datasets, then to results of computing different social network measures on these datasets.

In the second experiment we test the effect of using different network measures on capturing the evolution trend of three versions of a social network, where each version corresponds to a different evolution mechanism.

The remainder of this subsection details these two experiments.

Experiment 1: Comparing Different Dimensionality Reduction Techniques on Existing Datasets of Two Learning Algorithms

In the first experiment, we obtained the data from a simple network configuration in which each node interact with the surrounding neighbours with varying weights in each run. Two simulations were conducted each with a different learning algorithm used by the agents (nodes) in the network and the behaviour of the network was recorded in 499 time snapshots for each simulation. Two datasets (called "type0" and "type9"), each in the form of an edgelist, were fed to the next step of dimensionality reduction.

Dimensionality Reduction of the Edgelist With No Network Measures .

We pre-processed each dataset such that each snapshot represents an instance and each

corresponding edge-list, after transposing it into a single vector, represents the number of original dimensions to be reduced (in this case we had a matrix of size 499×1500).

Dimensionality Reduction with Social Network Metrics In the second part, we computed four metrics that are most common in social network analysis, namely, out-degree, in-degree, betweenness, and closeness. For all of these measures, we used the weighted version of the measure. We also added one other measure and called it "self-load" to represent the weight of the link sent by a node to itself. We then applied some dimensionality reduction techniques (PCA, Isomap, and MDS) on the matrix preprocessed as in the previous part.

Experiment 2: Comparing Different Evolution Mechanisms on a Static Weighted Network

In this experiment we deal with a single static weighted network that is available on the Internet, namely, the Freeman's EIES network with 48 nodes and 695 directed weighted links. To test the hypothesis we imposed in this research, it is necessary to have an evolving network. For this reason we developed three different mechanisms for generating an evolving network from a stationary one as follows.

Mechanism 1

1. Let network_0 = network_original with weights set to 0 (or some small value)
2. Set network_current = network_0
3. Let growable_edges = all edges in network_current (at this point all edges have weights $<$ their weight in network_original)
4. Let e = an edge selected randomly from growable_edges
5. If network_current.weight(e) $\geq$ network_original.weight(e) then
   network_current.weight(e) = network_original.weight(e)
   remove e from growable_edges
   else
   Increase the edge's weight by some small constant
6. repeat steps 4,5 until no more edges in growable_edges

Mechanism 2 is same as mechanism 1 but with step 4 replaced with:
4. Let e = an edge selected from growable_edges proportionally to network_original.weight(e).

Mechanism 3 is same as mechanism 1 but with step 4 replaced with:
4. Let e = an edge selected from growable_edges proportionally to network_current.weight(e) - in network_original.weight(e).

3.3.2 Settings for Task 2: Testing the Visualization Techniques on Networks of Adaptive Agents

In this subsection we describe the settings required for generating edgelists of weighted networks of adaptive agents using the platform we built using NetLogo for this purpose.

For our platform, the parameters that need to be specified for the network to be created are the number of nodes (n) and the average node degree ($\bar{k}$). A random network is created (when the "set up" button is pressed) by setting n nodes at random positions, where each node is randomly assigned one of two types: "row" (for row player) or "column" (for column player), and then link each node to its possible nearest k neighbors based on the its type (a "row" node can only link to a "column" node and vice-versa).

Other parameters that need to be specified by the user prior to running the system are: the value type to be associated with the weights (this can be the value of the iterator, reward, number of wins, or incentive to deviate), the game to be played (e.g. battle of the sexes), the learning algorithm (e.g. Q-learning, IGA), and whether the learning parameters are variable or constant. This last parameter (variable or constant learning parameters) controls the condition of convergence. In variable learning the parameters of the learning algorithm (exploration rate in Q-learning, and step size in IGA for example) decrease by a very small amount after each stage and the system is said to reach convergence when they approach zero. Whereas in constant learning these parameters never change, i.e. they are fixed, and convergence is tested by checking the stability of the 'behavior' rather than the parameters. For example, when the agents tend to choose the same action repeatedly for some time, or when the average (or total) payoff is constant over some period of time (say 100 ticks for example) then we can say that the behavior of the system is stable.

In each iteration each player has to learn two things: what action to play (one action among all players) and which neighbor(s) to play with. Learning about the action uses the learning algorithm as set by the user (Q or IGA), whereas learning about the neighbor to play with always uses Q-learning.

For imposing disruptions either in the learning of k random nodes or by disconnecting k random links (where k is a positive integer) we devised two buttons "disrupt a learning" and "disrupt a link" disrupt a random node's learning functionality or a random link, respectively. When the learning functionality of some agent is disrupted, the agent stops learning and it plays a random action instead and it will not update its learning parameters (Q or IGA policy). On the other hand, when a link is disrupted it is disconnected, thus, it will not be engaged in game playing in later stages. Practically, this means that it will receive zero as its payoff in each later stage.

The final step before running the test experiments is to decide upon the values of the different parameters. For the setting of platform we use battle of the sexes as the game to be played. The learning algorithms we test the visualization on are Q-learning and IGA. For fixed learning parameters in Q-learning we set exploration rate $\epsilon = 0.05$ and learning rate $\alpha = 0.1$. The reward discount is set to 0.9. In variable learning exploration rate and learning rates are initially set to 1, and it decays by percentage of 98% after each iteration, whereas the learning rate decays by 99%. In fixed learning parameters for IGA, the step size(η) is set to 0.01, whereas in variable learning it is initially set to 0.5 and then it decays by 99% after each iteration.

Chapter 4

Results and Discussion

In this thesis we identified two tasks and for each task we outlined several research questions that we seek to answer experimentally. The first task was to study comparisons between different combinations of dimensionality reduction techniques on different combinations of social network measures. The second task was to test the resulting combination on networks of learning agents.

The questions we seek to answer in the first task are:

1. Which dimensionality reduction technique is good for visualizing evolving networks?
2. Which social network measure (or combination of measures) is good for visualizing evolving networks?, and what are the suitable parameter settings of these measures?

Whereas the questions we seek to experimentally answer in the second task are:

1. Can the plots of dimensionality reduction on weighted network metrics capture differences in the dynamics of different learning algorithms, i.e. can we use this technique of visualization in distinguishing learning algorithms?
2. Can dimensionality-reduced weighted network metrics be used to plot and indicate disruptions in learning and disruptions in network structure?
3. Can we use weighted network metrics for identifying the disrupted nodes?

To answer these questions we conducted several experiments with settings as explained in the previous chapter. In this chapter we demonstrate the results of these experiments and discuss the findings.

4.1 Results of Task 1: Comparative Study on Combinations of Dimensionality Reduction Techniques and Social Network Measures

4.1.1 Question 1: Which dimensionality reduction technique to use?

In experiment 1 we have two datasets of a dynamic network, with each dataset corresponding to a different learning algorithm applied on the agents in the network. We applied four dimensionality reduction techniques, namely PCA, Isomap, MDS-fast, and MDScale, on each of the two datasets with one target dimension and without using any social network metric yet. Figure 4.1 illustrates the result.

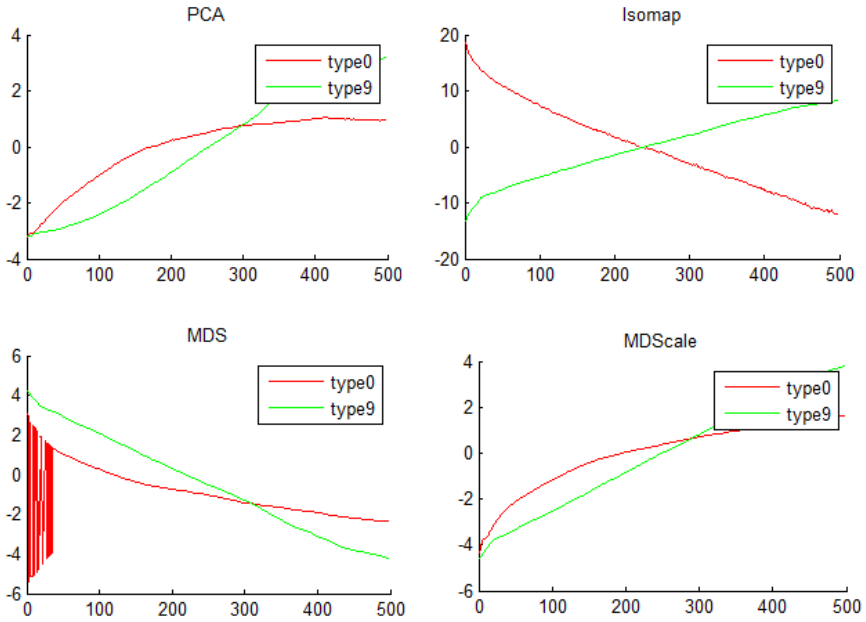

Figure 4.1: Different dimensionality reduction techniques on two datasets (called "type0" and "type9") representing edgelists of two dynamic networks (No network measures are used).

We notice that the resulting figures of all the methods are very similar although the differences in the computation time are quite noticeable with PCA proved to be the method with least computational time (1:2:7:75 were the ratios for PCA:Isomap:MDS scale, and MDS fast respectively). We also make the note that while the two plots that represent the two different data sets are distinctive, but they are linear with no hidden trends captured.

We then test the different dimensionality reduction techniques with social network measures (out-degree, in-degree, betweenness, closeness, and self-load). Figures 4.2 - 4.4 show the results.

We observe that, unlike the first part of the experiment where no network measures are used, at least one plot line deviate from the linear trend as in the result of applying PCA and MDS on the "type9" dataset. This strengthens the case of using PCA as well as social network measures. However, to plot the effect of each network measure we move to experiment 2 as in the following subsection.

4.1.2 Question 2: Which network measure(s) to use?

Before delving into answering this question, we first verify that using network measures in visualization is useful. We do so by comparing the results when network measures are not used with the plots when network measures are used.

As an example we plot the PCA mapping of two lists: one is the edgelist without network metrics (Figure 4.5) and the other is the list of nodes with computed weighted degree centrality over time (Figure 4.6). In this run we disrupt 10 links at time 36.

We notice that the visualization without network metrics fails to capture this disruption in contrast to the case where network metrics were used for the visualization. All the other runs we test (25 more runs) confirm this same conclusion.

We then proceed to the second experiment (explained in details in 3.3). In this experiment we applied three different evolving mechanisms on a social network to turn it from a static one into a dynamic network. We had five runs for each mechanism. We then applied dimensionality

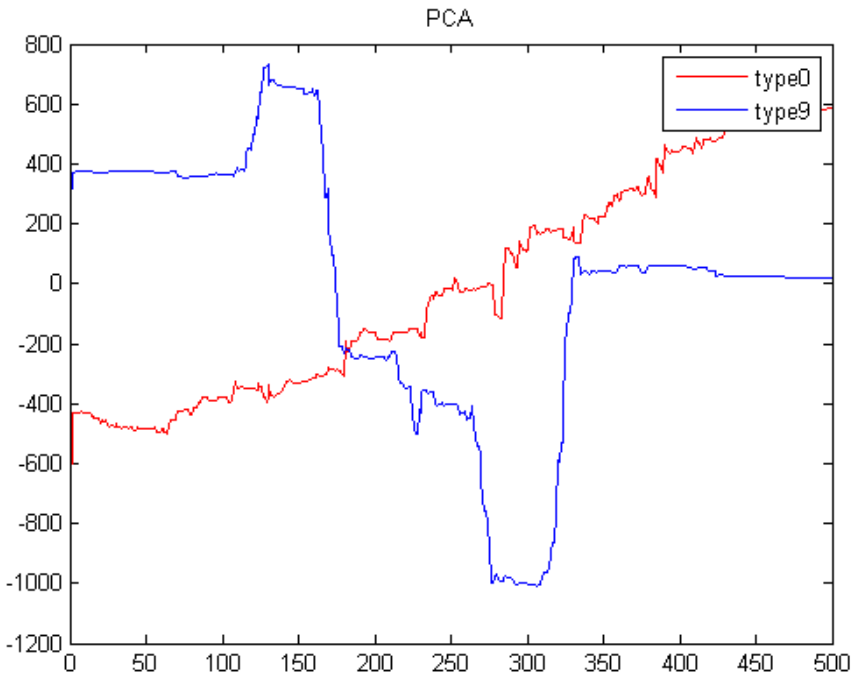

Figure 4.2: Comparison between the results of PCA on two datasets corresponding to two different dynamic networks with network measures applied.

reduction (PCA and MDS with 1 target dimension) on the resulting matrices, as in the previous experiment, first with no network measurements (Figures 4.7 and 4.8), and then with five weighted network metrics, namely, out-degree, in-degree, betweenness, closeness, and clustering coefficient(Figures 4.9 and 4.10).

Two important observations are made. Firstly, dimensionality reduction with network metrics could capture the different trends of network evolution as opposed to dimensionality reduction of the raw matrix, i.e. with no network metrics, where all the plots were identical and linear. Secondly, there exists a distance between plots of different mechanisms which implies that using dimensionality reduction with social network measures for different evolutions of network metrics can be used to differentiate between the underlying mechanisms and, hence, can be used in the classification of networks based on their underlying evolution dynamics.

After confirming the usefulness of a combination of social network measures in visualization, we need to test the effect of each measure individually in order to pinpoint the most effective network metric. For this, we examined the effect of eliminating each of the five network measures we employed on the result of dimensionality reduction. Figures 4.11 - 4.15 illustrate the results.

We observe that the weighted betweenness, followed by the out-degree, have the greatest effect on maximizing the distance, i.e. difference, between the different mechanisms in this case.

However, after conducting several experiments on adaptive networks we favour weighted degree centrality over betweenness.

4.1.3 Question 3: What values for setting the parameters of network metrics?

Before testing different values for degree centrality, we compare it with the C-degree measure. In the following experiment we compare degree centrality with both $\alpha = 0.5$ and $\alpha = 1$ (which represents strength) and c-degree (Figures 4.16 and 4.17). We found that degree centrality was

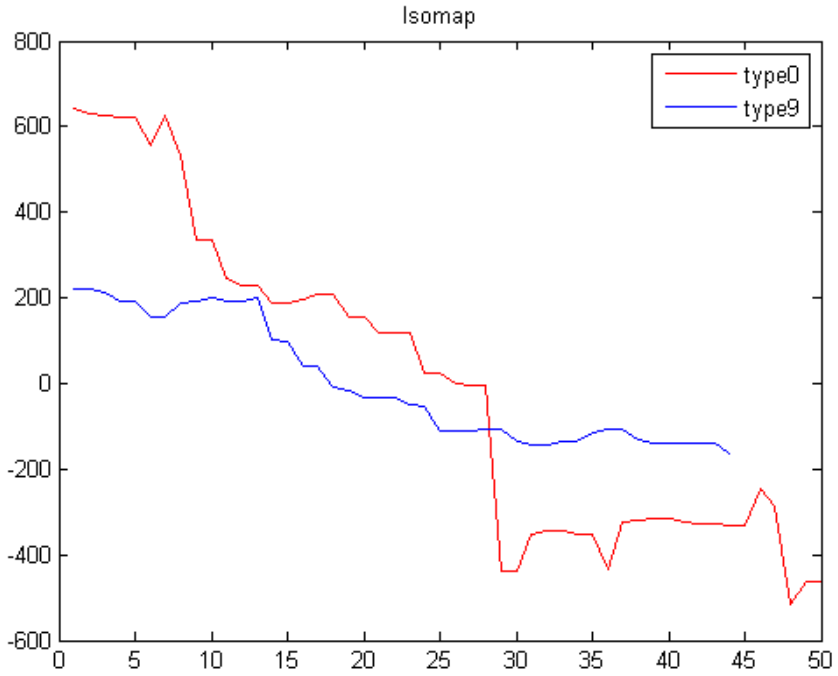

Figure 4.3: comparison between the results of Isomap on two datasets corresponding to two different dynamic networks with network measures applied.

more effective in capturing disruptions in some cases (see the last experiment in Figure 4.17). Also we found that node strength did not offer extra information over weighted degree with $\alpha = 0.5$. Thus, we choose to use degree centrality with $\alpha = 0.5$ in following experimentations.

4.2 Results of Task 2: Testing Dimensionality-Reduced Network Metrics on Networks of Adaptive Agents

Task 2 involves conducting experiments on different networks of adaptive agents. For this we built a platform (as described in Section 3.3) from which we generate edgelists of weighted networks where adaptive agents engage in playing some game (battle of the sexes in this case) using different learning algorithms (Q-learning or IGA in our case). The resulting edgelist is used to compute the desired network metrics to which we apply dimensionality reduction. This section shows the results of these experiments.

4.2.1 Question 1: Can it distinguish different learning algorithms?

We conducted different runs using both learning algorithm (Q-learning and IGA) and then we tried to use the dimensionality-reduced network metric technique of visualization on these runs as in Figure 4.18.

We can clearly observe the distinction made by this visualization technique on the two learning algorithms.

4.2.2 Question 2: Can it capture disruptions in learning?

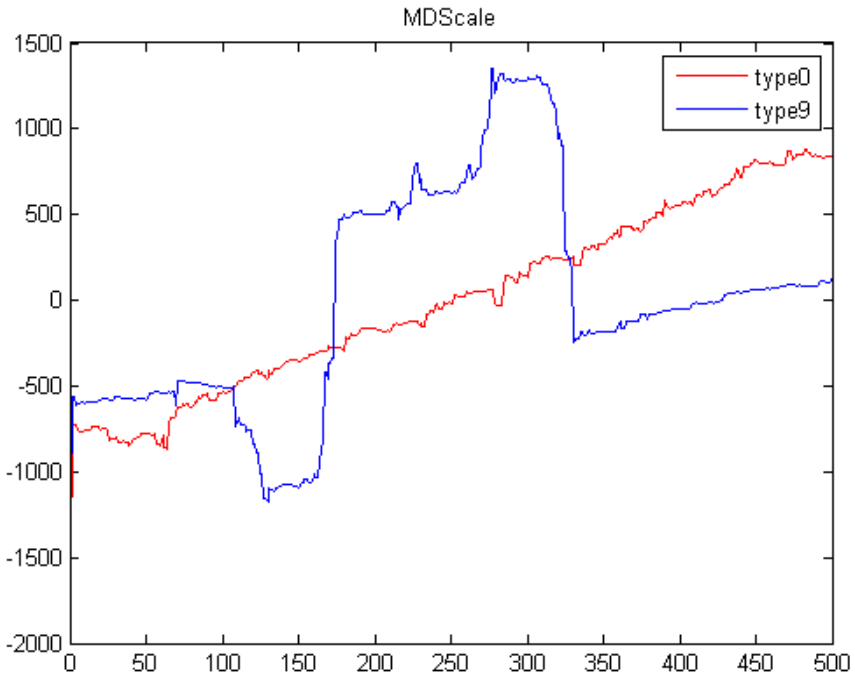

Figure 4.4: comparison between the results of MDS on datasets corresponding to two different dynamic networks with network measures applied.

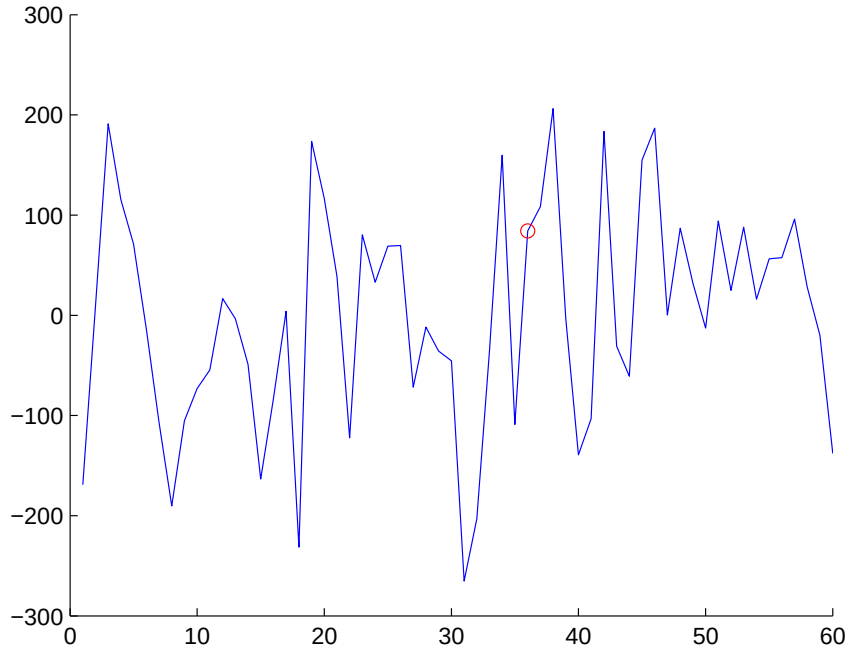

Figure 4.5: Results of PCA on a weighted network (size 100×3) in which 10 random links are disrupted at time 36 as depicted by the red circle. No network metrics are used.

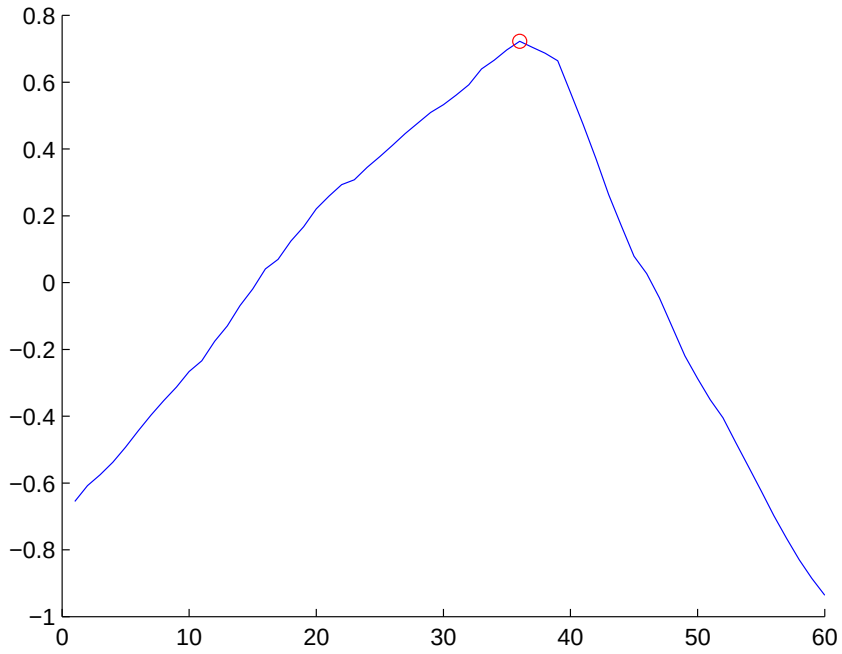

Figure 4.6: Results of PCA on a weighted degree centrality of nodes corresponding to a weighted network of 100 nodes and average node degree of 3 in which 10 random links were disrupted at time 36 as depicted by the red circle.

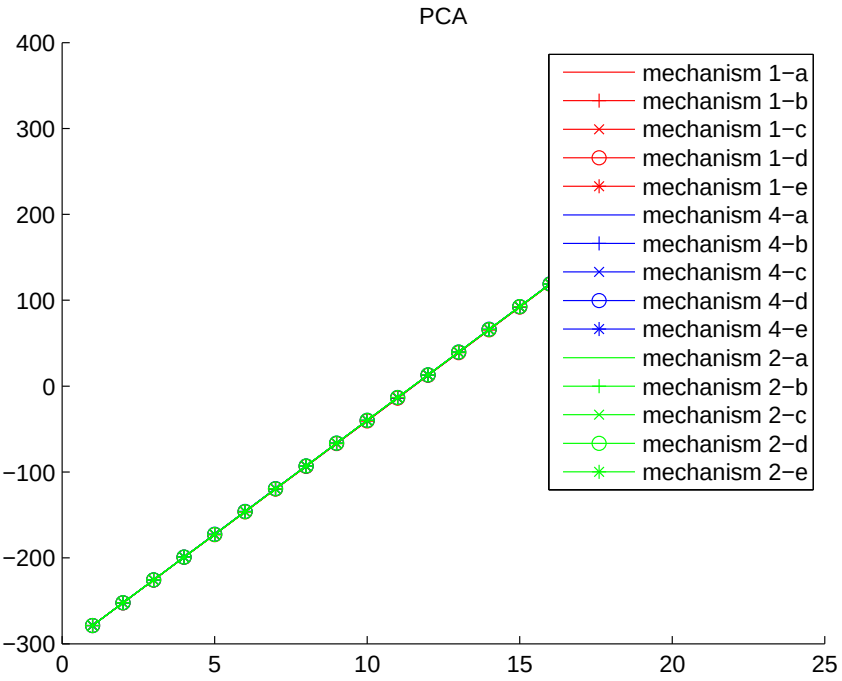

Figure 4.7: Comparison between results of PCA on different evolving networks generated by different mechanisms (5 runs for each mechanism with no network metrics applied).

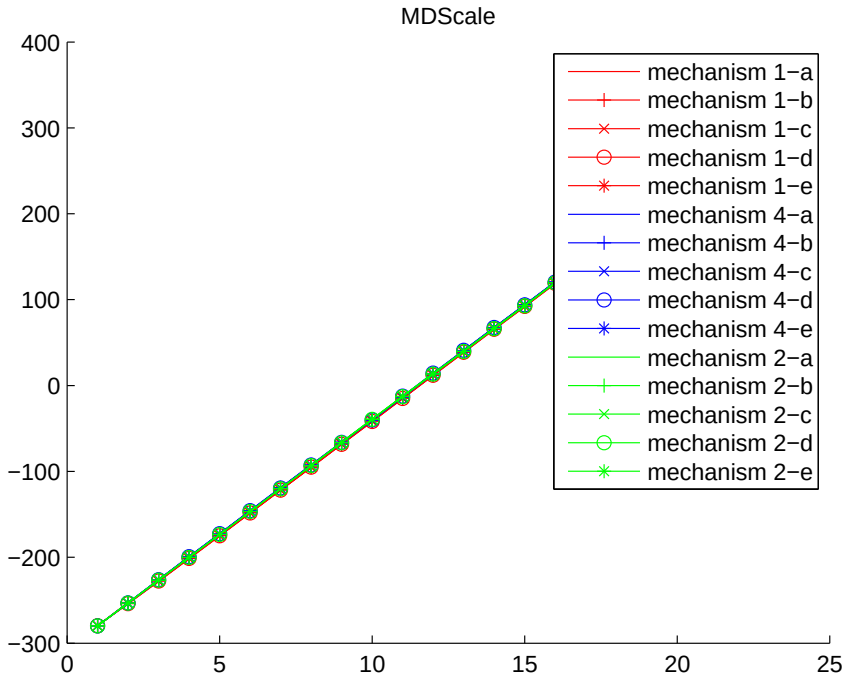

Figure 4.8: Comparison between results of MDS on different evolving networks generated by different mechanisms (5 runs for each mechanism with no network metrics applied).

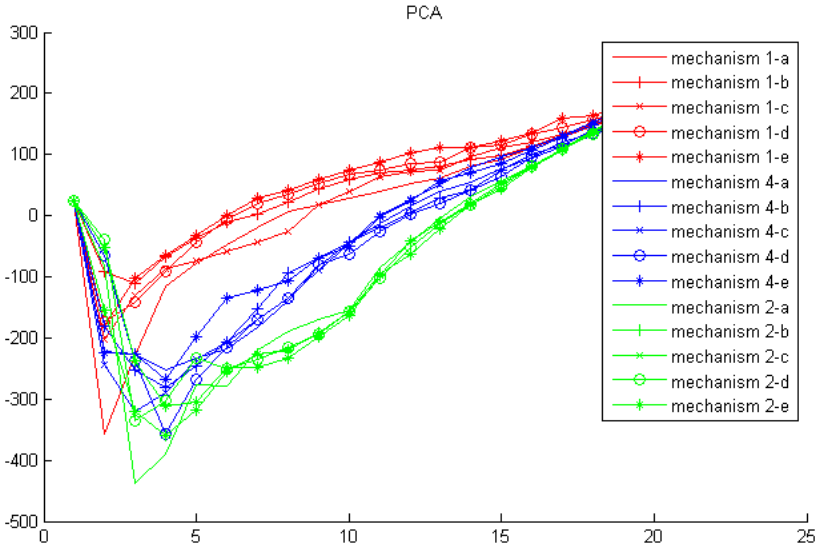

Figure 4.9: Comparison between results of PCA on different evolving networks generated by different mechanisms (5 runs for each mechanism with network metrics).

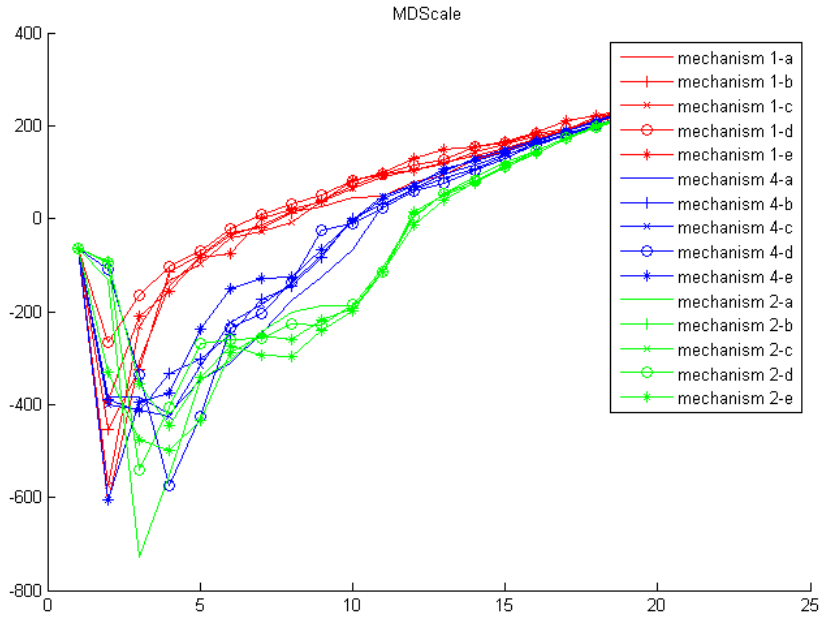

Figure 4.10: Comparison between results of MDS on evolving network generated by different mechanisms (5 runs for each mechanism with network metrics).

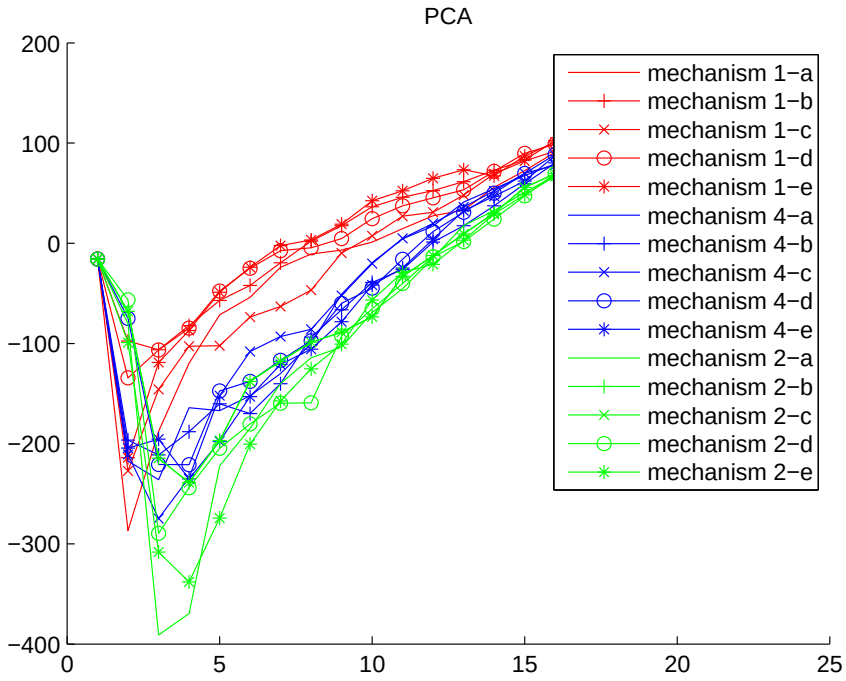

Figure 4.11: Result of PCA on different evolving networks generated by different mechanisms (5 runs for each mechanism with 5 network metrics) with weighted in-degree eliminated.

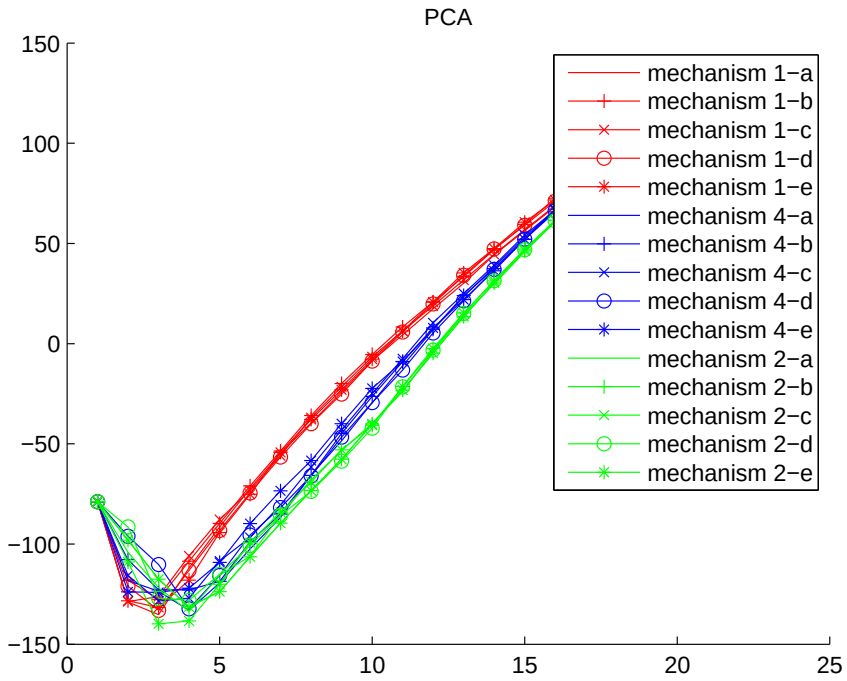

Figure 4.12: Result of PCA on different evolving networks generated by different mechanisms (5 runs for each mechanism with 5 network metrics) with weighted out-degree eliminated

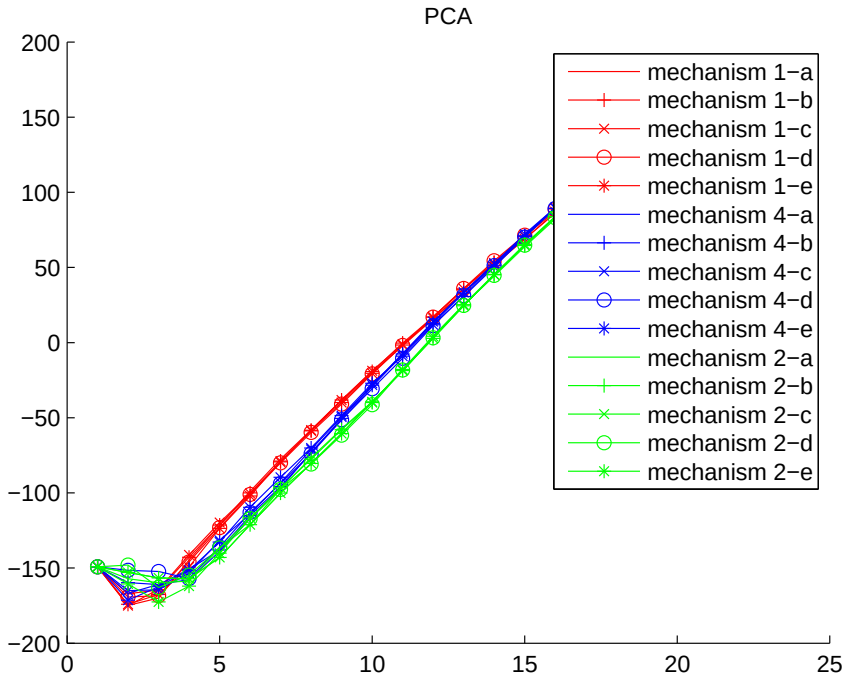

Figure 4.13: Result of PCA on different evolving networks generated by different mechanisms (5 runs for each mechanism with 5 network metrics) with weighted betweenness eliminated

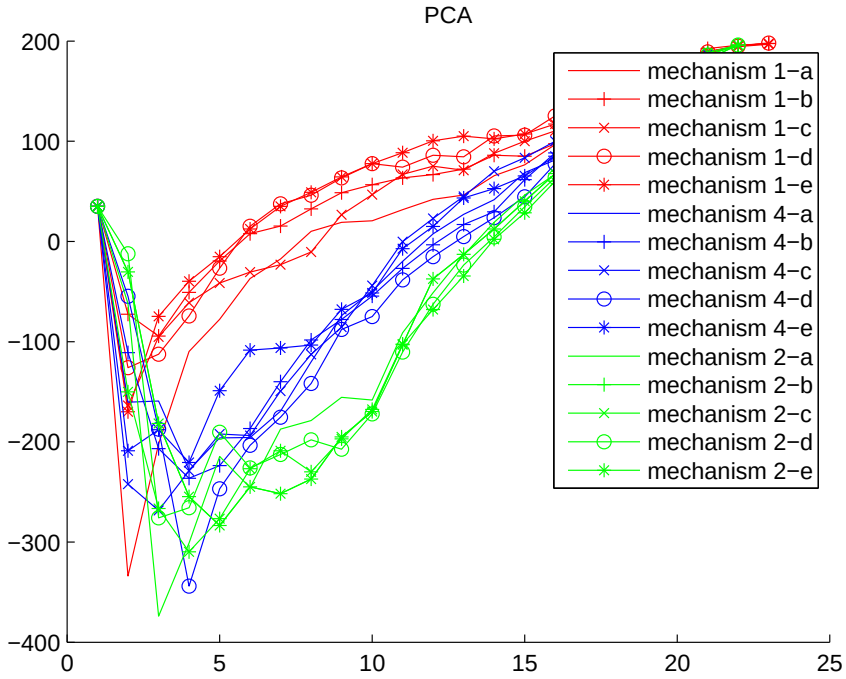

Figure 4.14: Result of PCA on different evolving networks generated by different mechanisms (5 runs for each mechanism with 5 network metrics) with weighted closeness eliminated

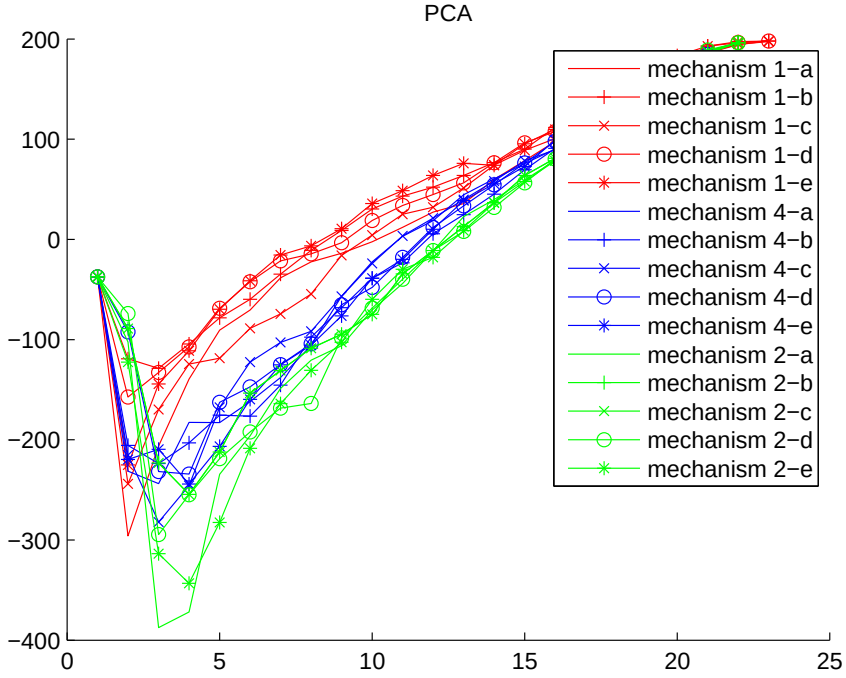

Figure 4.15: Result of PCA on different evolving networks generated by different mechanisms (5 runs for each mechanism with 5 network metrics) with weighted clustering eliminated

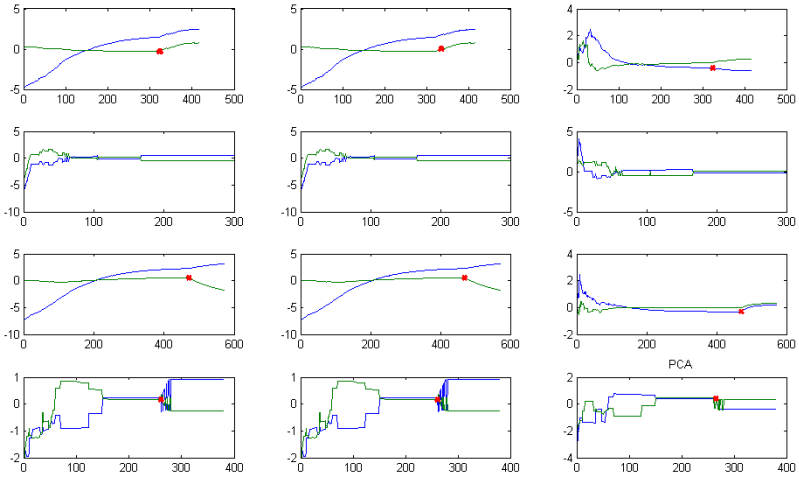

Figure 4.16: Results of PCA (2 dimensions) on a weighted network of adaptive agents. The first column shows the results of weighted degree(in and out with $\alpha = 0.5$), the second column shows the results of node strength (weighted degree with $\alpha = 1$), and the third column is for the c-degree. Each row corresponds to a different experiment with all experiments performed with variable learning parameters as follows: 1: Q-learning, learning disruption, 2: IGA: learning disruption, 3: Q, link disruption, 4: IGA, link disruption.

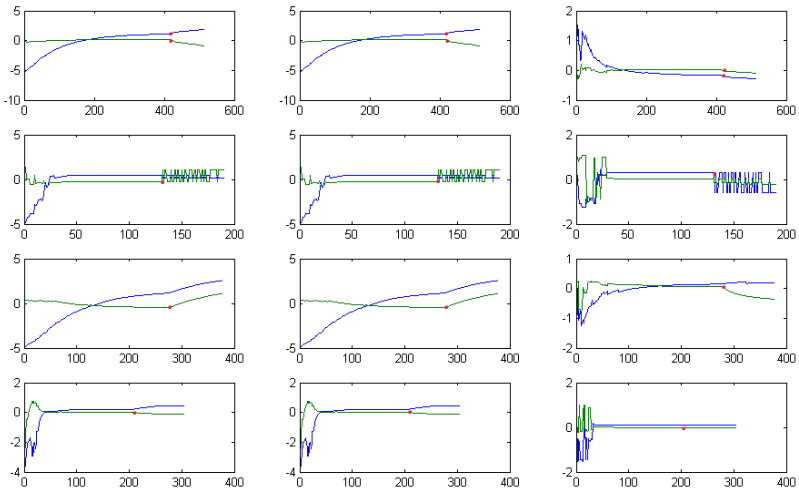

Figure 4.17: Results of PCA (2 dimensions) on a weighted network of adaptive agents. The first column shows the results of weighted degree(in and out with $\alpha = 0.5$), the second column shows the results of node strength (weighted degree with $\alpha = 1$), and the third column is for the c-degree. Each row corresponds to a different experiment with all experiments performed with variable learning parameters as follows: 1: Q, learning disruption, 2: IGA: learning disruption, 3: Q, link disruption, 4: IGA, link disruption.

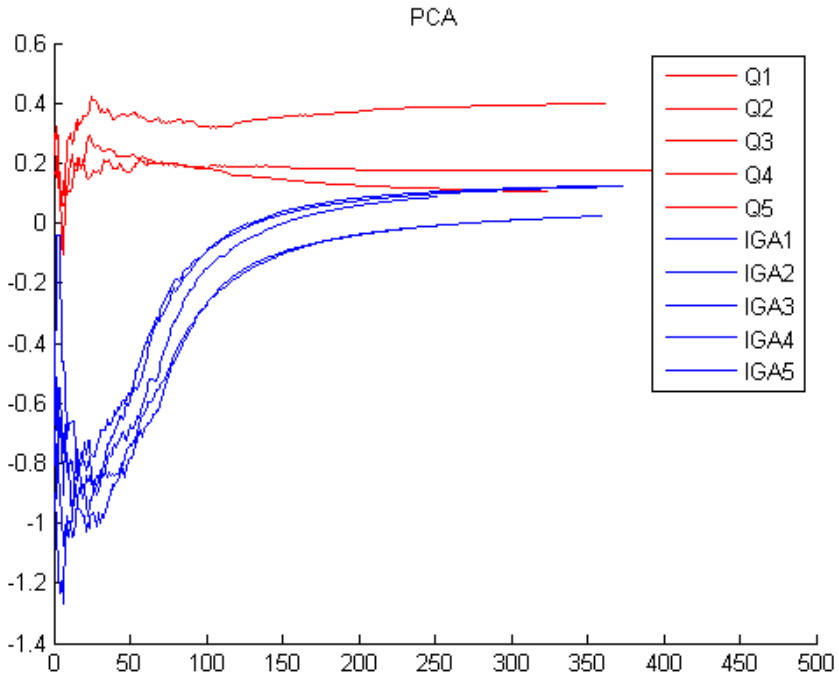

Figure 4.18: Result of PCA (with target dimensions is 1) on weighted closeness for weights corresponding to values of the iterator (Q vector in Q-learning and policy in IGA) in 10 different runs on a network of 5 nodes and average node degree of 2. The game played is Battle of the Sexes.Parameters for the learning algorithms are as follows: Q (ε initially is 1 and iteratively decays in multiples of 0.98, $\alpha = 0.1$, $\gamma = 0.9$). IGA (η initially is 0.03 and then iteratively decays in multiples of 0.99).

When learning is disrupted at some stage for some agent, it will simply stop learning. This is simulated by having it playing a random action in each forthcoming stage. Thus, it will stop maintaining the values of the learning parameters.

We examined different runs for adaptive agents situated in random networks of different sizes (number of nodes $\times$ average node degree),ranging from 5×3 to 100×4, where the learning of random node(s) is disrupted at different times after convergence. We then computed the degree centrality for the resulting edgelist on which we used PCA to reduce the dimensionality. We plot the results of dimensionality reduction in 2D and mark the point of disruption in each run as shown in Figures 4.19 (with Q-learning) and 4.20 (with IGA) .

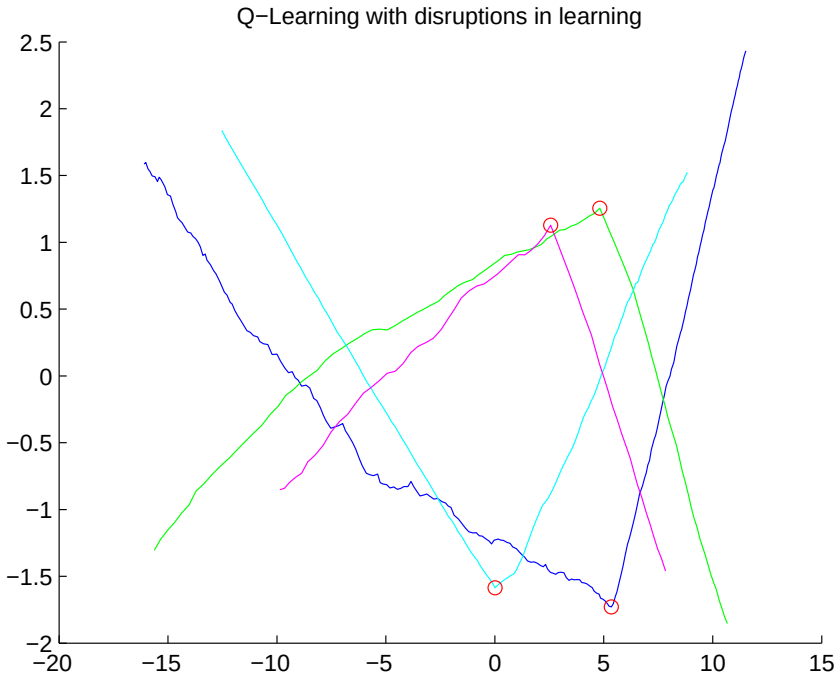

Figure 4.19: Results of PCA (2 dimensions) on weighted degree (in-degree and out-degree with $\alpha = 0.5$ corresponding to weighted networks of different node and average link degrees. The learning of 20% random nodes is disrupted at different times after convergence. Learning algorithm is Q-learning.

4.2.3 Question 3: Can it capture disruptions in network structure?

When a link is disrupted, i.e. disconnected, at some stage it will not take part in playing games with neighbors and, hence, will receive a zero payoff. As in the previous section, we plot the results of the weighted degree centrality of different runs where in each run one or more random links are disconnected after convergence. Figures 4.21 and 4.22 illustrate these plots in 2D where each plot correspond to a different run and the point of disruption is circled in red.

4.2.4 Question 4: Can it be used to identify the type and source of disruption?

If we can distinguish between different types of disruptions and identify the source of this disruption, then we can say that our visualization technique can be used as a means of explanatory data analysis for multiagent systems.

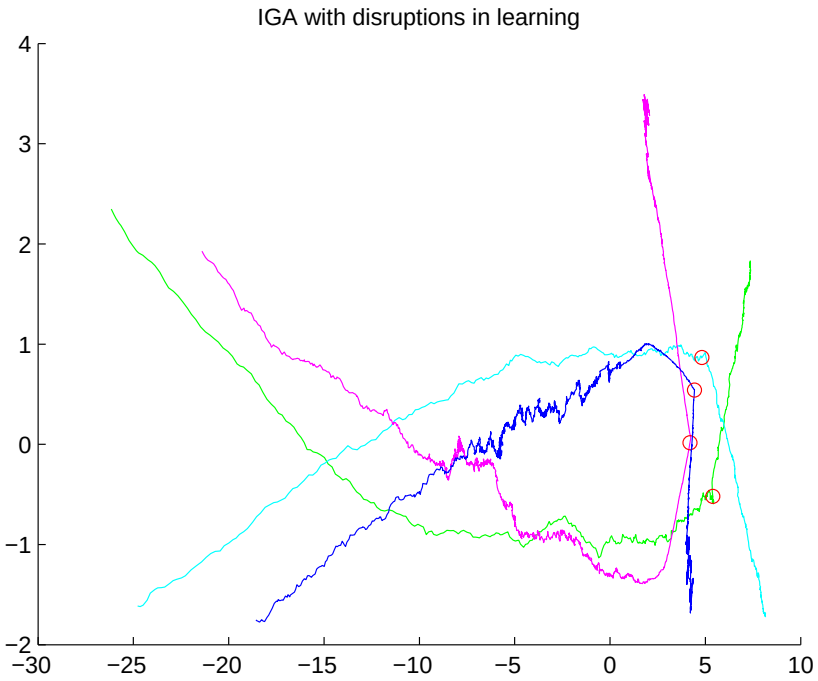

Figure 4.20: Results of PCA (2 dimensions) on weighted degree (in-degree and out-degree with $\alpha = 0.5$ corresponding to weighted networks of different node and average link degrees. The learning of 20% random nodes is disrupted at different times after convergence. Learning algorithm is IGA.

As an experiment we plotted different lists of weighted node degree centrality metric that we computed for previous runs over time and we could observe that these network metrics of disrupted nodes show different trends than those of undisrupted nodes (see for example Figure 4.23 below).

However, in the experiments we conducted it was not clear how to distinguish between different types of disruption.

4.3 Discussion

In this section we summarize the findings we reached at for tasks 1 and 2 by presenting the answers provided by the experiments we conducted to the questions we posed earlier.

The questions we aimed at answering in the first task pertain to finding a combination of a dimensionality reduction technique and social network measures. From experiment 1 we could conclude that using PCA as a dimensionality reduction technique is good for our visualization purposes. It shows same patterns illustrated by other techniques (Isomap and MDS) while outperforming theses methods in its computational time (it required much more less time for pruducing the outputs). Moreover, we found that setting the target number of intrinsic dimensions to 1 or 2 is sufficient for our visualization purposes. Thus, we succeeded at reducing the large number of parameters to only one or tow parameters.

In the second experiment we aimed at testing different social network measures. We could observe that using a combination of these measures is informative. This was verified when comparing plots of dimensionality reductions in edgelists (with no network measures computed), to lists of network measures computed. We observe that the plots of network measures could capture hidden patters that could not be captured using the edgelists alone.

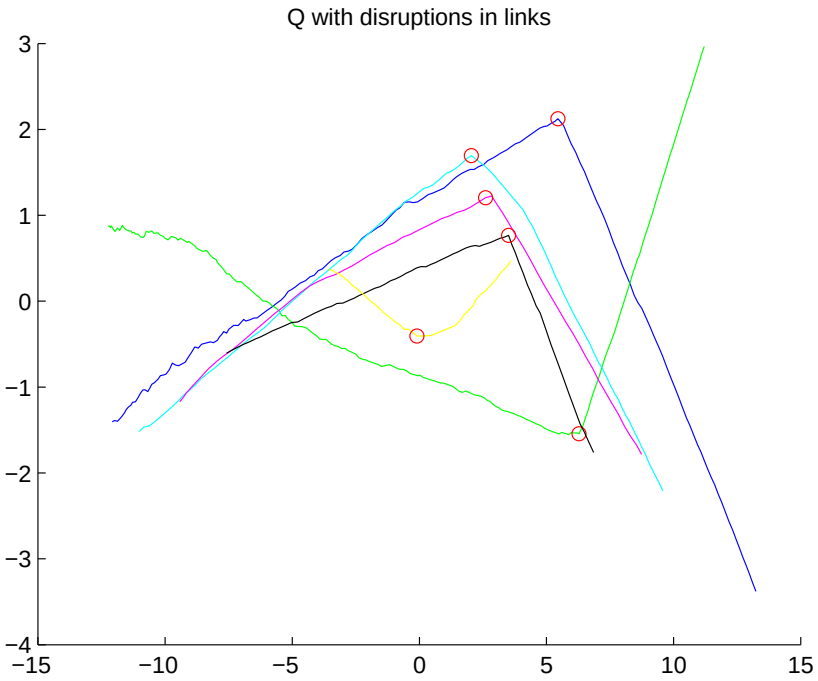

Figure 4.21: Results of PCA (2 dimensions) on weighted degree (in-degree and out-degree with $\alpha = 0.5$) corresponding to different weighted networks. Random links are disrupted at different times after convergence. Learning algorithm is Q-learning.

Moreover, when used individually these measures also succeed at plotting the underlying trends. This experiment evaluated the use of betweenness, centrality degree, closeness, clustering coefficient, and c-degree and found that the use of weighted centrality degree suffices to our visualization purpose. Finally, we conducted some experiments to find a suitable value for the α parameter used in computing the weighted degree centrality. We found that setting this parameter to the value of 0.5 (which places equal importance on the number of ties and the weights of these ties) generates better results. We summarize these findings of the first task in the following central claim.

Claim 1: using dimensionality reduction with social network measures in visualizing the performance of networks of adaptive agents can capture the performance of the system using few parameters (1 or 2) while being able to capture hidden patterns (that could not be captured using weights alone without computing the network measures).

In the second task, we applied this technique of dimensionality-reduced network measures on networks of adaptive agents. We succeeded at answering the research questions pertinent to task 2 as stated in the following claims which collectively prove the capability of the proposed technique in capturing the performance of multiagent systems at the macro level (claim 2) as well as at the micro level (claims 3,4, and 5).

Claim 2: using network metrics in visualizing the performance of adaptive networks under different learning algorithms can produce distinctive plots for each learning mechanism.

We experimentally tested this claim as summarized in Section 4.2.1 and showed that the resulting plots of weighted network metrics corresponding to two different learning algorithms, namely Q-learning and IGA (where the weights represent the values of Q_a and $max(\alpha, \alpha - 1)$ respectively), are distinguishable.

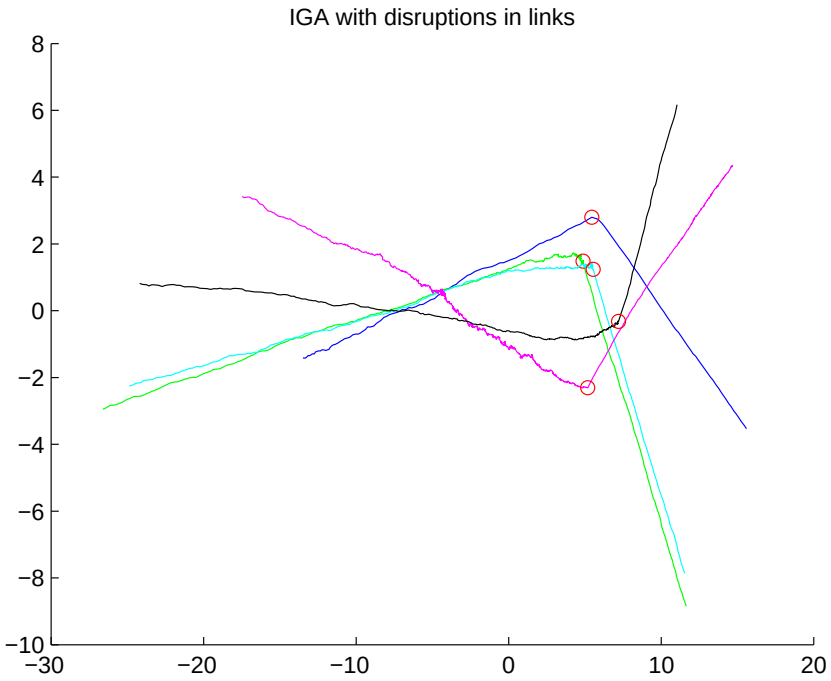

Figure 4.22: Results of PCA (2 dimensions) on weighted degree (in-degree and out-degree with $\alpha = 0.5$) corresponding to different weighted networks. Random links are disrupted at different times after convergence. Learning algorithm is IGA.

Claim 3: using network metrics in visualizing the performance of adaptive networks can be informative in capturing disruptions in the agent learning process

As shown in Section 4.2.2, when reducing the dimensionality (to two dimensions in this case) of a list of node degree centrality of some dynamic network in which we incur learning disruptions in a number of random nodes, the resulting plots significantly show the point of disruption. Moreover when plotting the result of PCA mapping in one dimension, where the other dimension is the time, we can identify the unique point of time at which the disruption occurred as shown in Figure 4.24 below.

Claim 4: using network metrics in visualizing the performance of adaptive networks can be informative in capturing disruptions in the network structure.

Similar to the previous claim, we conducted several runs of different adaptive networks in which we disconnected random links after convergence and used the visualization technique we are proposing to test whether it can capture the incurred disruptions. Results of these runs are shown in Section 4.2.3. This confirms our claim.

Claim 5: using network metrics in visualizing the performance of adaptive networks can be informative in identifying the type and source of disruption.

While in this research, we could not prove the first part of this claim (that we can use this visualization technique in identifying the type of disruption, whether it is in learning or network structure), but experimental results suggested that we can use it in identifying the source of disruptions (i.e. which nodes are disrupted). This was shown in Section 4.2.4 .

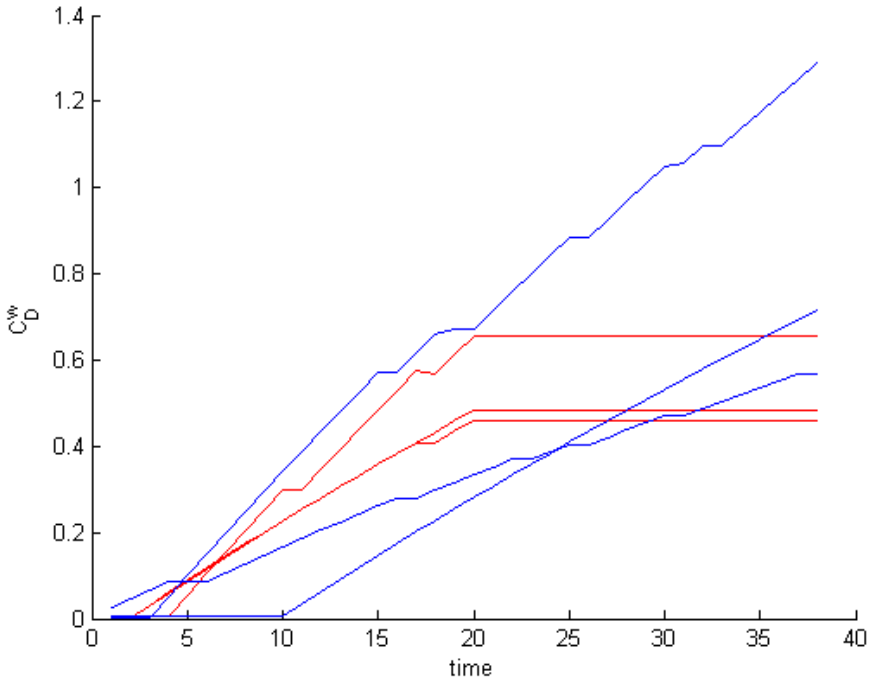

Figure 4.23: Plotting weighted degree centrality of six nodes, three of which have their learning disrupted (the ones in red) at time 20.

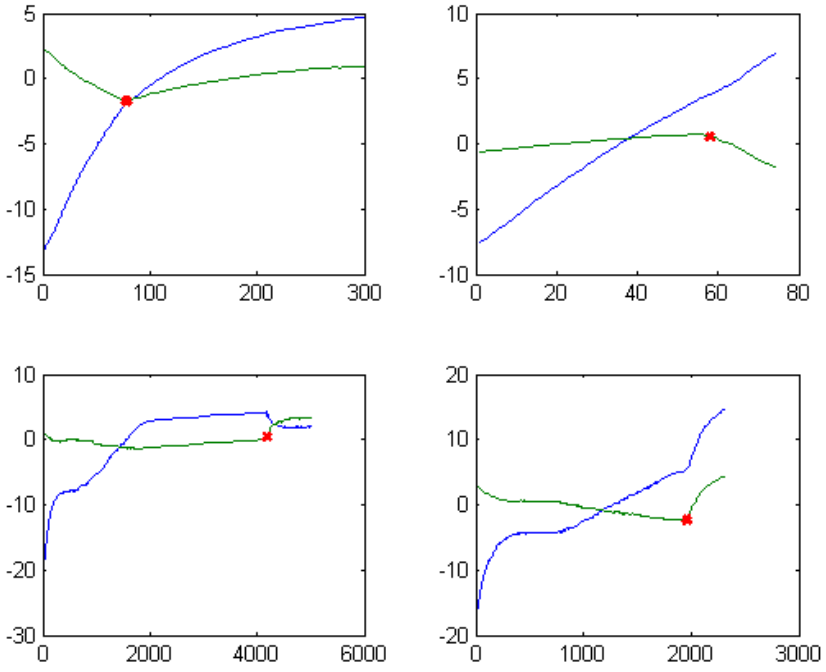

Figure 4.24: Results of PCA on different weighted degree centrality of nodes corresponding to different weighted networks. Different disruptions occurred at different times and the point of disruption is marked by a red circle. The first row corresponds to runs that used Q-learning and the second row corresponds to runs that used IGA.

Chapter 5

Conclusion and Future Work

In this dissertation we aimed at finding a technique for visualizing the performance of multiagent systems in a way that is capable of summarizing the global performance of the whole system (on its macro level) by as few parameters as possible, while being able to remain sensitive to the dynamics of the individual agents (the micro level). For this we proposed the use of a combination of dimensionality reduction and weighted network metrics as a means of visualizing the performance in networks of adaptive agents. We suggested this technique as one possible solution to satisfy all the conditions of (still not fully realized in other existing MAS visualization techniques): a) summarizing the many parameters in one or two parameters, b) summarizing the performance of the system on its macro level (i.e. the global performance of the whole system) , and c) capturing the performance on the micro level (i.e. being sensitive to the dynamics of individual nodes).

To verify our proposed method in terms of these requirements, we specified two tasks: firstly, we were required to perform a comparative study of different combinations of dimensionality reduction techniques and social network measures, and secondly had to test this technique, based on the findings from the first task, on networks of adaptive agents.

For the first task we conducted a couple of initial experiments with the purposes of testing different dimensionality techniques with as few target parameters as possible (1 or 2 intrinsic parameters), testing the effectiveness of a combination of social network metrics (as opposed to cases where no network measures are used), and to identify the most effective network measures for our purpose of visualization.

The experimental tests we conducted for the first task could verify the following claim.

Claim 1: *Using dimensionality reduction with social network measures in visualizing the performance of networks of adaptive agents can capture the performance of the system using few parameters (1 or 2) while being able to capture hidden patterns (that could not be captured using weights alone without computing the network measures).*

Moreover, the findings suggested that the use of PCA as a dimensionality reduction technique with weighted degree centrality (with $\alpha = 0.5$) and/or weighted closeness with (with $\alpha = 0.5$) gives good visualization results.

Then we proceeded into testing dimensionality-reduced weighted network metrics on networks of adaptive agents who engage in playing games with each other, as required in the second task. For this we developed a testing platform using NetLogo [25], the multiagent programmable modelling environment. We conducted our runs on "the battle of the sexes" game using two learning algorithms: the Q-learning and the IGA. The output of this developed system is a weighted edgelist that we fed into t-net [12], a software for analysing weighted networks, to produce the corresponding list of the desired weighted network metrics for the nodes over time. We then applied PCA, a dimensionality reduction technique that is part of the dimensionality reduction toolbox in Matlab, on the lists of computed network metrics to generate the desired

visualization.

The results of the experiments have confirmed several claims we made in this study as in the following list. These claims collectively satisfy both requirements of the desireed visualization technique, that is of being able to capture the performance of the multiagent systems at the macro level (claim 2), as well as at the micro level (claims 3-5).

Claim 2: *Using network metrics in visualizing the performance of adaptive networks under different learning algorithms can produce distinctive plots for each learning mechanism.*

Claim 3: *Using network metrics in visualizing the performance of adaptive networks can be informative in capturing disruptions in the the multiagent learning algorithm.*

Claim 4: *Using network metrics in visualizing the performance of adaptive networks can be informative in capturing disruptions in the network structure.*

Claim 5: *Using network metrics in visualizing the performance of adaptive networks can be informative in identifying the type and source of disruption.*

Many things can be done, as a future work, to consolidate our findings and build on top of it. For example, we can extend the testing to other games (such as the Prisoner's Dilemma and Matching Pennies) and other learning algorithms. We can also increase the degree of freedom pertinent to the dynamics of the network by letting the agents create new links and collapse existing ones. We can also investigate other metrics for weighted networks. Different dimensionality reduction techniques can also be tested.

Aside from trying different tunings and parameters, one important extension to this work is by testing whether this technique can be used for explaining the performance of the system, for example by unambiguously identifying the type of the disruption in case it occurred in the system.

Bibliography

[1] S. Abdallah. Using graph analysis to study networks of adaptive agent. In *Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems: volume 1 - Volume 1*, AAMAS '10, pages 517–524, Richland, SC, 2010. International Foundation for Autonomous Agents and Multiagent Systems.

[2] A. Barrat, M. Barthélemy, R. Pastor-Satorras, and A. Vespignani. The architecture of complex weighted networks. *Proceedings of the National Academy of Sciences of the United States of America*, 101(11):3747–3752, March 2004.

[3] M. Bowling. Convergence and no-regret in multiagent learning. In Lawrence K. Saul, Yair Weiss, and Léon Bottou, editors, *Advances in Neural Information Processing Systems 17*, pages 209–216. MIT Press, Cambridge, MA, 2005.

[4] E. W. Dijkstra. A note on two problems in connexion with graphs. *Numerische Mathematik*, 1(1):269–271, December 1959.

[5] L. C. Freeman. Centrality in social networks conceptual clarification. *Social Networks*, 1(3):215 – 239, 1978-1979.

[6] I. T. Jolliffe. *Principal Component Analysis*. Springer, second edition, October 2002.

[7] L. P. Kaelbling, M. L. Littman, and A. W. Moore. Reinforcement learning: A survey. *Journal of Artificial Intelligence*, 4(1):237–285, 1996.

[8] J. A. Lee and M. Verleysen. *Nonlinear Dimensionality Reduction*. Springer Publishing Company, Incorporated, 1st edition, 2007.

[9] MathWorks. Matlab toolbox for dimensionality reduction. `http://homepage.tudelft.nl/19j49/Matlab_Toolbox_for_Dimensionality_Reduction.html`, November 2010.

[10] J.F. Nash. Equilibrium points in n-person games. *Proceedings of the National Academy of Sciences of the United States of America*, 36:48–49, 1950.

[11] M. E. J. Newman. Analysis of weighted networks. *PHYS.REV.E*, 70:056131, 2004.

[12] T. Opsahl. *Structure and Evolution of Weighted Networks*. PhD thesis, University of London, London, UK, 2009. pp. 104-122.

[13] T. Opsahl, F. Agneessens, and J. Skvoretz. Node centrality in weighted networks: Generalizing degree and shortest paths. *Social Networks*, 32(3):245–251, July 2010.

[14] K. Pearson. On lines and planes of closest fit to systems of points in space. *Philosophical Magazine*, 2(6):559–572, 1901.

[15] Y. Shoham and K. Leyton-Brown. *Multiagent Systems: Algorithmic, Game-Theoretic, and Logical Foundations*. Cambridge University Press, Cambridge, UK, 2009.

[16] S. P. Singh, M. J. Kearns, and Y. Mansour. Nash convergence of gradient dynamics in general-sum games. In *Proceedings of the 16th Conference on Uncertainty in Artificial Intelligence*, UAI '00, pages 541–548, San Francisco, CA, USA, 2000. Morgan Kaufmann Publishers Inc.

[17] R. S. Sutton and A. G. Barto. *Reinforcement Learning: An Introduction (Adaptive Computation and Machine Learning)*. The MIT Press, March 1998.

[18] J. B. Tenenbaum, V. Silva, and J. C. Langford. A Global Geometric Framework for Nonlinear Dimensionality Reduction. *Science*, 290(5500):2319–2323, 2000.

[19] W. S. Torgerson. Multidimensional scaling: I. theory and method. *Psychometrika*, 17:401–419, 1952.

[20] H. van den Herik, D. Hennes, M. Kaisers, K. Tuyls, and K. Verbeeck. Multi-agent Learning Dynamics: A Survey. In *Cooperative Information Agents XI*, Lecture Notes in Computer Science, pages 36–56. Springer Berlin / Heidelberg, September 2007.

[21] F. Vega-Redondo. *Game Theory and Economics*. Cambridge University Press, Cambridge, MA, 2001.

[22] J. M. Vidal. *Fundamentals of Multiagent Systems: Using NetLogo Models*. Unpublished, 2006.

[23] J. von Neumann and O. Morgenstern. *Theory of games and economic behavior*. Princeton University Press, 1947.

[24] C. Watkins. *Learning from Delayed Rewards*. PhD thesis, University of Cambridge,England, 1989.

[25] U. Wilensky. Netlogo. http://ccl.northwestern.edu/netlogo/. Center for Connected Learning and Computer-Based Modeling, Northwestern University, Evanston, IL., 1999.