

EVOLUTION OF AEC PROJECT NETWORKS:  
AN AGENT-BASED MODELING APPROACH

By

Nishchay Pidiha

A THESIS

Submitted to  
Michigan State University  
in partial fulfillment of the requirements  
for the degree of

Construction Management – Master of Science

2021

## **ABSTRACT**

### **EVOLUTION OF AEC PROJECT NETWORKS: AN AGENT-BASED MODELING APPROACH**

By

Nishchay Pidiha

In Architecture, Engineering, and Construction (AEC) industry, projects call for collaboration between different expertise areas. To improve it between different stakeholders, an in-depth understanding of the communication network structure is crucial. While Social Network Analysis (SNA) shows promise in analyzing communication network structures, the AEC literature to date shows its use mainly in a cross-sectional manner. A recent longitudinal case study shows that these networks are not static and can evolve overtime. However, due to fast-paced delivery of any AEC project, researchers often arrive at missing data, sometimes depriving them from concluding statistically significant results in a longitudinal study. Hence, there is a need for further investigating the evolution of AEC project teams network structures through a simulation that emulate certain aspects of a social network structure evolution over time.

In response to this need, the study aims to explicate, how real-world size AEC project networks evolve over project delivery based on similarity of node characteristics (e.g., homophily) and node behaviors in networks (e.g., node's popularity). To do so, the study adopted selection model, and simulated multiple agent-based models basing its initial condition on an empirical case study. The evolutions of the network structures were analyzed overtime. Finally, the study compared the results with recent similar works. Deliverables include an improved understanding of AEC project network structures and characteristics, and practical implications relating to team collaboration and coordination considering project characteristics such as team size, and complexity.

Copyright by  
NISHCHAY PIDIHA  
2021

I would like to dedicate this writing to my parents who have taught me values and have supported me in the pursuit of every little or big goal I wanted to achieve in life. I am, for life, in debt for their effort.

## **ACKNOWLEDGEMENT**

With a few words of gratitude, I would like to acknowledge my thesis advising committee chair and members to whom I owe the accomplishment of this work.

I am immensely grateful to Dr. Sinem Mollaoglu, for her unceasing advice, expert guidance, and support. Without her this work would have not been accomplished.

I am immensely thankful to the thesis advisory committee member: Dr. Kenneth Frank, and Dr. Dong Zhao. Their involvement and idea sharing were the inception point for the approach of this study and had made all the difference enhancing the quality of this work.

I would also like to thank Dr. Ran Xu for taking the time out to review and provide his invaluable input in the simulation model and coding that guides this research.

I would also like to thank all the Dr. Matt Syal, Dr. Mohamed El-gafy, Dr. Tim Mrozowski, and Dr. George Berghorn for their invaluable support during these unprecedented times of Covid-19 pandemic.

I am thankful to all the IOPT-4 research team members for investing their time in discussing my works and its improvement, throughout the journey of my master's degree research-work: Dr. Angelo Garcia, Meltem Duva, Dr. Faizan Shafique, Dr. Joseph Thekinen, Hasan Bayhan, and Nishchhal Nihal Pandey.

Last but not the least, I would like to also thank you the SPDC staff for making everything work smoothly within the school: Jill Selke, Pat Daughenbaugh, Lauri Stephens, and Bill Balluff.

# TABLE OF CONTENTS

LIST OF FIGURES .....	viii
Chapter 1 INTRODUCTION.....	1
1.1. Overview.....	1
1.2. Need Statement .....	3
1.3. Research Goal and Objectives .....	4
1.4. Scope.....	5
1.5. Overview of Methods .....	6
1.6. Results and Deliverables.....	8
1.7. Reader's Guide.....	8
Chapter 2 LITERATURE REVIEW .....	9
2.1. Introduction.....	9
2.2. Overview of Social Network Analysis (SNA) and Key Concepts.....	9
2.3. Social Network Structures .....	12
2.4. AEC Project Teams and Social Network Analysis .....	15
2.4.1 Overview of SNA in the AEC Literature .....	15
2.4.2 AEC Project Teams, Coordination, and Knowledge Transfer .....	16
2.4.3 Overview of Longitudinal SNA Works of AEC Project Teams .....	18
2.5. Node and Dyad Characteristics in SNA and Evolution of Network Structures.....	19
2.6. ABM, AEC Research Uses, and SNA Integration.....	24
2.7. Summary .....	28
Chapter 3 METHODOLOGY .....	30
3.1. Introduction.....	30
3.2. Research Goals and Objectives.....	30
3.3. Research Approach and Scope.....	32
3.4. Model Description and Simulations.....	33
3.4.1 Selection Model for Homophily/Heterophily and Popularity Seeking .....	33
3.4.2 ABM Conditions and Steps to Run the Simulation .....	35
3.5. Research Quality .....	40
3.6. Summary .....	42
Chapter 4 RESULTS.....	43
4.1 Introduction.....	43
4.1 Homophily/Heterophily versus Popularity (Degree / Eigenvector Centrality) .....	43
4.2.1 Experiment 1: Small Sized Network using Degree Centrality and Homophily .....	48
4.2.2 Experiment 2: Real-world Sized Network using Degree Centrality and Homophily .....	53
4.2.3 Experiment 3: Small Sized Network using Eigenvector Centrality and Homophily .....	58
4.2.4 Experiment 4: Real-world Sized Network using Eigenvector Centrality and Homophily .....	64
4.2.5 Experiment 5: Small Sized Network using Degree Centrality and Heterophily .....	70
4.2.6 Experiment 6: Real-world Sized Network using Degree Centrality and Heterophily .....	72
4.2.7 Experiment 7: Small Sized Network using Eigenvector Centrality and Heterophily .....	74
4.2.8 Experiment 8: Real-world Sized Network using Eigenvector Centrality and Heterophily .....	76

4.2	Summary .....	78
Chapter 5 DISCUSSION AND CONCLUSION .....		79
5.1	Introduction.....	79
5.2	Summary of Findings.....	79
5.3	Discussions .....	80
5.4	Conclusion .....	86
APPENDICES .....		89
APPENDIX A: Netlogo Code for Experiments.....		90
APPENDIX B: Netlogo Interface .....		142
REFERENCES .....		145

## LIST OF FIGURES

Figure 3.1 Investigation Framework.....	32
Figure 3.2 Type-1 Initial Setup: a NetLogo Screenshot .....	37
Figure 3.3 Type-2 Initial Setup: a NetLogo Screenshot .....	38
Figure 3.4 Diagram Illustrating Agent’s Decision Making Process .....	39
Figure 4.1 Matrix Explaining all Eight Types of Simulation Experiments for the Research	45
Figure 4.2 Experiment-1 Simulation Results.....	49
Figure 4.3 Plotlines Summarizing the Observed Network Structures in Simulation Experiment-1.....	52
Figure 4.4 Experiment-2 Simulation Results.....	54
Figure 4.5 Plotlines Summarizing the Observed Network Structures in Simulation Experiment-2.....	57
Figure 4.6 Experiment-3 Simulation Results.....	59
Figure 4.7 Plotlines Summarizing the Observed Network Structures in Simulation Experiment-3.....	63
Figure 4.8 Experiment-4 Simulation Results.....	65
Figure 4.9 Plotlines Summarizing the Observed Network Structures in Simulation Experiment-4.....	69
Figure 4.10 Plotlines Summarizing the Observed Network Structure in Simulation Experiment-5.....	71
Figure 4.11 Plotlines Summarizing the Observed Network Structure in Simulation Experiment-6.....	73
Figure 4.12 Plotlines Summarizing the Observed Network Structure in Simulation Experiment-7.....	75
Figure 4.13 Plotlines Summarizing the Observed Network Structure in Simulation Experiment-8.....	77



Figure B.1 NetLogo Screenshot Illustrating Code of for A Simulation Model.....	142
Figure B.2 NetLogo Screenshot Illustrating Interface Tab of a Simulation Model.....	143

# Chapter 1 INTRODUCTION

## 1.1. Overview

In almost any Architecture, Engineering, and Construction (AEC) project, different expertise areas come together for a specific period to execute the delivery of a project that span programming, planning, design, and construction activities. Often the stakeholders involved are employed by different companies but are contractually bonded together for the delivery of a specific project with a defined scope keeping it within an estimated budget and schedule requirements. With so many preexisting constraints and a specified project goal to achieve, AEC projects call for exhaustive collaboration between, and integration of, different expertise areas. However, communication gaps between the involved stakeholders yield risks to effective collaboration. Such risk often results in information flow bottlenecks, inefficiency in decision making, reworks, and cost and schedule overruns. There is always a factor of cost related to this ineffectiveness that one or the other involved parties must bear. The situation sometimes aggravates so much that one or several of these involved parties, approaches the methods of arbitration or litigation for getting a compensation for the losses incurred (Sweet and Schneier, 2017). This, of course, involve even more time and expenses on each stakeholder's side. All of these successive unfavorable chains of events can be avoided to an extent by improving collaboration and communication dynamics among project team members. To improve collaboration between different stakeholders the first step is to comprehensively understand the structure in a project team communication network.

Social Network Analysis (SNA) shows promise in understanding such structures. Majority of the prior research, however, have only looked at AEC teams' social network structures in a cross-sectional manner. Recent similar studies claim that these networks are not static and can evolve

overtime (Garcia et al., 2020a and b). AEC project teams, thus, should be studied longitudinally using SNA. There is overwhelming assumption of work teams in AEC project context, hence, there is a need for investigating longitudinally, the evolution of AEC project team communication network structures during project delivery that stimulates the knowledge transfer. Nevertheless, collection of project team members' data longitudinally is an arduous task. Moreover, due to fast-paced delivery of any AEC project, researchers often end arrive at some missing data, sometimes depriving them from concluding statistically significant results in a longitudinal study. For this reason, an open-source agent-based modeling software is used to simulate and observe a social network structure evolution over time.

The goal of this study is to explore the evolution of network structures in AEC project teams based on team member/node characteristics (i.e., expertise, roles, and tiers). The study objective is to use an Agent-Based Model (ABM) approach to study the evolution of AEC project team networks and develop theoretical and practical implications. The unit of analysis in this research project is a network structure that appears through evolution in early stages of project delivery. To begin, researcher used NetLogo (Wilensky, 1999), an open-source agent-based modeling software to simulate partially a project team social network structure. The initial condition for simulations were based on an empirical case study of an AEC project team network studies by Garcia et al. (2020a and b). Later, the researcher analyzed the simulation results, both the intermediate and final sociograms of the evolved network structures. Finally, upon comparison with the empirical study, the results provide a better understanding to the AEC project networks phenomenon and node's characteristic variables such as expertise, roles, and tiers.

## 1.2. Need Statement

It is evident that lack in collaboration between AEC project teams ultimately foster loss in productivity. As discussed before it sets a start to unfavorable chains of events detrimental to the project progress. The project teams constantly thrive to maintain good relation and healthy communication among their team members so that everyone can work towards a common goal. In spite of their best efforts almost every AEC project face some amount of rework, schedule, and cost overruns. This is believed to happen because project leaders and members can never know for sure if the existing collaboration level among team members is good enough, other than just heuristically analyzing its potency. In other words, a system is required which can predict, before it is too late, if the collaboration level is optimal in a team or not. To thoroughly build such a system, however, one must first understand the communication network structure on which the collaborative effort thrives.

The use of SNA as a network analysis tool makes the most sense as it can access the dynamic-nature of the ever-evolving communication network structure of AEC project team (Garcia et al., 2020a and b). Although sparse, there are studies that used SNA to analyze AEC project team networks longitudinally analyzing the structures of such teams. However, collection of project team members' data longitudinally is an arduous task. Furthermore, due to fast paced AEC project deliveries, researchers often wind up with missing data, sometimes depriving them from concluding statistically significant results in a longitudinal study.

To respond to this gap in literature, there is a need to take an agent-based simulation approach to understand the multidimensional dynamic network structures that evolve to support knowledge sharing in AEC project teams.

### 1.3. Research Goal and Objectives

The goal of the research is to explore the evolution of network structures in AEC project teams based on team member/node characteristics (i.e., expertise, roles, and tiers). More specifically the study aims to shed light onto how real-world size AEC project networks evolve over project delivery based on similarity of node characteristics (e.g., homophily) and node behaviors in networks (e.g., node's popularity).

The objectives of the study are to:

- Review the literature and study empirical data to identify node characteristics key to project team networks and their structures;
- Using an Agent-Based Model (ABM) approach, study the evolution of AEC project team networks through a computer-based simulation; and
- Develop theoretical and practical implications.

To accomplish these objectives, NetLogo (Wilensky, 1999), an open-source agent-based modeling software, is used for creating the simulation experiments. The study adopted the selection model from the influence and section model (Frank and Fahrback, 1999; Frank and Xu, 2020), and simulated multiple agent-based models basing its initial condition on the observations in a recent study under AEC project team network structure domain (Garcia et al., 2020a and b).

The main research question that directed this research is as follows:

*“To what extent the similarity/difference in individual and dyadic characteristics influence the multidimensional dynamic network structures that evolve to support knowledge sharing in AEC project teams in early stages of project delivery? And what are the probable evolutions in these network structures overtime during project delivery?”*

## 1.4. Scope

The unit of analysis in this research project is network structure that appears through evolution in early stages of project delivery. The simulation experiments performed were bound by a recent study (Garcia et al., 2020a and b) that explained the network evolution in early phase of project delivery, thus the results of the simulation also hold representative of the same phase of AEC project delivery. The scope of the investigation was to simulate agents (project team members) with node characteristics and observe the results of interaction between these agents based on difference in their characteristics until when the network structure stabilizes. In all the simulation experiments the types of agents' characteristics were kept minimal to understand interactions better. In other words, any agent was either similar to another agent or not similar. The proportions of similarity among the agents, and the seeking need of connection to a new agent, were the parameters that were controlled by variables discussed in detail in the Chapter 2 and 3, the Literature review and Method sections respectively.

The network size studied were of two kinds: First, a relatively small network size for theory understanding and explanation; Second, a real-world AEC team project size, to predict the network evolution of such a network.

Qualitative analysis was carried out further to observe any trend in the evolution of network structure based on the variables like homophily/ seek for node characteristics (i.e., expertise, roles, and tiers) similarity, popularity seeking need of each node, and network size. Via comparison, the trends in the simulation experiment results are expected to signal the effective use of network structure in AEC project teams.

## 1.5. Overview of Methods

Extensive literature review on social networks, social network structure evolution simulation methods, and AEC network literature was conducted. The study then adopted the selection model from the influence and section model (Frank and Fahrback, 1999; Frank and Xu, 2020). Furthermore, basing the initial conditions for the simulation over Garcia et al. (2020a and b) several ABM simulations were run. The variables like homophily/ seek for similar node characteristics (i.e., expertise, roles, and tiers), popularity seeking need of each node, and size of the network were changed accordingly for each experiment to record the intermediate and final structure coming out of each simulation run. The results were then compared to find the trends that are expected to signal the effective use of network structure in AEC project teams.

To maintain research quality, several validation tests and reliability techniques were implemented on the ABM simulation experiments. The validation tests such as, event validity, extreme condition test, historical data validation, internal validity, parameter variability or sensitivity analysis, predictive validation and face validity were conducted (Sargent, 2013). The reliability techniques that were used include, open-source licensed distribution software usage, and model documentation with separate implementation techniques and conceptual description records (Richiardi et al., 2006). Details about each test and technique administered are as follows.

- Validation tests that were administered are: (Sargent, 2013)
  - Event validity, tested by comparing the occurrence of events within the simulation with real-world event from empirical case study (Garcia et al., 2020a and b);
  - Extreme condition test, administered by tuning all the variables to their extremities and comparing the results as to be valid. For instance, when there existed no

difference in node characteristics, for any value of homophily or node popularity the structure never broke;

- Historical data validation, conducted by using the simulation software and coding techniques from an existing similar model, influence and selection model (Frank and Fahrback, 1999; Frank and Xu, 2020);
  - Internal validity, examined by performing several replications (runs) of the stochastic model and observing consistency in results;
  - Parameter variability or sensitivity analysis, performed by changing the values of input and internal parameters (e.g., changing popularity definition from degree centrality to eigenvector centrality in the selection model equation) to determine if the results changed sufficiently;
  - Predictive validation, by forecasting the system behavior based on available theory and then comparing the actual behavior of the model; and
  - Face validity, by conducting expert interviews with Dr. Kenneth Frank and Dr. Ran Xu, co-authors of Frank and Xu (2020), about the simulation process and results, to check whether the model and/or its behavior are viable.
- Reliability techniques that were used are:
    - Use of open-source licensed distribution software, using NetLogo (Wilensky, 1999) for coding and simulating the ABM simulation experiments; and
    - Thorough documentation of each of the implementation steps for running the experiments (software interface and implementation knowledge) and conceptual model.



## 1.6. Results and Deliverables

The results and deliverables of the study are:

- 1) Verification of trends observed in an empirical study (Garcia et al., 2020a and b) showing that the network structures are dynamic, and they can evolve from core-periphery to cohesive subgroups in early stages of design during project delivery;
- 2) Practical applications for improved communication and management of AEC project networks;
- 3) Insights to variables that can have impacts on AEC network structures; and
- 4) Directions for future research.

## 1.7. Reader's Guide

In the following sections, Chapter-2 presents the extensive literature review in varied domains (e.g., construction management, organization science, computer science, and social networks) for defining the key methods for conducting the current research. Methodology for performing the ABM simulation-based experiment study is canvassed in Chapter-3. Chapter-4 lists the key findings from the simulation experiments. Chapter-5 presents a comparative analysis of the results, and enumerate their theoretical and practical applications in AEC project teams. It also concludes this dissertation emphasizing the most relevant points and offer recommendations for future research.

## Chapter 2 LITERATURE REVIEW

### 2.1. Introduction

This chapter discusses the concept of SNA, its key concepts, social network structures and observed presence of such structures in existing studies on project teams. With the aid of this literature review, the researcher attempts to substantiate the existence of dynamic nature of social network in AEC project teams, and then justifies the importance of studying these dynamics. This section also expatiates the findings of other prior studies that explored the non-static nature of communication network structure of AEC project during their delivery. It also presents the reason behind usage of agent-based modelling approach for the current research, and states several accomplished research that used a similar approach in myriad of domains (e.g., epidemiology, psychology, organizational research, and AEC). Following this, the researcher then, concludes the literature review section.

### 2.2. Overview of Social Network Analysis (SNA) and Key Concepts

SNA is a process by which one can investigate and analyze a network structure of a connected group or an organization through the use of network and graph theory (Otte & Rousseau, 2002). Here, sociogram is an information map of the network and is mentioned as being resilient yet constantly adapting in nature (Kadushin, 2012). Kadushin (2012) also propound that the initial applications of SNA were to investigate the working of an organization or workplace, and understand the characteristics of leadership in the organization. Due to the complexity involved and evolving nature of these workplaces, a social network graph is necessary to comprehensively capture and display the dynamics to any researcher analyzing the network over a timeline. Some of the basic definitions, concepts, and metrics of SNA are discussed below.

- **Graph (Network)** is the graphical representation of the Nodes and Edges. Also known as **sociogram**, a term coined by Jacob Moreno in 1953 (Kadushin, 2012).
- **Components of sociograms:**
  - **Nodes (vertices)** in a SNA graph represents people or subject whose relationship with other people or subject(s) is the interest of study (Otte & Rousseau, 2002).
  - **Edges (links)** in a SNA graph represents the relationship between the nodes (Otte & Rousseau, 2002).
  - **Hub** is a component of a network which has a node that have a high degree centrality (Barabási, 2016).
  - **Bridge** is a link between two nodes which if broken separates the network into two different components (Kereri and Harper, 2019).
- **Sub-network characteristics:**
  - **Cluster** is the grouping of actors within a network (Kereri and Harper, 2019).
  - **Dyad** represents the two node and the edge between them. Its study focuses on the dyadic properties which include reciprocity, flexibility, etc. (Kereri and Harper, 2019).
  - **Triad** represents a subgroup of three nodes and the possible edges among them (Kereri and Harper, 2019).
  - **Clique** is a subgraph containing three or more nodes and all the nodes are connected to one another (Kadushin, 2012).
  - A node's **neighborhood** is the set of its immediately connected nodes in the network.

- **Social network analysis metrics:**

- **Network density** is a measure of up to how much extent a network is connected, mathematically equal to the ratio of number of ties in the network over the total possible number of ties between all pairs of nodes (Kereri and Harper, 2019).
- **Cohesion (connectivity)** is the measure of the connectedness and togetherness among nodes within a network. Measures of cohesion includes reciprocity, network density, clique and structural equivalence (Lee et al., 2018).
- **Clustering coefficient** is a ratio, a numeric value between 0 and 1. It is mathematically equal to the number of closed triplets in a neighborhood of a node divided by the total number of triplets in the neighborhood (Kereri and Harper, 2019).
- **Closeness centrality** (analogous to mean) is the mean of all the shortest path distances from a node under consideration, to every other node in the network. It is a measure of speed with which an information from a node can reach to the whole network (Kereri and Harper, 2019).
- **Betweenness centrality** (analogous to median) is a measure of centrality of a node that quantifies the number of times the node can act as a bridge along the shortest path between two other nodes. (Freeman, 1977).
- **Degree centrality** (analogous to mode) is the total number of links going into or out of the node. It is a measure of node's influence or popularity (Kereri and Harper, 2019).
- **Eigenvector centrality** is another measure of influence of a node on its network. It assigns scores to all the nodes according to a rule that connection to high-scoring nodes has more contribution to the score of the considered node, than equal connections to low-scoring nodes (Newman, 2008).

- **Geodesic distance** is a measure of either the distance between the two nodes with the greatest separation in a network, or the distance between two nodes under consideration (Chinowsky et al., 2008).

### 2.3. Social Network Structures

- Understanding social network structures to perform SNA is crucial. The literature presented several types of social network structures based on information flow within these structures, randomness in structures, and their subgroup properties. The classification is presented hereafter:
- **Classification of social networks based on information flow notation:**
  - If existence of the link between any node A and another node B necessarily implies the existence of link from node B back to A then the network is called **undirected network** (Otte & Rousseau, 2002). The links do not show arrowhead between any connected nodes.
  - If existence of the link between any node A and other node B does not necessarily imply the existence of link from node B back to A then the network is called **directed network** (Otte & Rousseau, 2002). The links show arrowhead between any connected nodes for representing the direction of the information flow.
- **Classification of social networks based on randomness in the structure:**
  - **Linear graph (regular network)** represents a case in which nodes are linearly arranged, thus the average path length between two nodes is usually high (Golbeck, 2013).
  - **Random graph (random network)** represents a case in which each node (member) can access one another within the network and there is no clustering (Kereri and Harper,

- 2019). Mathematically it can be defined as placing  $N$  number of nodes and placing connections between them, such that the connecting tie of each pair has an independent probability  $p$  (Newman et al., 2008).
- A network is a **small world** if any two nodes (people) in the network can access each other through a small network path (Hexmoor, 2015). These networks exhibit high density within the clusters but also have bridging ties from cluster to cluster (Kereri and Harper, 2019).
  - **Classification of social networks based on subgraph properties:**
    - **Core-periphery structures** consist of two kinds of nodes. The cohesive subgroup that are closely connected to each other called core, and the set of nodes that are not connected to each other but are connected to one or a few nodes from the core called periphery (Borgatti and Everett, 2000). This type of network structure emphasizes team coordination (Garcia et al., 2020a and b).
      - A **bow-tie network structure** is essentially a directed core-periphery structured network that have a bow-tie configuration. It has a core at the center having low evolve-ability representing the highly constrained part of the network. The core is surrounded by fan-in component of incoming edges, and fan-out component of the outgoing ones. These fan-in and fan-out periphery components have fewer constraints and higher evolve-ability (Csermely et al. 2013).
      - **Rich club networks** are the ones with dense core nodes with peripheral nodes only connecting to the core nodes but not each other. Thus, they have a little nestedness (Csermely et al. 2013).

- **Nested networks**, however, are similar to rich clubs but also have their core nodes connected with each other. Here nestedness is increased but the peripheral nodes still remain disconnected with each other (Csermely et al. 2013).
- **Onion network structure** have their peripheral nodes connected to the core (as in rich club and nested networks) as well as other periphery nodes. This creates concentric layers of links resembling to the layers present in an onion, hence the name (Csermely et al. 2013).
- **Cohesive subgroups** are the sub network structures that consist of nodes that are densely connected to each other but sparsely to the nodes outside of the subnetwork. The optimal situation when all the subgroup members interact with all the other members within the subgroup the cohesive subgroup becomes a clique (Frank, 1995).
- **Egocentric network** is a personal network (Lee et al., 2018). The person in consideration is termed as **ego** and his/her connections are **alteri** (Djomba and Zaletel-Kragelj, 2016).

Building up on the definitions listed above the author explored the presence of such networks, and more importantly, the factors that encouraged the formation of these networks through the current simulation-based study.

## 2.4. AEC Project Teams and Social Network Analysis

This section discusses the SNA literature with a concentration to AEC project teams, specifically why SNA serves as an effective tool to analyze the network structure, and goes into details of coordination and communication giving evidence of how they are influenced by bridges and boundary spanners. Finally, the results and findings of some of the recent research that studied, longitudinally, an AEC project team using SNA tools and methods are presented.

### 2.4.1 Overview of SNA in the AEC Literature

Since the product of an AEC industry project is a result of the combined effort and implementation of knowledge base of the different expertise involved. It is evident that collaboration between the involved organizations and integration of different expertise areas are much needed aspect for an AEC project's successful completion. However, if there is a communication gap between these organizations, it can undermine the efficiency of information exchange between trades. Thus, there is a need of an analysis technique to track and examine the efficacy of communication between involved stakeholders. Literature suggests that SNA can serve as a technique to better record and plot these interactions and information exchanges using graph theory.

In AEC project management domain, SNA can be defined as a tool that can help in analyzing the interconnectivity and interdependence of project participant(s) in the iterative and interactive social structure (Lee et al., 2018). Findings of Lee et al. (2018) also suggests that SNA can prove to be a useful tool for improving the interdisciplinary interactions between involved project team members, thereby, enhancing project delivery efficiency. The study justifies this by performing a literature review and then classifying the applicable metrics that could be used to quantify each of the project management knowledge area such as, quality management, schedule management, resource management, etc. In a different study over an AEC competition project team, Chinowsky



et al. (2008) formulated a social network model for construction. This comprised of two segments, the dynamics, and the mechanics, that are stated to be essentially present in a team's network structure to achieve high performance. In this novel study Chinowsky et al. (2008) linked the team performance, in separate categories, with the SNA metrics measuring them. The model estimated the reason behind poor performances in specific areas with the help of defined metrics. This proves that not only can SNA interpret performance of AEC teams but can also help the team improve their weaknesses in time, which were otherwise, remained untapped. In a successive study by Chinowsky et al. (2010), four project teams that provided construction services were analyzed based on the prior model. Once again, the model illustrated that existing issues in trust and communication (parts of dynamic segment of the model) between project team members correlated with, the areas where the project failed to meet its objectives. Although substantive in producing and testing a social network model for construction teams, the research lacks to take into account other variables such as project delivery type, contractual relationship, etc., which may have an underlying effect on the team performance and the observed social network structure. Nonetheless, multiple research corroborated the importance of performing SNA on AEC teams, which is why SNA stays as an integral part of the knowledgebase that backs the current research.

#### 2.4.2 AEC Project Teams, Coordination, and Knowledge Transfer

In a comprehensive study conducted over a large electronic product manufacturing company product design divisions Hansen (1999) found that, within a network weak ties are good for searching information and strong ties are better option for complex knowledge transfer. Later in a successive research Hansen (2002) discovered that the amount of knowledge transfer was inversely proportional to the network path length connecting the considered entities. The connecting network path length, however, was directly proportional to the project completion time.

Reagans and McEvily (2003) further studied these findings and claimed that both, better social cohesion and increased network range, favors the ease of knowledge transfer. The three studies are compendious in terms of exploring the effects of cohesion, and weak and strong ties on knowledge transfer. However, there are other factors like, diverse background of the project team, and key organizational practices for managing these project teams, that can have an effect on the knowledge transfer, issue resolution, and ultimately the efficiency of the team. Di Marco et al. (2010) compared the evolution of social network structure of two culturally diverse team to study the effect of a bridger or a boundary spanner in such teams. Findings suggested that not only does a bridger helps in resolution of issues arising from cross cultural boundaries but also trigger the emergence of other bridgers, hence better integrating the team. In addition to the cultural boundary factor the organizational practices of cross functional teams also affect the coordination and knowledge transfer. A study by Laurent and Leicht (2019) asserts that cross functional teams should be formed at early stages of design with a focus on target value design approach. Moreover, the leadership should have a cross disciplinary knowledge so as to communicate and coordinate effectively.

Learning from the trends observed, the researcher studied weak and strong ties as the literature showed that each have serves a certain function in a social network better than the other. Moreover, special importance is given to understand the role of bridgers observed in the simulation experiments.

### 2.4.3 Overview of Longitudinal SNA Works of AEC Project Teams

Parraguez et al. (2015) conducted a longitudinal study over a complex engineering project for three-point interval, starting from conceptual design stage, going through detail design stage, and finally ending at system integration stage. It was observed that network was highly central at the conceptual design stage. The centrality decreased during the detailed design stage and then again rose to some extent in system integration stage. Network clustering values changed from low to high, and then to medium, however, during the three stages respectively. These research results did not provide any conclusion on the network structures observed however, it hinted that the social networks are not static, and evolution of network is task dependent. For instance, it is possible that it is the project's unique-goal-information transfer from the core leaders to the whole organization that led the network to become highly central in the concept design stage or, it is the highly decentralized departments doing their assigned job during detailed design that made the network structure look highly clustered during that phase. Parraguez et al. (2015) surely opens the ground for the aforementioned discussions.

A recent literature review by Kereri and Harper (2019) revealed that social networks of construction teams resemble the formation of a small world, evolving from the initially emerged clusters based upon professional, contractual, task, and trade relations between the team members. Another research in AEC project domain longitudinally studied the evolving nature of social networks over three time points during the schematic design phase of an AEC project (Garcia et al., 2020a and b). The study showed that during initial times the network forms a core-periphery structure and later due to core members' engagement with peripheral members the network evolves to form triangles and consecutively, cohesive subgroups. The research also found that the network evolution went hand in hand with the type of major task the network severed at different time

points. The result showed that core-periphery network structures indicated the presence of team coordination type of tasks and triangles and subgroups suggested towards the existence of deep knowledge sharing tasks (Garcia et al., 2020a). Kereri and Harper (2019) and Garcia et al. (2020a) were directly performed over AEC project teams and were thorough enough to divulge the relationship between task in consideration and the corresponding network structures.

## 2.5. Node and Dyad Characteristics in SNA and Evolution of Network Structures

Literature talks about several node and dyad characteristics that can influence a network structure. A study on AEC team project members by (Garcia et al., 2020a) presents many node level characteristics such as tiers, roles, expertise similarity, years working in AEC Industry, and meeting participation and contributions as influential conditions to communication network structure evolution. Although, due to missing data the study failed to conclude any statistically significant results with respect to correlation between network structure and node characteristic, qualitative analysis of the node characteristics (like expertise and role), complemented the observed structures. Lee et al. (2016) in a communication research over Korean immigrants' support exchange social network, found that the node characteristics like gender and age did not show any significant relation to the network structure observed. This finding overlapped with a study by Yuan and Gay (2006), that conclude gender and racial characteristics of a node to be insignificant as compared to social characteristics of a node in a study over computer mediated university student teams.

There are dyad level characteristics that are found significant in influencing the social network structure evolution. Yuan and Gay (2006) discovered that nodes having prior informal connections (i.e., some prior experience together), was a dyadic characteristic that significantly correlated to the probability of new work-related ties they could form. Furthermore, this particular characteristic

hints towards homophily that is also found significantly influential in several studies. Homophily, the observed tendency of people associating with like other, is mentioned as one of the most robust empirical regular phenomena in social life (Lazarsfeld and Merton, 1954). Yuan and Gay (2006) research concluded that the higher the number of common traits a couple of node has the higher is the likelihood of a work-related tie forming in between them. Not only was the homophily arising through common traits was found influential but also the homophily by location similarity was found correlated with the probability of ties between nodes. An inter organizational network study using exponential random graph models by Broekel and Hartog (2013) also found that homophily arising by having prior similar institutional background (shared values) is statistically significant in relation to network structure of the inter-organization communication. Thus, homophily was found significantly correlated to the probabilities of tie development between these organizations, i.e., the characteristic was significant at network level too.

Another important dyad level characteristic is a node's predilection towards popularity seeking. It is observed that nodes tend to connect with popular others in order to seek conformity to the running popular trends. In engineering and construction organization settings "conformity to corporate culture" accounts for 25.8% out of all the 9 factors considered, that motivates a project team member to share knowledge with their team members (Javernick-Will, 2012). The characteristic of popularity seeking is also studied and tested by Frank and Xu (2020) with the help of an influence and selection agent-based model simulation research. Through this research the author examined the effect of homophily and popularity seeking characteristics of nodes over the evolution of network structure overtime. It also took into account of the temporal influence a node creates on its connections.

The research parametrizes the relative effect of homophily in terms of alpha,  $\alpha$  ( $0 \leq \alpha \leq 1$ ) and trade it against popularity seeking urge of each node within the simulation model. The simulation model also uses an influence factor  $K$  ( $0 \leq k \leq 1$ ) that takes care of the influence any node puts on another node while interaction happens between nodes within the simulation model. The equations that represent the influence and selection part of the model are as follows:

$$U_{ijt} = (1-\alpha) p_{jt-1} - \alpha |y_{it-1} - y_{jt-1}| \quad (1)$$

$$y_{it} = (1-k\alpha) y_{it-1} + k\alpha \frac{\sum w_{ijt-1} y_{jt-1}}{\sum w_{ijt-1}} \quad (2)$$

Where,

$U_{ijt}$  is utility function for selection part of the model;

$\alpha$  is homophily factor, ( $0 \leq \alpha \leq 1$ );

$p_{jt-1}$  is the node  $j$ 's popularity;

$t$  is time;

$|y_{it-1} - y_{jt-1}|$  is the difference in node characteristics of node  $i$  and  $j$  at time  $t-1$ ;

$y_{it}$  is the newly influenced characteristic of a node  $i$  at time  $t$ ; and

$K$  is influence factor, ( $0 \leq k \leq 1$ ).

In equation 1,  $U_{ijt}$  is the utility function for selection part of model. It is the utility of any node  $i$  wanting or selecting to make a connection with a node  $j$  at any time  $t$ .  $p_{jt-1}$  is the popularity (degree centrality) of the node  $j$  at time  $t-1$ .  $\alpha$  is the homophily seeking factor and  $|y_{it-1} - y_{jt-1}|$  is the difference in node characteristics of node  $i$  and  $j$  at time  $t-1$ .

In equation 2,  $y_{it}$  is the newly influenced characteristic of a node  $i$  at time  $t$ . It is based on the resultant sum of its initial node characteristic  $y_{it-1}$  at time  $t-1$  and the influence the node  $j$  had on

node  $i$  from time  $t-1$  to time  $t$ , represented by the quantity  $k\alpha \frac{\sum w_{ijt-1} y_{jt-1}}{\sum w_{ijt-1}}$ . Here  $k$  ( $0 \leq k \leq 1$ ) is

the influence factor, i.e., the factor of influence that can be transferred between any two nodes.

These factors were added at node level (micro level) and the simulation results presented the network level (macro level) evolution that were triggered.

The network structure evolution resultant graphs from the simulation were found to be analogous to the results observed in AEC project team social network structure evolution discussed earlier in the literature review. For instance, when  $\alpha$  (homophily) and  $k$  (influence) are high the structure breaks into cohesive subgroups (Frank and Xu, 2020). This seems analogous to the later stages in AEC project delivery when after project coordination is successfully accomplished similar trade members comes together to use the project info and produce results forming cohesive subgroups (Kereri and Harper, 2019; Garcia et al., 2020a and b). Similarly, when  $\alpha$  (homophily) is low, irrespective of  $K$  (influence), the structure forms a core-periphery (Frank and Xu, 2020), analogous to the early stages in AEC project delivery when project coordination is in progress and core members of team are providing crucial information to different trade members producing a core-periphery structure (Kereri and Harper, 2019; Garcia et al., 2020a and b). This similarity seeds the need of the author's current research.

However, the use of degree centrality to translate popularity (node's importance) into the simulation might end up not providing actual results from the simulation. This is because degree centrality just takes into account of the number of outgoing or incoming links to a node which sure means that the node is important, but the notion might miss the nodes that are not having as many

connections but are connected to a few but important nodes. For instance, a node could be a bridging node between two highly degree central nodes. In this case if popularity is equated to degree centrality the simulation calculation will miss the most important bridging node that have relatively low degree centrality but high importance. Now, as defined in the literature earlier eigenvector centrality assigns scores to all the nodes according to a rule, that connection to high-scoring nodes has more contribution to the score of the considered node, than equal connections to low-scoring nodes (Newman, 2008). This being a relative measure takes into account of nearly all the possibilities of a node being popular in reality and just not a node with the greatest number of unimportant connections. Thus, as the literature suggests, eigenvector centrality measure was used in place of degree centrality, for translating the notion of “popular node” into some of the simulation experiments, to observe if it provides more realistic results.

Moreover, literature suggests that in context of project teams in AEC industry homophily should be approached differently versus the project networks in other domains. For instance, in influence and selection model at any instance a node connection meant a node characteristic is going to change the other connected node characteristic. But in AEC domain a node’s characteristics like expertise or role does not change the connected node’s expertise or role. In other work an architect connecting with an owner does not make the owner an architect no matter the strength of the connection or time for which it influences the connected team participant. For this reason, only selection model was considered for the current research simulation models.

Another dissimilarity in the context of an AEC project team and other organizational teams is that, often times heterophilic nodes (nodes having dissimilar characteristics) have to connect with each other in order to attain a common goal. In other words, two nodes having dissimilar node characteristics sometimes seek to connect with each other to achieve project success. This may be



a dissimilarity based of tiers (a tier 3, a project engineer wanting to connect with a tier 2, senior construction manager for gaining information), or role (an architect wanting to connect with a contractor for change order removal), or expertise (a plumber wanting to connect with an electrician for rerouting of MEP items) or a combination of these all. The testing of this phenomenon has never been performed in the literature. Thus, for this research experiments also considered trading heterophily versus popularity seeking, the two characteristics that seems to drive social network structure evolution in AEC teams.

## 2.6. ABM, AEC Research Uses, and SNA Integration

Among all the longitudinal case studies on AEC project social network discussed above there is a commonality, the arduous task of data collection. First, it is usual with AEC project teams to constantly have different parties working, joining, and leaving the project as it progresses. Second, unlike any other industry AEC projects are very fast paced which brings together often unacquainted people from different trades to work together towards a common goal for a short intense period. It is due to these two aforementioned reasons the collection of data from the project team parties often encounters a problem of missing data. This sometimes even incapacitate the capability of an investigator to present a statistically significant result after putting in long hours first collecting and then analyzing the data. Financially expensive, time involving, and physically arduous, the collection of project team members' social interaction data longitudinally could be replaced, to some extent, by taking an Agent-Based Model (ABM) simulation experiment-based approach.

ABM origin lies in the early studies in cellular automata (Wolfram and Mallinckrodt, 1995) and artificial life (Langton, 1995). Currently it is widely used in organizational research (Miller and Lin, 2010; Burton and Obel, 2011). An ABM computer simulation overcomes some of these

challenges and provides a researcher with the benefit of replicating the important characteristics of a human behavior in a computer-generated reality (simulation). Thus, instead of collection of project team members' data longitudinally only the initial condition relationship is plotted in the simulation. The agents (analogy of people in the simulation world) are left to interact with each other based on variable values of agents' characteristics.

In essence, the most integral unit of any ABM is an agent. ABM is a model in which a real-world occurrence can be modeled that comprise of a system that is dynamic in nature, i.e., several actors within a system continually interact with each other to produce different results (Castiglione, 2006). In ABM the agents act as autonomous entities and based on set of rules these actors make decisions accordingly in order to replicate a real-life scenario (Sawhney et al., 2003). The state of an agent is defined by its attributes. With change in attributes the behavior of the agents' changes as well. Thus, ensuing a change at macro level when the agent interacts with each other in a simulation. Hence, by using aa ABM a researcher can analyze a macro level patter emerging out from agents' behavior and interactions. Since ABM can take into account of large number of agent and complex system (such as a social network structure) with rules to predict any event very quickly as compared to the time-interval the natural phenomenon takes to occur, it was deemed fit for the use in the current research. Moreover, it is thought be the core reason why research in AEC domain have started using ABM to emulate and analyze complex organizational environment within a simulation model.

A compendious research by (Ding et al., 2016), that dwelled into the problem of managing AEC industry demolition waste, used an ABM approach. A longitudinal trend of demolition waste quantities was forecasted based on an ABM simulation where interactions between demolition agents (e.g., demolition companies, waste management companies, waste disposal companies,

etc.) were tracked and simulated overtime. The observed trends would help keep these organizations (demolition agents) a check on their work before it is too late. This means that the use of ABM in predicting longitudinal trends are proved to be helpful with respect to the amount of time saving that can be performed. That is, simulating the whole process in place of collecting the data longitudinally is a huge success in itself. Another comprehensive research used ABM to understand the impact of lack or absence of instructions on production performance in an AEC industry workplace (Lahouti, 2013). The agents in this model were coded such that they make decisions based on the cues the agents' surroundings provide them, when there is a lack of instruction to complete a job. This hints that ABM can be used to represent complex decision making behavior of human in a simulation. A different study, again, successfully forecasted the demand for recycled mineral construction material by creating an ABM that emulate the decision making process of the actors involved (Knöri et al., 2007). Furthermore, Raoufi and Robinson Fayek (2018) support the argument of ABM suitable for handling human behaviors in simulations. Their study replicated construction crew decision making behavior. All in all, the discussed ABM studies (Knöri et al., 2007; Lahouti, 2013; Ding et al., 2016; Raoufi and Robinson Fayek, 2018) justifies the use of ABM as a modelling technique for the current research.

Since the emergence of agent-based modelling approach, which dates back to the early studies in cellular automata (Wolfram and Mallinckrodt, 1995), it has also been extensively used in many research that studied social interaction phenomenon by coupling ABM with SNA (Will et al., 2020). The use of ABM for SNA can explain complex human interactions with so much ease that its application in research ranges through different fields of study such as epidemiological (Frias-Martinez et al., 2011), psychology (Frank and Fahrback, 1999; Frank and Xu, 2020), organizational research (Lin, 2014), and AEC domain (Ding et al., 2016).

Computing advancement has ubiquitously enabled capturing large amounts of human behavior data. The footprints from these datasets if analyzed correctly can provide incisive knowledge about virus spreading. Thus, Frias-Martinez et al. (2011) used an ABM coupled with social network information of people communicating over phone calls to study 2009 H1N1 flu outbreak patterns. It was found that the limitation of studying the spatial-temporal human behavior dynamics were overcome by using the ABM that mimics population social patterns.

The use of ABM in explaining polarization in social communication network is comprehensively explained in Frank and Xu, (2020) influence and selection ABM model, that, simulates social networks to study the phenomenon in temporal manner. The study was not only able to complement the reason for the polarization in current U.S. voting population, but also, support the evolutionary grouping pattern observed in cohesive groups of terrorist organizations such as Red Army terrorist groups, Al Qaeda, and Islamic state (ISIS).

Another research conducted by Lin (2014) used ABM that looked into mechanism that strengthen or weaken a closure and brokerage structures of organizational teams. The study mapped out the possible evolutionary path of the network based on a discovery of a static network. Basing the ABM of the social network over an empirical study is very similar to the author's current research-approach, where an ABM initial condition and various parameters were set over an empirical study.

The pervasive coupled use of ABM and SNA have led researchers to develop realistic initial condition models for simulation use purposes (Hamill and Gilbert, 2009). The reason for skepticism among some social scientists about ABM, and for the low acceptance of results that comes out of an agent-based methodology is often the unavailability of methodological standard references. Richiardi et al. (2006) have worked at producing a very well written protocol that can

be used to maintain research quality while using ABM to conduct research on complex phenomenon like analyzing social communication pattern.

The literature provides various instances of ABM and SNA that supports the use of ABM usage for current research. Accessing the current research goal of exploring the evolution of network structures in AEC project teams based on team member/node characteristics using an ABM approach shows promise in annexing valuable insights to the body of knowledge.

## 2.7. Summary

Through this chapter the author discussed the overview, key concepts, and metrics of SNA. Later the literature study provided enough evidence to substantiate the importance of performing an SNA on AEC project teams. Although rare, the researcher found longitudinal SNA studies performed over the years. The review, therefore, enumerate the important findings and trends observed in these studies performed over both non-AEC and AEC industry projects. The review revealed that although AEC projects should be studied longitudinally for SNA, due to the intrinsic problem of arduous data collection and missing data instances, adopting an ABM computer simulation methodology could be a better option to virtually analyze a social network evolution overtime.

Studies also suggest the use of ABM technique to be a better fit for replicating a complex real-life system, be it AEC or non-AEC project team communication network structure. This is due to the capacity of ABM to consider a huge number of autonomous entities and produce results of their interactions almost instantaneously, saving time, energy, and money that would have been otherwise consumed to collect the data over the years to come to a conclusion.

The literature then discusses studies that used ABM in research from various domains and explains the simulation model from one of them in detail. The part of the simulation model that inspired the current research are then expatiated (Frank and Fahrbag, 1999; Frank and Xu, 2020). The simulation model decision making process for the current investigation are discussed later in the Chapter-3 and 4, Methodology and the Results sections respectively.

## Chapter 3 METHODOLOGY

### 3.1. Introduction

To response to the needs that were laid out in the literature review the study aims to elucidate, how real-world size AEC project networks evolve over project delivery based on similarity of node characteristics (e.g., homophily) and node behaviors in networks (e.g., node's popularity). To do so, the study adopted selection model (Frank and Fahrback, 1999; Frank and Xu, 2020), and simulated multiple agent-based models basing its initial condition on an empirical study (Garcia et al., 2020a and b). The evolutions of the network structures were analyzed overtime. The study then compared its findings with the empirical study. The current research's methods for the simulation experiments and their results analysis are discussed in the subsections hereafter.

### 3.2. Research Goals and Objectives

The goal of the study is the exploration of the evolution of network structures of AEC project teams based on team member/node (i.e., expertise, roles, and tiers) and dyadic (homophily, heterophily, and popularity seeking) characteristics. More specifically the study aims to interpret how real-world size AEC project networks evolve over project delivery based on similarity of node characteristics (e.g., homophily) and node behaviors in networks (e.g., node's popularity).

The main objectives of the study are to:

- Review the literature and study empirical data to identify node characteristics key to project team networks and their structures;
- Using an Agent-Based Model (ABM) approach, study the evolution of AEC project team networks through a computer-based simulation; and
- Develop theoretical and practical implications.

To accomplish these objectives, NetLogo, an open-source agent-based modeling software, is used for creating the simulation experiments. The study adopted the selection model from the influence and section model presented in a study by Frank and Xu (2020), and simulated multiple agent-based models basing its initial condition on an empirical case study data as initial condition anchorage (Garcia et al., 2020a and b).

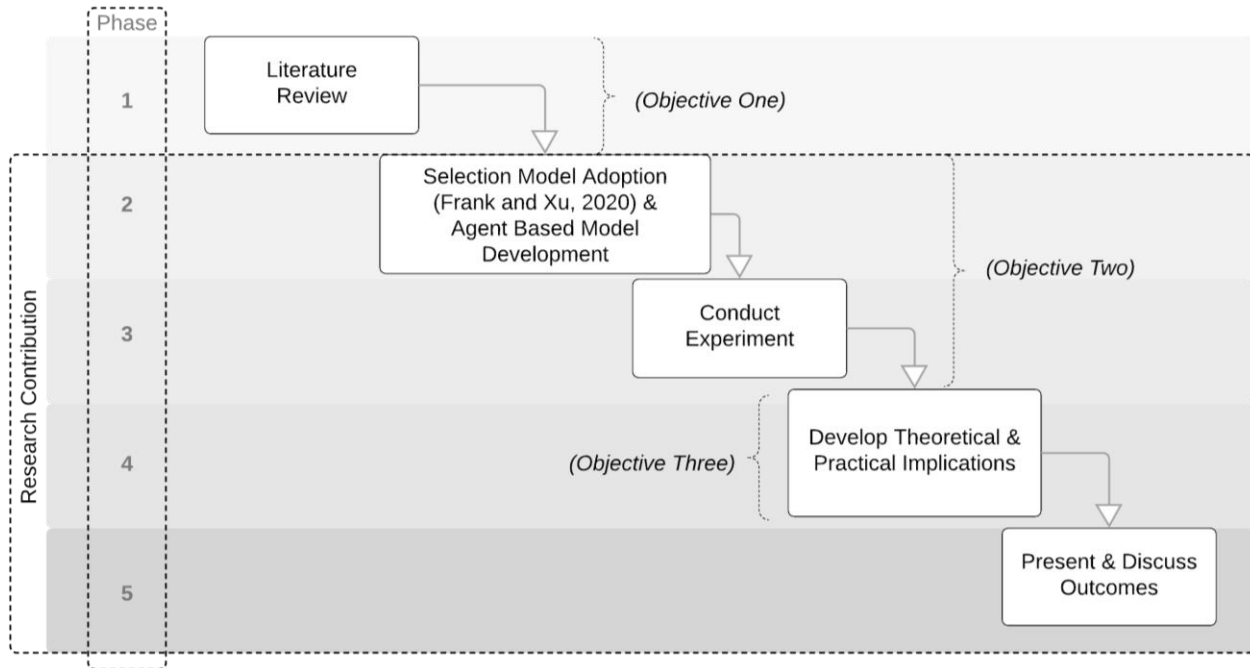
The main research question that directed this research is as follows:

*“To what extent the similarity/difference in individual and dyadic characteristics influence the multidimensional dynamic network structures that evolve to support knowledge sharing in AEC project teams in early stages of project delivery? And what are the probable evolutions in these network structures overtime during project delivery?”*



### 3.3. Research Approach and Scope

This explanatory and empirical research was categorized into five phases, as illustrated in the Figure 3.1.



**Figure 3.1** Investigation Framework

Figure 3.1 shows the five phases contributing to the three research objects mentioned above. Phases two, three, four, and five accounts for the contribution the study annexed to the body of knowledge.

The unit of analysis in this research project is network structure that appears through evolution in early stages of project delivery. The simulation experiments performed had initial conditions anchorage mimics the network evolution in early phase of project delivery, thus the results of the simulation also hold representative of the same phase of AEC project delivery. The scope of the investigation was to simulate agents (project team members) with their characteristics and observe

the results of interaction between these agents on the resultant communication network structure produced until when the network structure stabilizes.

### 3.4. Model Description and Simulations

#### 3.4.1 Selection Model for Homophily/Heterophily and Popularity Seeking

As explained in the literature review section, the current research adopted from the selection and influence (Frank and Fahrback, 1999; Frank and Xu 2020). Moreover, as approach towards node characteristics like tiers, roles, expertise similarity, years working in AEC Industry (Garcia et al., 2020a), homophily (Yuan and Gay, 2006; Lee et al., 2016) and popularity seeking differs in context of an AEC project team than that of other project teams. Thus, parts of the original equation 1 ( $U_{ijt} = (1-\alpha) p_{jt-1} - \alpha |y_{it-1} - y_{jt-1}|$ ) were modified for the some of the experiments accordingly. In total eight experiments were run, with all the odd numbered experiment (1,3,5, and 7) run on the small sized networks, and all even numbered experiments (2,4,6, and 8) run on the real-world sized networks. The parts of the original equations 1 and 2 that were partially modified or discarded are discussed hereafter.

Among the original equation 1 and 2, the second equation was responsible for the influence part of the model and in AEC team members social network context this part of the model is not considered a part of the current study simulation model for the reasons explained in the literature review section.

In the first equation ( $U_{ijt} = (1-\alpha) p_{jt-1} - \alpha |y_{it-1} - y_{jt-1}|$ ),  $U_{ijt}$  is the utility function for selection part of model. It is the utility of any node  $i$  wanting or selecting to make a connection with a node  $j$  at any time  $t$ .  $p_{jt-1}$  is the popularity (degree centrality) of the node  $j$  at time  $t-1$ .  $\alpha$  is the homophily seeking factor and  $|y_{it-1} - y_{jt-1}|$  is the difference in node characteristics of node  $i$  and  $j$  at time  $t-1$ . For

Experiment one and two the selection equation was taken into the new model as it is. Note, however, the homophily characteristic in the model can represent homophily seeking via any type of node/dyad characteristic (e.g., tiers, roles, and expertise). For Experiment three and four the term  $p_{jt-1}$ , representing the popularity of a node via degree centrality, was changed to represent popularity of any node  $j$  with its eigenvector centrality. This was done to observe, based on the findings from the literature review, if the simulation provides more realistic results using eigenvector centrality over degree centrality whenever defining popularity of a node in an AEC project team communication network.

Furthermore, in experiment five and six, the author wanted to check whether an observed phenomenon of node selecting heterophilious other (selecting node with dissimilar node characteristics) was significant to correlate with evolution of social network structure in AEC team when traded against node's popularity seeking characteristic. To conduct such an experiment first, the equation 1 was modified to:

$$U_{ijt} = (1-\alpha) p_{jt-1} + \alpha |y_{it-1} - y_{jt-1}| \quad (3)$$

Here the negative sign between the original equation 1 is replaced with a positive sign. This basically translated what the author wanted the experiment to perform, into a mathematical expression for the simulation model. Therefore, experiment five and six used the modified equation 3 for its simulation runs. Note, here the term  $p_{jt-1}$  was still representing node's popularity seeking via degree centrality. To compare the simulation results with an empirical study results the author again changed the term  $p_{jt-1}$  to represent node's popularity seeking via eigenvector centrality. This was then used to run the experiment seven and eight.

Moreover, within each experiment (one to eight) the population of node had two types of characteristics. The ratio of nodes having one type of characteristic versus the ones having the other type was represented in terms of domain difference percentage form here after in this writing. In other words, if the population of 20 nodes in a simulation consists of 10 having a type of characteristic and other 10 having another type of characteristic, then the domain difference percentage is mentioned as 50%. If this population changes to 5 nodes of one type and 15 nodes of another type, then the domain difference percentage is mentioned as 25%. Thus, by the logic if all nodes have same characteristic then the domain difference percentage is 0%.

The NetLogo coding for the production of all the simulation model was based upon the codes of open-sourced materials, “starting from divergence.nlogo” & “random network homophily vs preferential attachment.nlogo” models (Frank, 2020). There are various procedures and codes that enabled production of all the discussed experiments. All these coding and the exact procedure for all the experiments conducted for this study are presented in the APPENDIX A section.

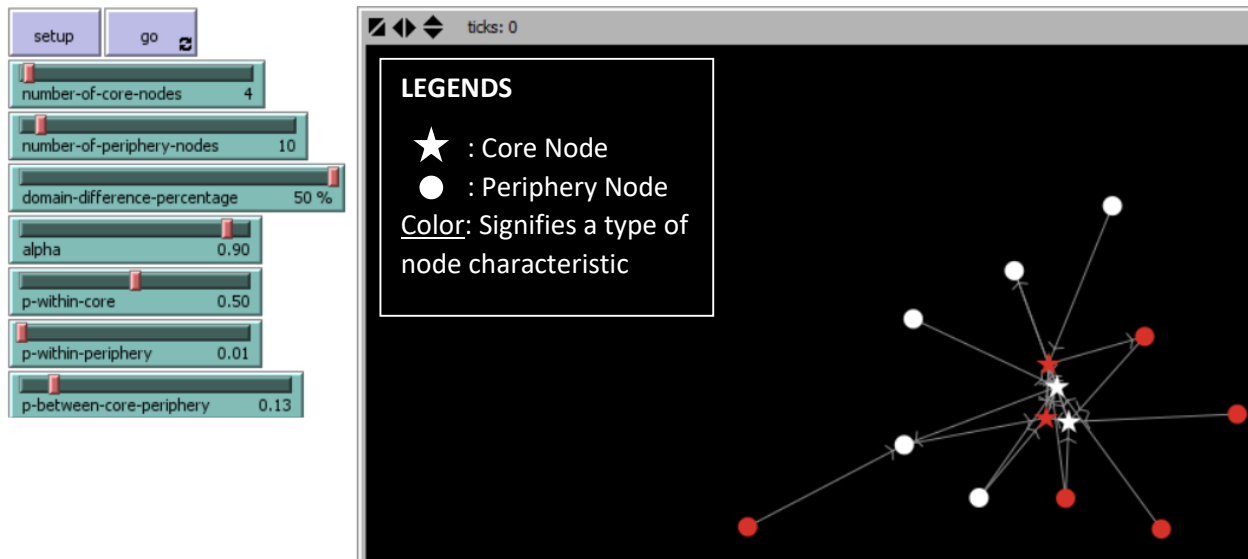
### 3.4.2 ABM Conditions and Steps to Run the Simulation

**Netlogo:** Initially developed by Wilensky (1999), Netlogo is a programmable and flexible ABM modeling environment and an open-source licensed software used for simulating phenomena that are natural and social.

Coders provide instructions to hundreds or thousands of "agents" all operating independently though the use of code tab. This feature of the program makes it possible for it to be used in the exploration of the connection between the micro-level behavior of an agent and the macro-level patterns that emerge from their interaction. The set of rules defined by the coder acts as a brain and guides each agent to make learned actions at any time in a running simulation. All such actions

are updated and displayed in real time in the “Interface” tab of the NetLogo simulation model. Moreover, the interface tab of the software in itself is a totally customizable area, with all the required buttons such as slider bars, switches, plots, text area, and simulation world screen, etc. User can utilize customizable interface to observe a part of the simulation result, or to give instructions, or change attribute values of the agent interacting in the simulation. APPENDIX B mentions two out of three tabs and all the interface items, useful to understand the simulation model created for the current research.

**Initial Experiment Conditions:** There are three network densities set up based on the empirical case study considered (Garcia et al., 2020a and b) and remain unvaried throughout all the experiment setups. “p-within-core,” “p-within-periphery,” and the “p-between-core-periphery,” the three densities, defines the network tie density among core nodes, peripheral nodes, and between core and periphery nodes, respectively (Figure 3.2). There are two variants of initial condition that are setup for the experiments. In the first type of initial setup, the initial condition is set to replicate a small-scale core-periphery network structure (4 core nodes and 10 periphery nodes). This is illustrated in the Figure 3.2.

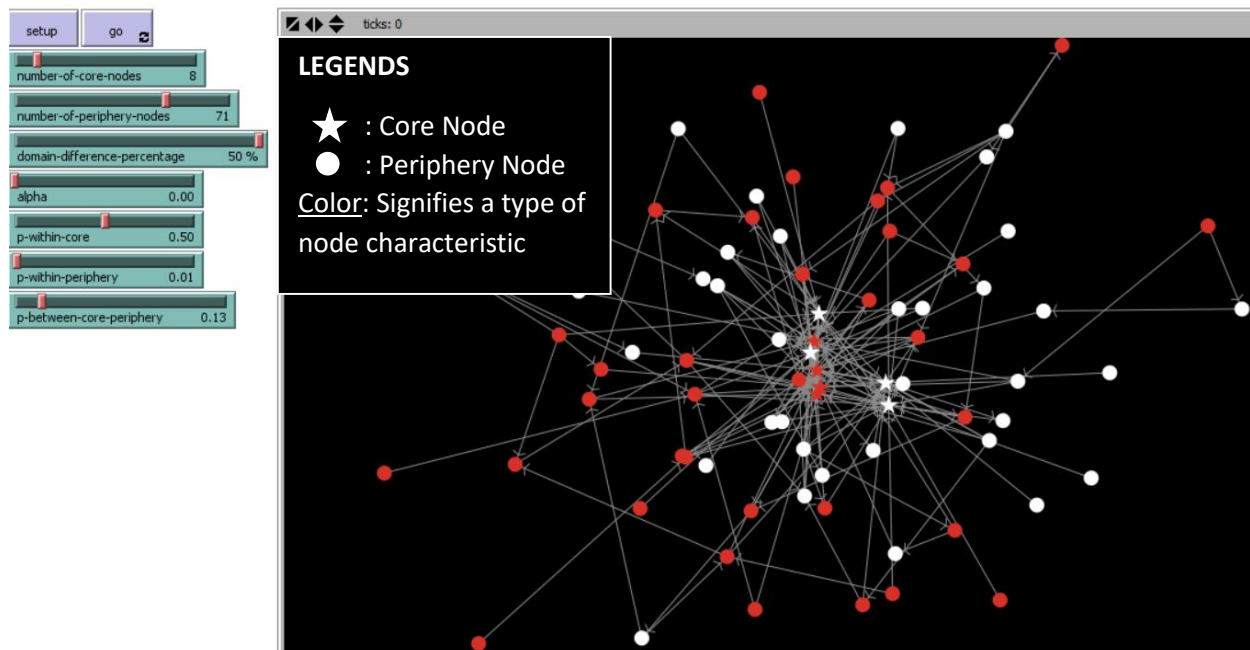


**Figure 3.2** Type-1 Initial Setup: a NetLogo Screenshot

In Figure 3.2 the purple-colored button on the top left corner “setup” sets up this initial condition based on the slider button inputs. These 7 green colored slider buttons present on the left side in the figure are set before running the setup button. The “number-of-core-nodes” button, as the name suggests, sets up the number of core node the initial condition of the network simulation will constitute of. Similarly, the “number-of-periphery-nodes” button sets up the number of periphery node the initial condition will have. The “domain-difference-percentage” button sets up the ratio of the nodes that have difference in their node characteristics. The color of the nodes in the simulation world (black screen) defines the two different node characteristic types. The shape of the nodes being star and circle defines that they are core and periphery nodes respectively. For this specific screenshot the domain-difference-percentage button is tuned up at 50%. This implies that 50% of the nodes have one characteristic and other 50% have the other. This is exactly what the color difference in nodes also depicts, i.e., 50% nodes are red in color and 50% nodes are white in color. This difference can be interpreted as difference in any kind of node characteristic (i.e., expertise, roles, and tiers). The “alpha” slider defines the homophily/heterophily factor and only

starts playing its role after the simulation starts, i.e., when the purple colored “go” button right of the setup button is clicked.

In the second type of initial setup, the initial condition is assumed to be at an early stage of project delivery of an AEC project with team communication network structure forming a core-periphery network structure with 8 core and 71 peripheral nodes based on the empirical case study considered. The only difference between this setup and the earlier one is that in this a real-world sized network is considered to observe its evolution overtime. This setup is illustrated in the Figure 3.3.

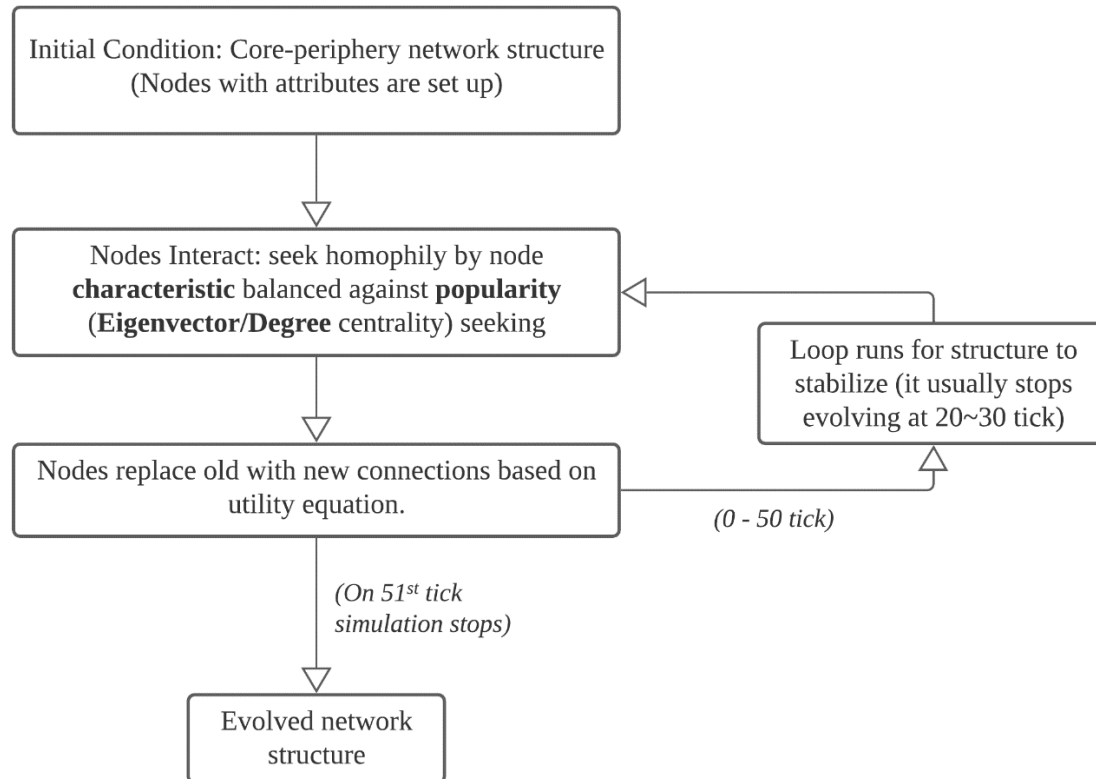


**Figure 3.3** Type-2 Initial Setup: a NetLogo Screenshot

In the Figure 3.3, the setup contains all the buttons and switches explained in the setup type-1. Only the “number-of-core-nodes” and “number-of-periphery-nodes” sliders are tuned on greater numbers, i.e., at 8 and 71 respectively. The black colored simulation world screen shows how the

network structure looks like with the set type-2 initial conditions. Just like type-1 initial setup, type-2 setup also starts the simulation run as soon as the “go” button is clicked.

The decision-making process that governs each agent in the current research simulations is shown in the Figure 3.5.



**Figure 3.4** Diagram Illustrating Agent’s Decision-Making Process

As shown in the Figure 3.5, for each experiment the initial condition comprises of a core-periphery network structure, with a variant in the network size: small sized network and real-world sized network. Each node has an attribute such that two nodes interacting in the simulation either have a same attribute or a different in attribute, coded as 0 and 1 respectively. As soon as the simulation start, the tick value becomes 1 and each node start to choose either popular other nodes having different or same attribute or strictly connects to a node with similar attribute. This selection is



based on a number of factors like utility value for connection between a node  $i$  and node  $j$  at any time  $t$  ( $U_{ijt}$ ), alpha ( $\alpha$ ) (Frank and Xu, 2020), and the probability induced by domain difference percentage.

Since this model represents a phenomenon that evolve overtime, the agents' decision-making process are set such that they interact repeatedly following a set rule. In NetLogo program such a cycle is called a "Tick". As the simulation model runs the tick numbers keep on increasing with an increment of 1. In other words, if the tick is 25, this means that the simulation is in its 25th cycle of the simulation run. Although, the initial runs of the current research simulation experiments provided a stabilized (not evolving any further) structures after running 20~30 ticks (loops), as per the literature, the simulation was run for 50 ticks for each experiment.

The network structure at the end of the simulation as well as the intermediate network structures are recorded and qualitatively analyzed.

### 3.5. Research Quality

To maintain research quality, several validation tests and reliability techniques were implemented on the ABM simulation experiments. The validation tests such as, event validity, extreme condition test, historical data validation, internal validity, parameter variability or sensitivity analysis, predictive validation and face validity were conducted (Sargent, 2013). The reliability techniques that were used include, open-source licensed distribution software usage, and model documentation with separate implementation techniques and conceptual description records (Richiardi et al., 2006). Details about each test and technique administered are as follows.

- Validation tests that were administered are: (Sargent, 2013)

- Event validity, tested by comparing the occurrence of events within the simulation with real-world event from empirical case study (Garcia et al., 2020a and b);
  - Extreme condition test, administered by tuning all the variables to their extremities and comparing the results as to be valid. For instance, when there existed no difference in node characteristics, for any value of homophily or node popularity the structure never broke;
  - Historical data validation, conducted by using the simulation software and coding techniques from an existing similar model, influence and selection model (Frank and Xu, 2020);
  - Internal validity, examined by performing several replications (runs) of the stochastic model and observing consistency in results;
  - Parameter variability or sensitivity analysis, performed by changing the values of input and internal parameters (e.g., changing popularity definition from degree centrality to eigenvector centrality in the selection model equation) to determine if the results changed sufficiently;
  - Predictive validation, by forecasting the system behavior based on available theory and then comparing the actual behavior of the model; and
  - Face validity, by conducting expert interviews with Dr. Kenneth Frank and Dr. Ran Xu, co-authors of Frank and Xu (2020), about the simulation process and results, to check whether the model and/or its behavior are viable.
- Reliability techniques that were used are:
    - Use of open-source licensed distribution software, using NetLogo (Wilensky, 1999) for coding and simulating the ABM simulation experiments; and

- Thorough documentation of each of the implementation steps for running the experiments (software interface and implementation knowledge) and conceptual model.

### 3.6. Summary

Through this section the author presents the goal, objectives, approach, and scope of the research. The researcher explains the five phases in which this research is divided in, namely, literature review, selection model adoption and ABM development, conduct experiment, develop theoretical and practical implications, and present and discuss outcomes. With a detailed process diagram the agent-based decision-making process developed for this research is then expatiated. The initial condition setups for all the simulation experiments are explained. With a detailed process diagram the agent-based decision-making process developed for this research is then expatiated. Finally, to maintain research quality several validation test and reliability techniques that were implemented on the ABM simulation experiments, were also enumerated.

## Chapter 4 RESULTS

### 4.1 Introduction

To present both theory explanation, and prediction of a realistically sized AEC project social network structure evolution, simulations were run on two different types of initial condition setups as explained in Chapter-3 Methodology. First, to track and analyze the effects of node characteristics like homophily/heterophily and popularity seeking on a network structure, a small sized core-periphery network structure was set as initial condition for the experiment. Later a real-world sized network was used to see how the structure evolve from the influence generated by the same characteristics.

### 4.1 Homophily/Heterophily versus Popularity (Degree / Eigenvector Centrality)

As shown in the Figure 4.1, eight experiments were run:

1. Homophily via node characteristics (e.g., tiers, roles, expertise similarity, years working in AEC Industry) and nodes' popularity via degree centrality for small networks;
2. Homophily via node characteristics against nodes' popularity via degree centrality for real-world sized networks;
3. Homophily via node characteristics and nodes' popularity eigenvector centrality for small networks;
4. Homophily via node characteristics and nodes' popularity eigenvector centrality for real-worlds sized networks;
5. Heterophily via node characteristics and nodes' popularity via degree centrality for small networks;

6. Heterophily via node characteristics and nodes' popularity via degree centrality for real-worlds sized networks;
7. Heterophily via node characteristics and nodes' popularity via eigenvector centrality for small networks; and
8. Heterophily via node characteristics and nodes' popularity via eigenvector centrality for real-worlds sized networks;

	<div>Popularity seeking via degree centrality <b>VS</b> Homophily via node characteristics similarity</div>	<div>Popularity seeking via eigenvector centrality <b>VS</b> Homophily via node characteristics similarity</div>
<div>Small sized core-periphery network structure</div>	<div>Exp. 1</div>	<div>Exp. 3</div>
<div>Real world sized core-periphery network structure</div>	<div>Exp. 2</div>	<div>Exp. 4</div>
	<div>Popularity seeking via degree centrality <b>VS</b> Heterophily via node characteristics dissimilarity</div>	<div>Popularity seeking via eigenvector centrality <b>VS</b> Heterophily via node characteristics dissimilarity</div>
<div>Small sized core-periphery network structure</div>	<div>Exp. 5</div>	<div>Exp. 7</div>
<div>Real world sized core-periphery network structure</div>	<div>Exp. 6</div>	<div>Exp. 8</div>

**Figure 4.1** Matrix Explaining all Eight Types of Simulation Experiments for the Research

Figure 4.1 illustrates a matrix showing what each experiment simulation had as a condition for agents to act upon. All the observations and results from each of the experiment mentioned in the figure are presented in detail in the upcoming sub-sections.

As soon as the simulation setup is complete based on type 1 or type 2 initial condition a core periphery network structure appears (Refer Figure 3.2 and 3.3). Then, upon running the simulation, the nodes start interacting with each other based on the following steps:

1. Based on the computation of all the node's utility of tie establishment, each node connects to the node having highest of this computed utility value using the following equation adopted from Frank and Xu (2020).

$$\text{Utility}_{ij} = (1 - \alpha) * \text{Popularity}_j - \alpha * \text{Difference in node characteristic}_{ij}$$

Here, Popularity of node j is degree centrality of node j. This first part of the equation  $((1 - \alpha) * \text{Popularity}_j)$  is responsible for the nodes behavior of seeking new knowledge from popular other. Difference in node characteristic of node i and node j give either 0 (meaning the nodes have same node characteristics and they will choose the popular other) or 1 (meaning the nodes have different node characteristics and the whole equation will produce some final utility value). This second half of the equation  $(- \alpha * \text{Difference in node characteristic}_{ij})$  is responsible for the nodes behavior of seeking the homophilious others.

$$U_{ijt} = (1 - \alpha) p_{jt-1} - \alpha |y_{it-1} - y_{jt-1}| \quad (1)$$

Where,  $U_{ijt}$  is utility function for selection part of the model;

$\alpha$  is homophily factor,  $(0 \leq \alpha \leq 1)$ ;

$p_{jt-1}$  is the node j's popularity;

t is time; and

$|y_{it-1} - y_{jt-1}|$  is the difference in node characteristics of node i and j at time t-1.

2. The out degree of each node is kept constant during this procedure, i.e., the node will have the same number of links as they had in the setup stage but will have a chance to select the nodes, they want to have these links with.
3. Finally, based on the investigator provide “domain difference percentage” values (50%, 25%, and 0%) the simulation provided different results.
4. This whole process is repeated for 50 times. The final structures were analyzed based on varied alpha, and percentage of node characteristics similarity.

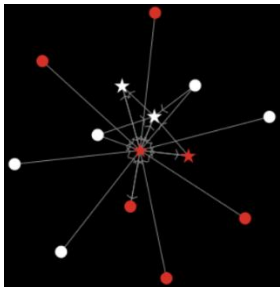
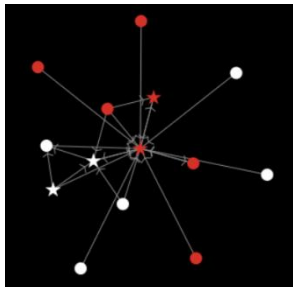
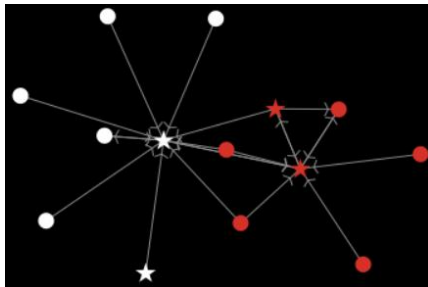
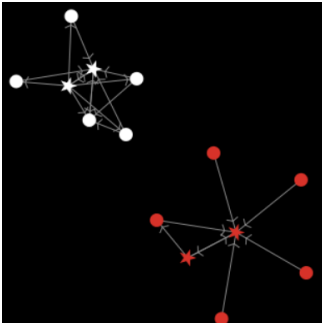
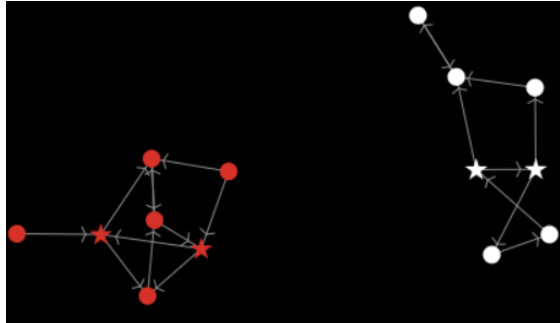
As explained in the Chapter-3 Methodology, each experiment (one to eight) the population of node had two types of characteristics. The ratio of nodes having one type of characteristic versus the ones having the other type was represented in terms of domain difference percentage. In other words, if the population of 20 nodes in a simulation consists of 10 having a type of characteristic and other 10 having another type of characteristic, then the domain difference percentage is mentioned as 50%. If this population changes to 5 nodes of one type and 15 nodes of another type, then the domain difference percentage is mentioned as 25%. Thus, by the logic if all nodes have same characteristic then the domain difference percentage is 0%.



#### 4.2.1 Experiment 1: Small Sized Network using Degree Centrality and Homophily

This experiment investigated if nodes seeking homophily via node characteristics similarity versus nodes seeking popular-other nodes via degree centrality influence the evolution of AEC project team social network structure overtime. This was conducted over a small sized network.

As soon as the simulation setup is complete a core periphery network structure appears, with 4 core and 10 periphery nodes (Refer Figure 3.2). The final and intermediate structure snapshots and the results observed during the simulation run, produced by varying, alpha from 0 to 1, and domain difference percentage from 50% to 0% are illustrated in the Figure 4.2.

Domain Difference Percentage = 50%			
Final Network Structure			
	(A) Core-periphery-and-triangles	(B) 1 or 2 Core-periphery-and-triangles bridged together	(C) 2 Core-periphery-and-triangles bridged together
Alpha	0 to 0.6	0.6 to 0.7	0.7 to 0.8
Final Network Structure			
	(D) 2 Core-periphery-and-triangles	(E) 2 Subgroups	
Alpha	0.8 to 0.9	1.0	
LEGENDS: ★ Core Node; ● Periphery Node; Color-Signifies a type of node characteristic.			

**Figure 4.2** Experiment-1 Simulation Results

Figure 4.2 (cont'd)

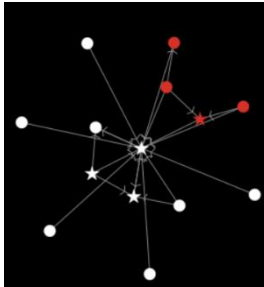
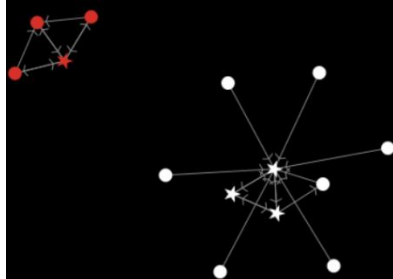
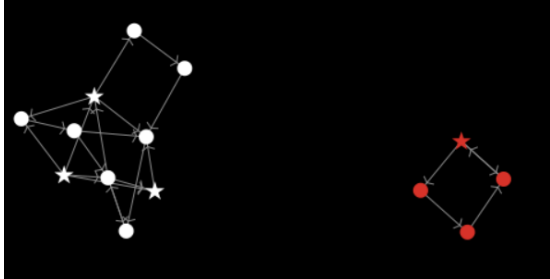
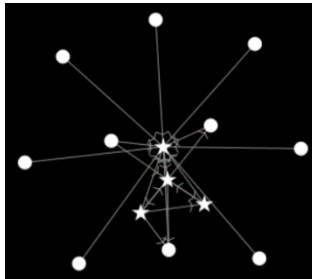
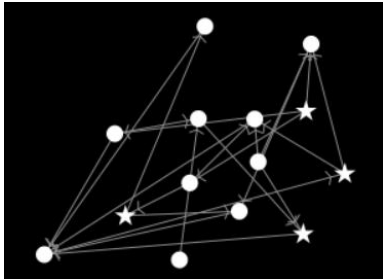
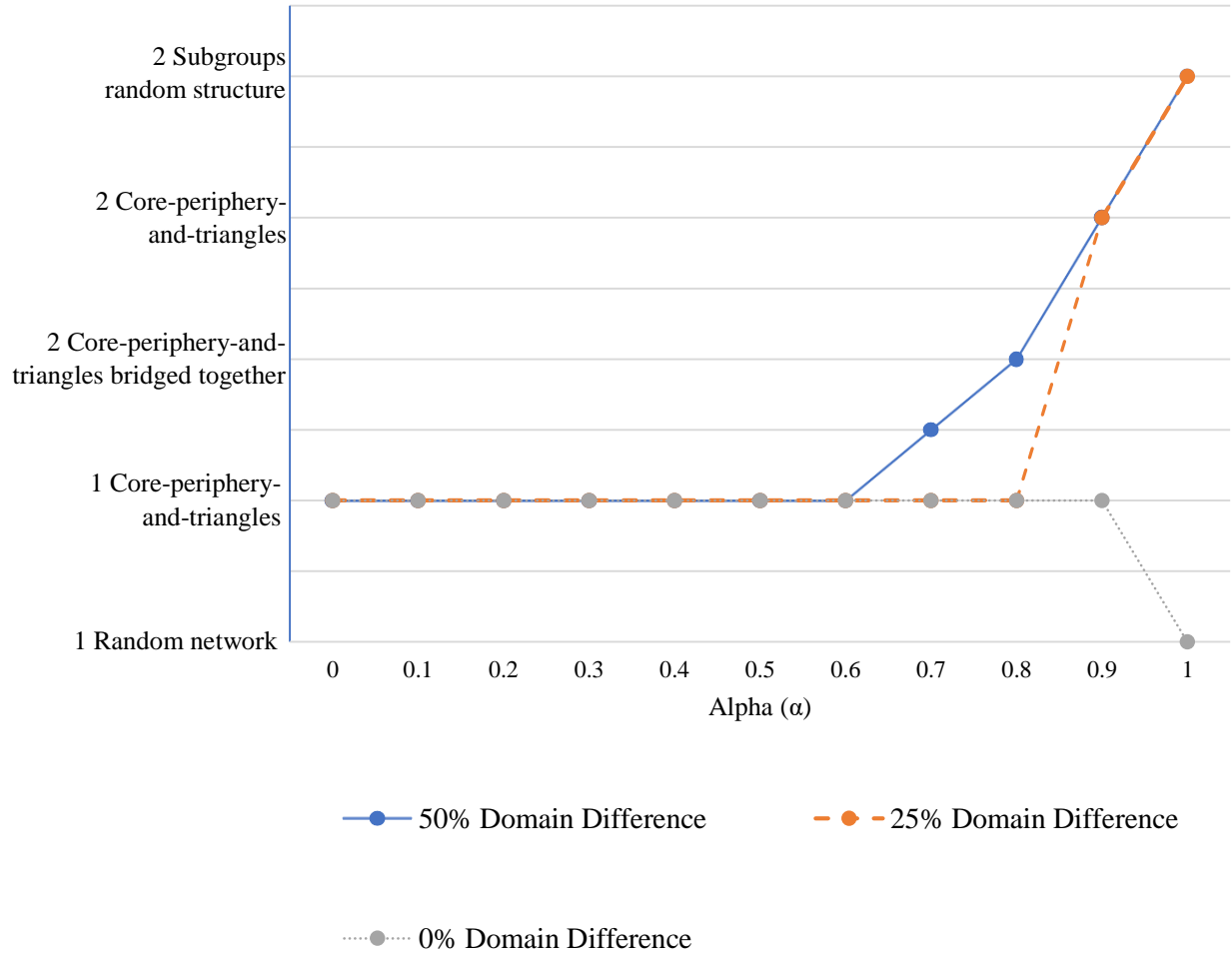
Domain Difference Percentage = 25%			
Final Network Structure			
	(F) Core-periphery-and-triangles	(G) 2 Core-periphery-and-triangles	(H) 2 Subgroups
Alpha	0 to 0.8	0.8 to 0.9	1.0
Domain Difference Percentage = 0%			
Final Network Structure			
	(I) Core-periphery-and-triangles	(J) Random Network	
Alpha	0 to 0.9	1.0	
LEGENDS: ★ Core Node; ● Periphery Node; Color-Signifies a type of node characteristic.			

Figure 4.2 shows the following:

- For 50% domain difference, as alpha (homophily coefficient) varies from 0 to 0.6 a core-periphery-with-triangles structure appears. As soon as alpha ranges between 0.6 and 0.7 either one core-periphery-with triangles structure or two core-periphery-with-triangles structure bridged with nodes start to appear. When alpha is between 0.7 and 0.8 only two core-periphery-with-triangles structure bridged with nodes appear. Just when alpha lies between 0.8 and 0.9 the structure breaks and form two core-periphery-with-triangles structures. When alpha equals one two subgroups with random structure shows up.
- For 25% domain difference, as alpha varies from 0 to 0.8 a core-periphery-with-triangles structure appears. As soon as alpha ranges between 0.8 and 0.9 two core-periphery-with-triangles structures appear. When alpha equals one two subgroups with random structure shows up.
- For 0% domain difference, for no value of alpha the structure breaks. As alpha varies from 0 to 0.9 a core-periphery-with-triangles structure appears. When alpha equals one a random structure shows up.

To summarize the observed network structures in simulation experiment one, Figure 4.3 depicts a plot showing the comparison of all the final structures that emerged out from varying domain difference percentage and the alpha values.



**Figure 4.3** Plotlines Summarizing the Observed Network Structures in Simulation Experiment-1

In Figure 4.3 vertical axis represents the types of network structures that finally appeared in the experiment one. The light grey horizontal gridlines in front of all the named structures represents the respective network structure type. Similarly, the horizontal gridlines that lie in between two such named gridlines, represent the possibility of a final structure that can be either of the two structures, the one that represents the gridline just above the considered gridline or just below the considered gridline. Horizontal axis plots the values of alpha for with the results were collected. Each dot's vertical level represents the structure that emerged at its corresponding alpha value on the horizontal axis. The three plot lines are for different domain conditions. The blue plotline

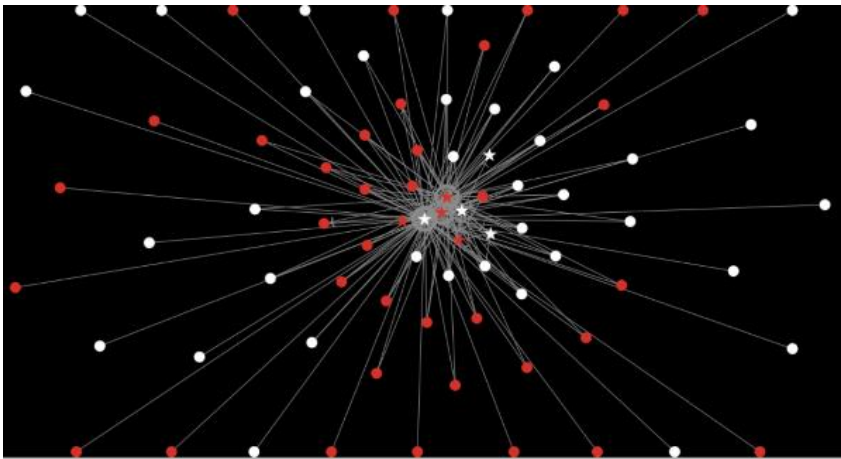
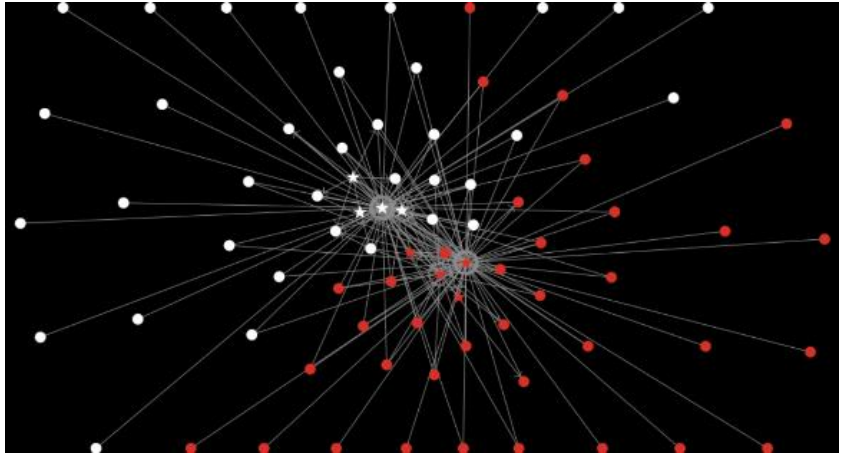
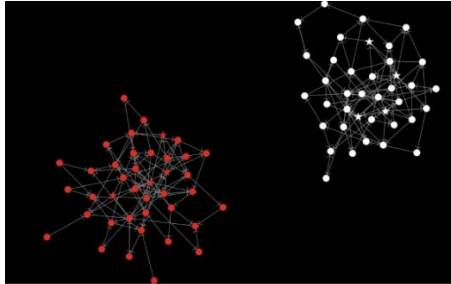
represents the final network structures that emerged when the domain difference percentage was 50%, i.e., when the node population had 50% of nodes having one kind of node characteristic and other 50% having another kind of node characteristic. The yellow plotline represents the final network structures that emerged when the domain difference percentage was 25%, i.e., when the node population had 25% of nodes having one kind of node characteristic and other 75% having another kind of node characteristic. The grey plotline represents the final network structures that emerged when the domain difference percentage was 0%, i.e., when the characteristics of all the nodes were same.

#### 4.2.2 Experiment 2: Real-world Sized Network using Degree Centrality and Homophily

This experiment investigated if nodes seeking homophily via node characteristics similarity versus nodes seeking popular-other nodes via degree centrality influence the evolution of AEC project team social network structure overtime. This was conducted over a real-world sized network.

After setting up the type-2 initial condition for the simulation a core periphery network structure appears, with 8 core and 71 periphery nodes (Refer Figure 3.3). The only change in this simulation setting is the network structure size, that is based on a real-world network structure found in an empirical case study by Garcia et al. (2021a and b).

The final and intermediate structure snapshots and the results observed during the simulation run, produced by varying,  $\alpha$  from 0 to 1, and domain difference percentage from 50% to 0% are illustrated in the Figure 4.4.

Domain Difference Percentage = 50%	
Final Network Structure	 <p>(A) Core-periphery-and-triangles</p>
Alpha	0 to 0.8
Final Network Structure	 <p>(B) 2 Core-periphery-and-triangles bridged together</p>
Alpha	0.8 to 0.9
Final Network Structure	 <p>(C) 2 Subgroups</p>
Alpha	1.0
LEGENDS: ★ Core Node; ● Periphery Node; Color-Signifies a type of node characteristic.	

**Figure 4.4** Experiment-2 Simulation Results

**Figure 4.4 (Cont'd)**

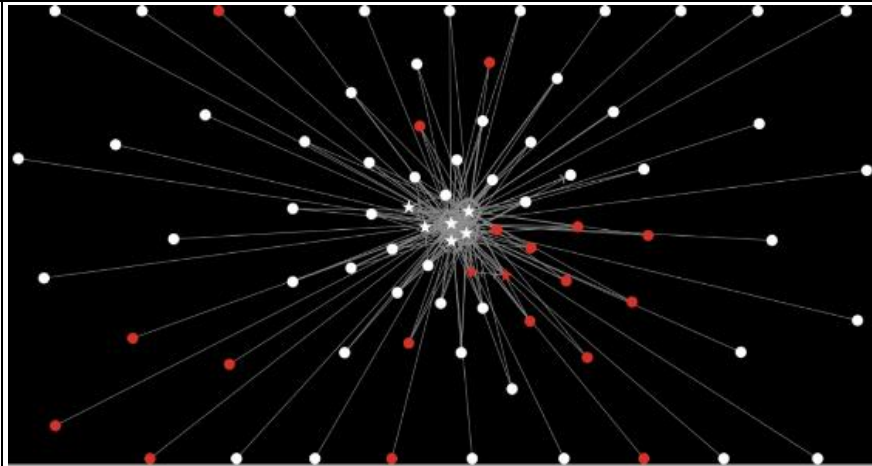
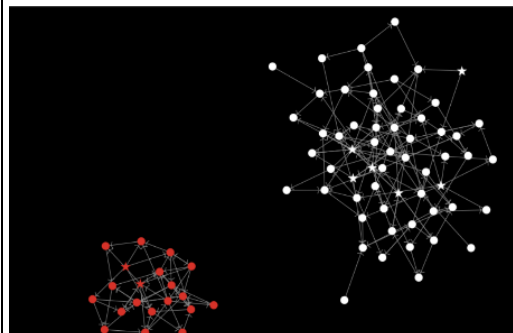
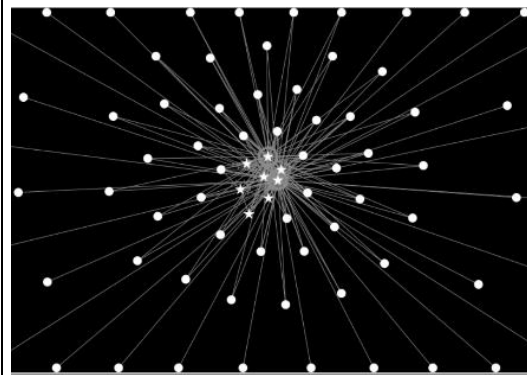
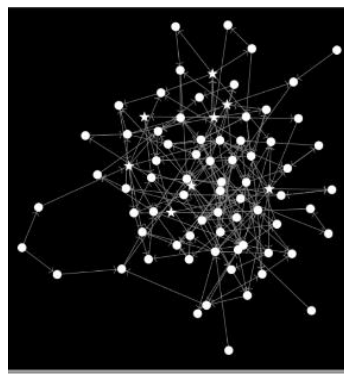
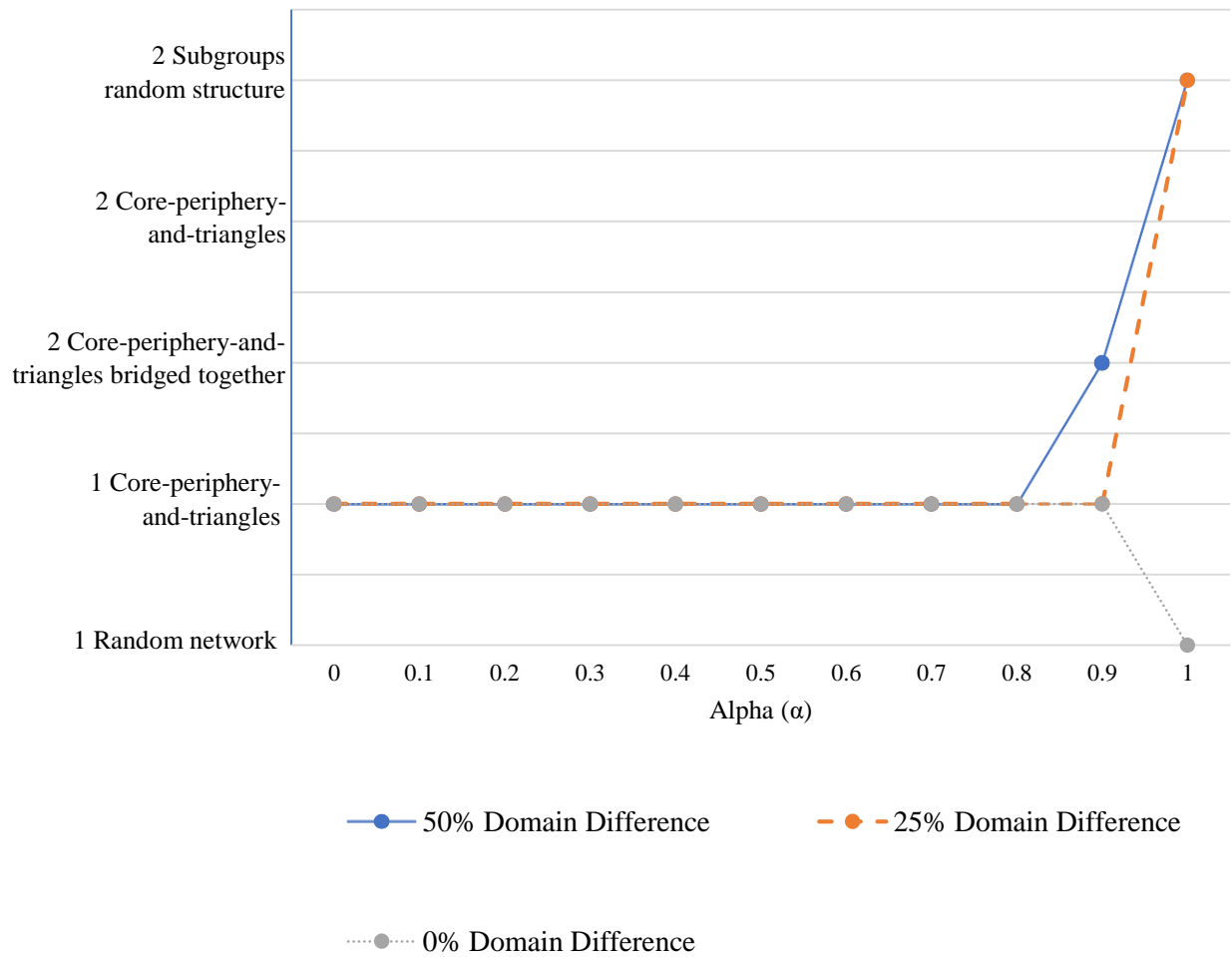
Domain Difference Percentage = 25%		
Final Network Structure	 <p>(D) Core-periphery-and-triangles</p>	
Alpha	0 to 0.9	
Final Network Structure	 <p>(E) 2 Subgroups</p>	
Alpha	1.0	
Domain Difference Percentage = 0%		
Final Network Structure	 <p>(F) Core-periphery-and-triangles</p>	 <p>(G) Random Network</p>
Alpha	0 to 0.9	1
LEGENDS: ★ Core Node; ● Periphery Node; Color-Signifies a type of node characteristic.		



Figure 4.4 shows the following:

- For 50% domain difference, as alpha (homophily coefficient) varies from 0 to 0.8 a core-periphery-with-triangles structure appears. As soon as alpha ranges between 0.8 and 0.9 two core-periphery-with-triangles structure bridged with nodes start to appear. When alpha equals one two subgroups with random structure shows up.
- For 25% domain difference, as alpha varies from 0 to 0.9 a core-periphery-with-triangles structure appears. When alpha equals one two subgroups with random structure shows up.
- For 0% domain difference, for no value of alpha the structure breaks. As alpha varies from 0 to 0.9 a core-periphery-with-triangles structure appears. When alpha equals one a random structure shows up.

To summarize the observed network structures in simulation experiment two, Figure 4.5 depicts a plot showing the comparison of all the final structures that emerged out from varying domain difference percentage and the alpha values.



**Figure 4.5** Plotlines Summarizing the Observed Network Structures in Simulation Experiment-2

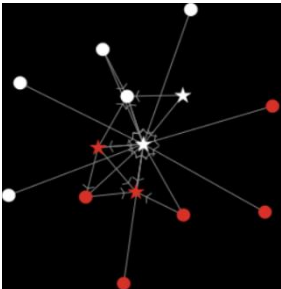
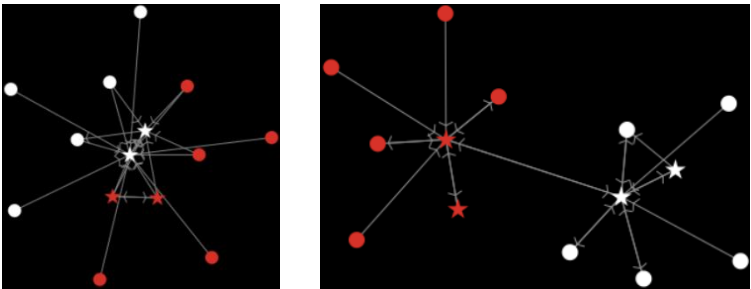
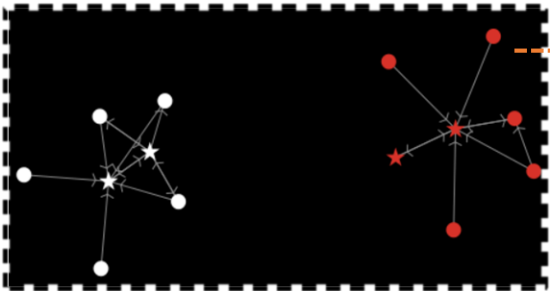
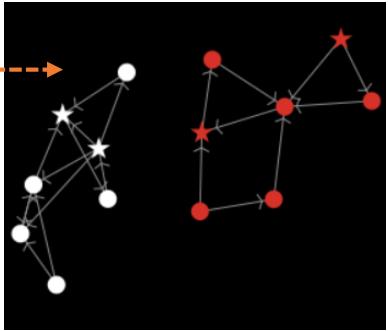
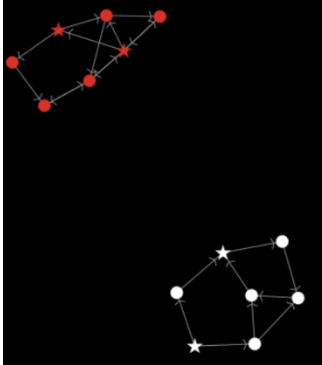
In Figure 4.5 vertical axis represents the types of network structures that finally appeared in the experiment two. The light grey horizontal gridlines in front of all the named structures represents the respective network structure type. Similarly, the horizontal gridlines that lie in between two such named gridlines, represent the possibility of a final structure that can be either of the two structures, the one that represents the gridline just above the considered gridline or just below the considered gridline. Horizontal axis plots the values of alpha for with the results were collected. Each dot's vertical level represents the structure that emerged at its corresponding alpha value on the horizontal axis. The three plot lines are for different domain conditions. The blue plotline

represents the final network structures that emerged when the domain difference percentage was 50%, i.e., when the node population had 50% of nodes having one kind of node characteristic and other 50% having another kind of node characteristic. The yellow plotline represents the final network structures that emerged when the domain difference percentage was 25%, i.e., when the node population had 25% of nodes having one kind of node characteristic and other 75% having another kind of node characteristic. The grey plotline represents the final network structures that emerged when the domain difference percentage was 0%, i.e., when the characteristics of all the nodes were same.

#### 4.2.3 Experiment 3: Small Sized Network using Eigenvector Centrality and Homophily

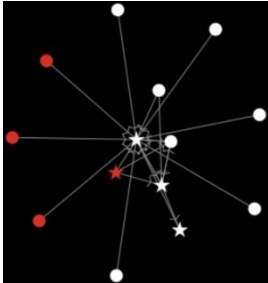
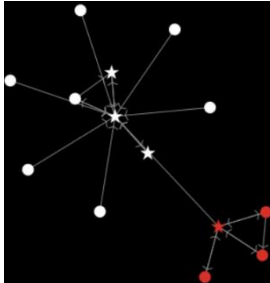
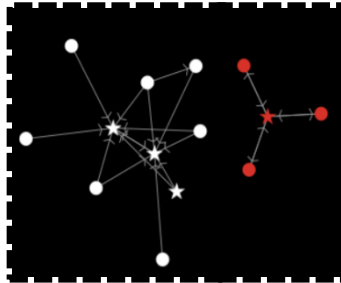
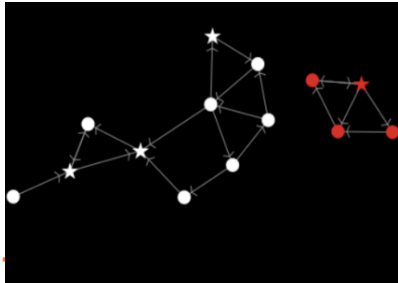
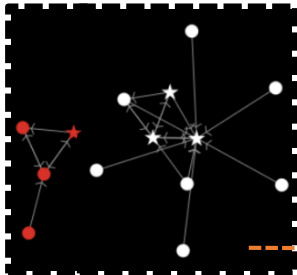
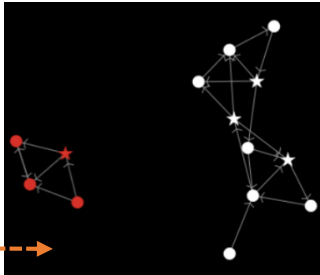
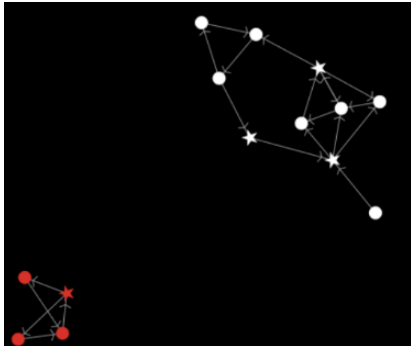
This experiment investigated if nodes seeking homophily via node characteristics similarity versus nodes seeking popular-other nodes via eigenvector centrality influence the evolution of AEC project team social network structure overtime. This was conducted over a small sized network.

Experiment three and four follows the exact same steps from experiment one and two respectively, with only a change in measurement of nodes' popularity. In both the experiments three and four the degree centrality is replaced by eigenvector centrality and the simulations are re-run. The initial condition produced by the setup in experiment three and four still remains the same as in experiment one and two respectively (refer figure 3.2 and 3.3). The final and intermediate structure snapshots and the results observed during the simulation run, produced by varying, alpha from 0 to 1, and domain difference percentage from 50% to 0% are illustrated in the Figure 4.6.

Domain Difference Percentage = 50%		
Final Network Structure	 <p>(A) Core-periphery-and-triangles</p>	 <p>(B) 1 or 2 Core-periphery-and-triangles</p>
Alpha	0 to 0.1	0.1 to 0.4
Final Network Structure	 <p>(C.1) 2 Core-periphery-and-triangles (as intermediate structure)</p>  <p>(C.2) 2 Subgroups (as final structure)</p>	 <p>(D) 2 Subgroups</p>
Alpha	0.4 to 0.9	1.0
LEGENDS: ★ Core Node; ● Periphery Node; Color-Signifies a type of node characteristic.		

**Figure 4.6** Experiment-3 Simulation Results

Figure 4.6 (cont'd)

Domain Difference Percentage = 25%				
Final Network Structure				
	(E) Core-periphery-and-triangles	(F.1) 2 Core-periphery -and-triangles bridged	(F.2) 2 Core-periphery (Intermediate structure)	(F.3) 2 Subgroups (Final structure)
Alpha	0 to 0.4	0.4 to 0.6		
Final Network Structure	 			
	(G.1) 2 Core-periphery and-triangles (Intermediate structure)		(G.2) 2 Subgroups (Final structure)	
Alpha	0.6 to 0.9		1.0	
LEGENDS: ★ Core Node; ● Periphery Node; Color-Signifies a type of node characteristic.				

**Figure 4.6 (cont'd)**

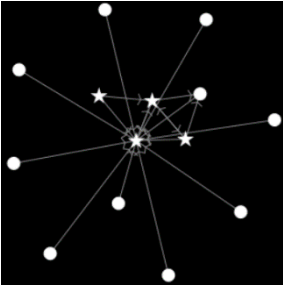
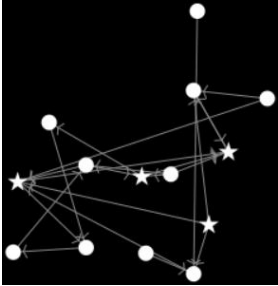
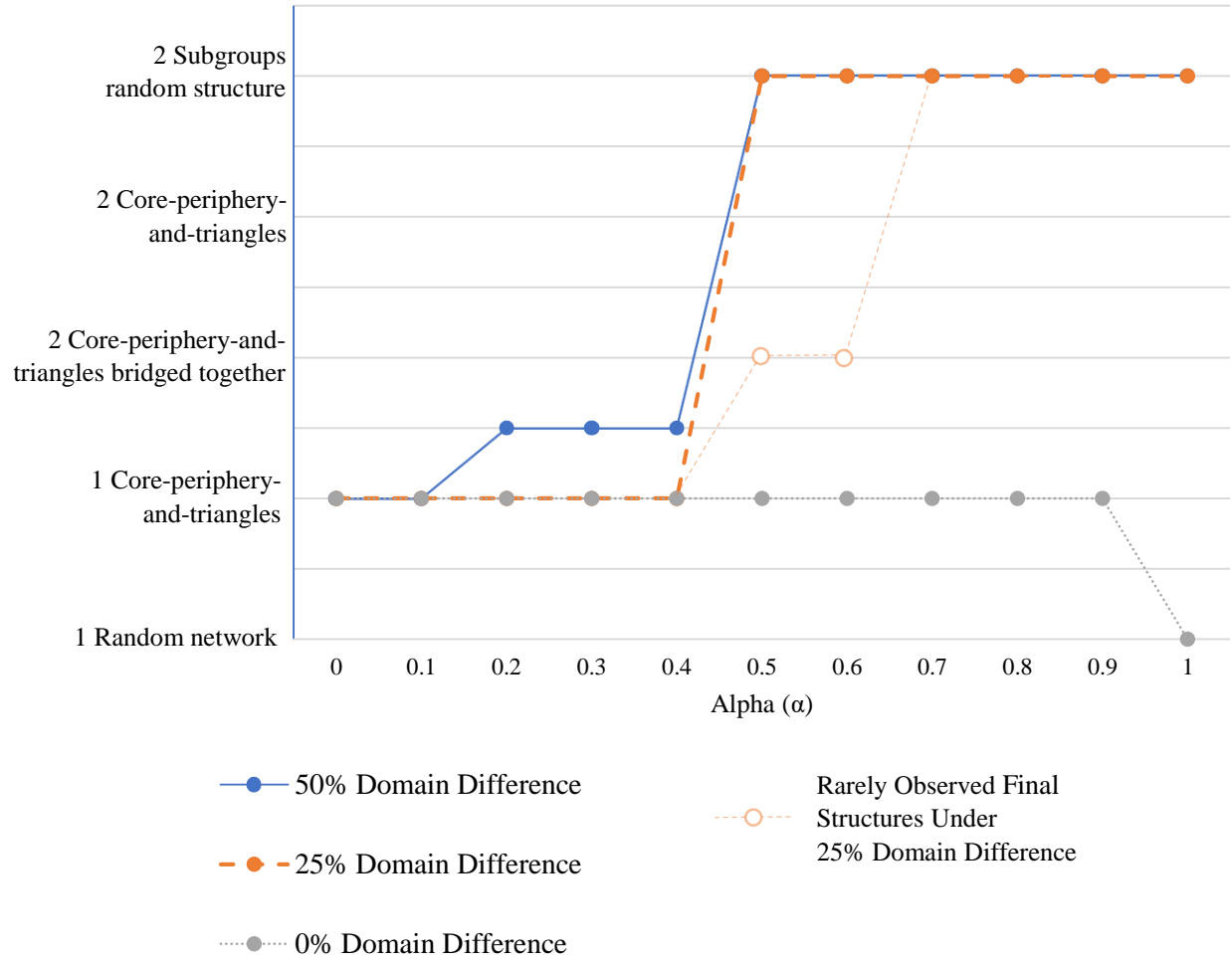
Domain Difference Percentage = 25%		
Final Network Structure	 <p>(I) Core-periphery-and-triangles</p>	 <p>(J) Core-periphery-and-triangles</p>
Alpha	0 to 0.9	1.0
LEGENDS: ★ Core Node; ● Periphery Node; Color-Signifies a type of node characteristic.		

Figure 4.6 shows the following:

- For 50% domain difference, as alpha (homophily coefficient) varies from 0 to 0.1 a core-periphery-with-triangles structure appears. As soon as alpha ranges between 0.1 and 0.4 either one core-periphery-with triangles structure or two core-periphery-with-triangles structure bridged with nodes start to appear. When alpha is between 0.4 and 0.9 structure breaks to form two core-periphery-with-triangles as an intermediate structure and then stabilizes to form two subgroups showing random structure. When alpha equals one two subgroups with random structure shows up at once.
- For 25% domain difference, as alpha varies from 0 to 0.4 a core-periphery-with-triangles structure appears. As soon as alpha ranges between 0.4 and 0.6 either two core-periphery-with-triangles structures bridged with nodes appear or two intermediate structures, two core-periphery-with-triangles, appears only to stabilize and form two subgroups showing random structure overtime. When alpha ranges from 0.6 to 0.9 two core-periphery-with-triangles appear and stabilizes into two subgroups showing random structure overtime. When alpha equals one two subgroups with random structure appears at once.
- For 0% domain difference, for no value of alpha the structure breaks. As alpha varies from 0 to 0.9 a core-periphery-with-triangles structure appears. When alpha equals one a random structure shows up.

To summarize the observed network structures in simulation experiment three, Figure 4.4 depicts a plot showing the comparison of all the final structures that emerged out from varying domain difference percentage and the alpha values.



**Figure 4.7** Plotlines Summarizing the Observed Network Structures in Simulation Experiment-3

In Figure 4.7 vertical axis represents the types of network structures that finally appeared in the experiment three. The light grey horizontal gridlines in front of all the named structures represents the respective network structure type. Similarly, the horizontal gridlines that lie in between two such named gridlines, represent the possibility of a final structure that can be either of the two structures, the one that represents the gridline just above the considered gridline or just below the considered gridline. Horizontal axis plots the values of alpha for with the results were collected. Each dot's vertical level represents the structure that emerged at its corresponding alpha value on the horizontal axis. The four plot lines are for different domain conditions. The blue plotline

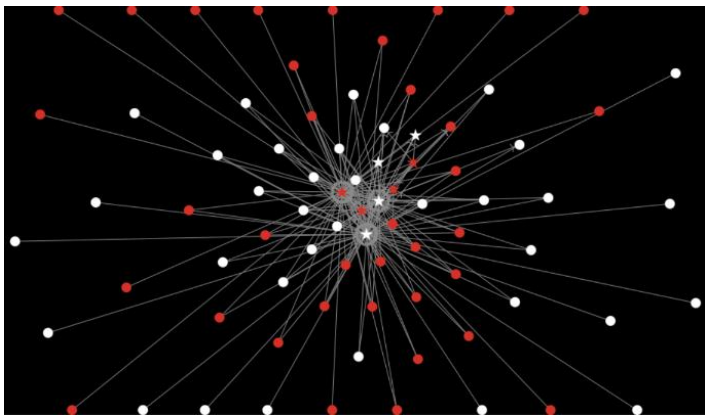
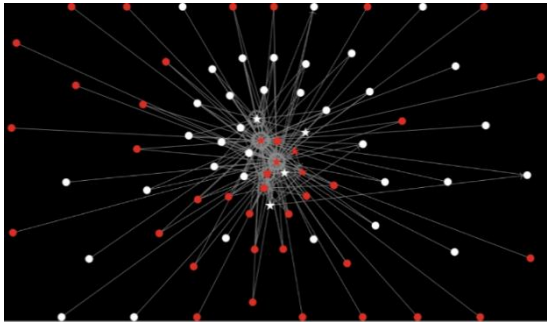
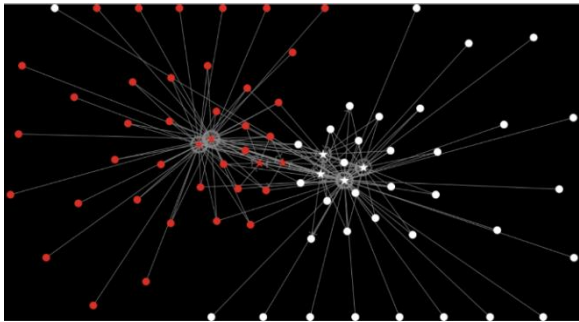
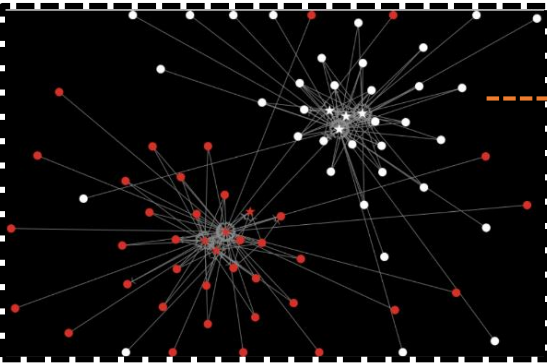
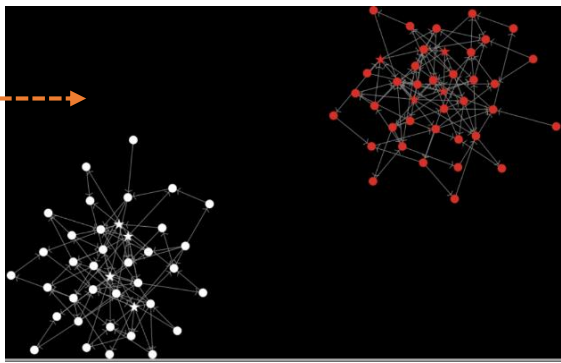


represents the final network structures that emerged when the domain difference percentage was 50%, i.e., when the node population had 50% of nodes having one kind of node characteristic and other 50% having another kind of node characteristic. The yellow plotline represents the final network structures that emerged when the domain difference percentage was 25%, i.e., when the node population had 25% of nodes having one kind of node characteristic and other 75% having another kind of node characteristic. The light-yellow plot line represents the rarely observed final network structures that emerge when the domain difference percentage was 25%. The grey plotline represents the final network structures that emerged when the domain difference percentage was 0%, i.e., when the characteristics of all the nodes were same.

#### 4.2.4 Experiment 4: Real-world Sized Network using Eigenvector Centrality and Homophily

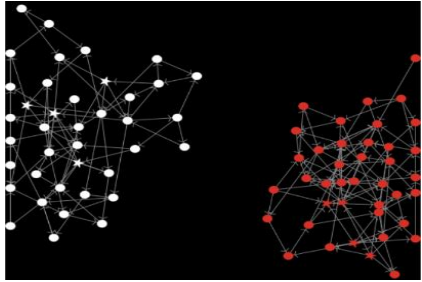
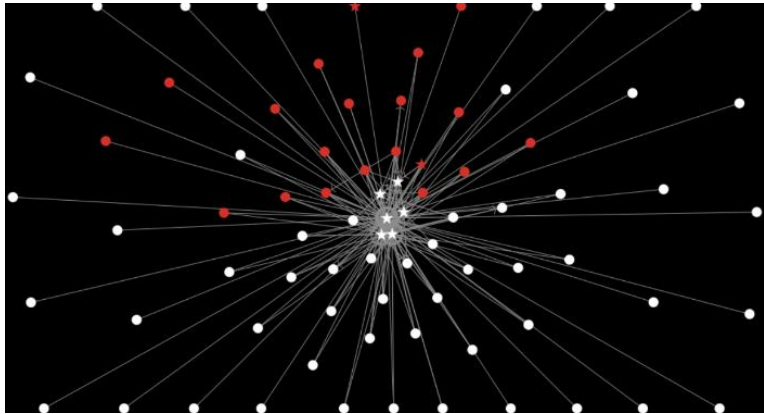
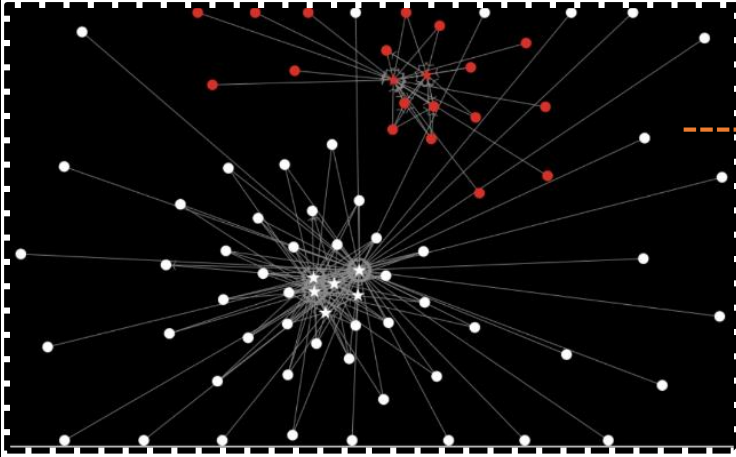
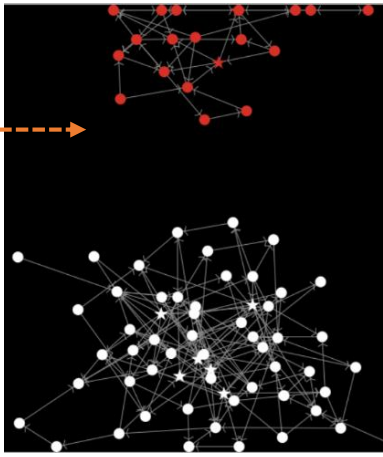
This experiment investigated if nodes seeking homophily via node characteristics similarity versus popularity seeking via eigenvector centrality influence the evolution of AEC project team social network structure overtime. This was conducted over a real-world sized network.

The final and intermediate structure snapshots and the results observed during the simulation run, produced by varying, alpha from 0 to 1, and domain difference percentage from 50% to 0% are illustrated in the Figure 4.8.

Domain Difference Percentage = 50%	
Final Network Structure	 <p>(A) Core-periphery-and-triangles</p>
Alpha	0 to 0.1
Final Network Structure	<div>   </div> <p>(B.1) Core-periphery-and-triangles (or) (B.2) 2 Core-periphery-and-triangle bridged together</p>
Alpha	0.1 to 0.4
Final Network Structure	<div>   </div> <p>(C.1) 2 Core-periphery-and-triangles (Intermediate structure) (C.2) 2 Subgroups (Final structure)</p>
Alpha	0.4 to 0.9
LEGENDS: ★ Core Node; ● Periphery Node; Color-Signifies a type of node characteristic.	

**Figure 4.8** Experiment-4 Simulation Results

**Figure 4.8 (cont'd)**

<b>Final Network Structure</b>	 <p>(D) 2 Subgroups</p>
<b>Alpha</b>	1.0
<b>Domain Difference Percentage = 25%</b>	
<b>Final Network Structure</b>	 <p>(E) Core-periphery-and-triangles</p>
<b>Alpha</b>	0 to 0.4
<b>Final Network Structure</b>	<div>  <p>(E.1) 2 Core-periphery-and-triangles (Intermediate structure)</p> </div> <div>  <p>(E.2) 2 Subgroups (Final structure)</p> </div>
<b>Alpha</b>	0.4 to 0.9
<b>LEGENDS: ★Core Node; ●Periphery Node; Color-Signifies a type of node characteristic.</b>	

**Figure 4.8 (cont'd)**

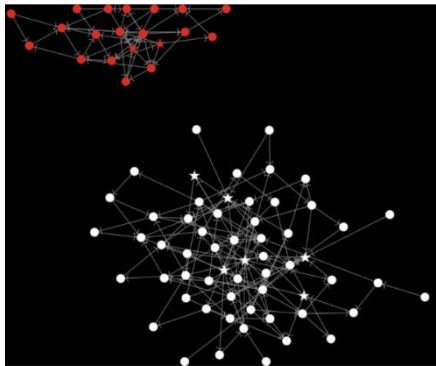
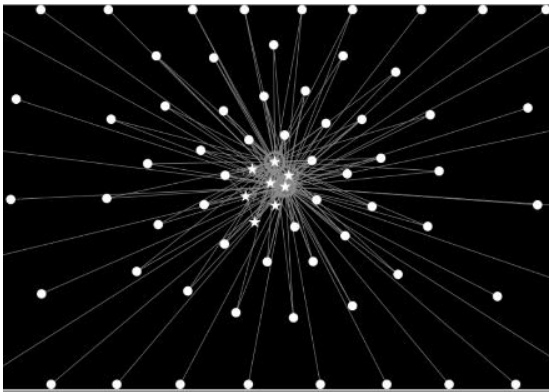
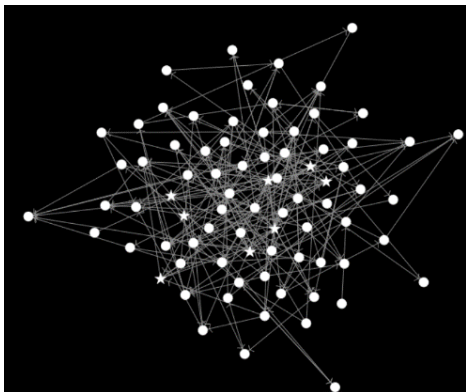
Final Network Structure	 <p>(D) 2 Subgroups</p>	
Alpha	1.0	
Domain Difference Percentage = 0%		
Final Network Structure	 <p>(F) Core-periphery-and-triangles</p>	 <p>(G) Random Network</p>
Alpha	0 to 0.9	1.0
LEGENDS: ★Core Node; ●Periphery Node; Color-Signifies a type of node characteristic.		

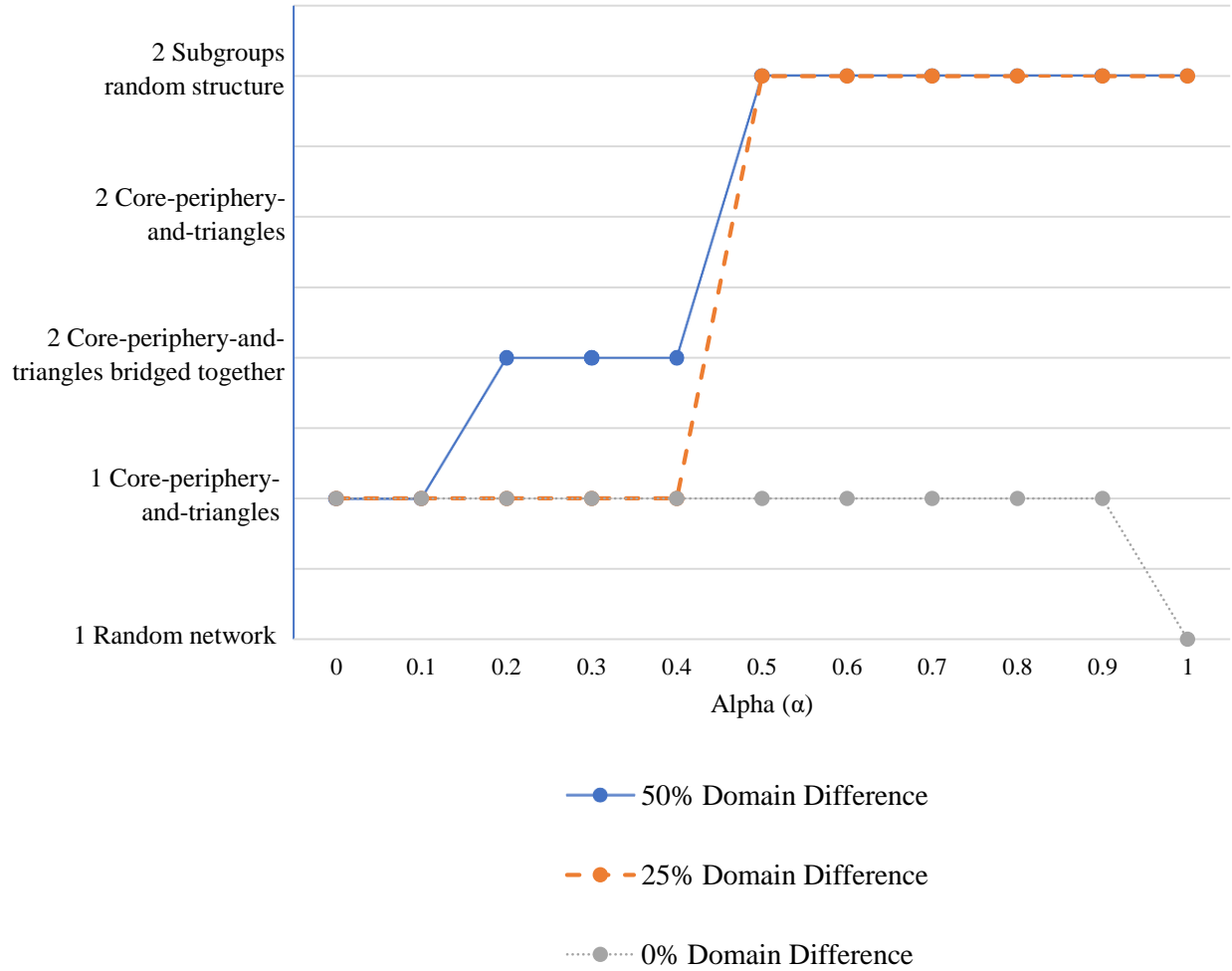
Figure 4.8 shows the following:

- For 50% domain difference, as alpha (homophily coefficient) varies from 0 to 0.1 a core-periphery-with-triangles structure appears. As soon as alpha ranges between 0.1 and 0.4 either one core-periphery-with triangles structure or two core-periphery-with-triangles structure bridged with nodes start to appear. When alpha is between 0.4 and 0.9 structure breaks to form two core-periphery-with-triangles as an intermediate structure and then

stabilizes to form two subgroups showing random structure. When alpha equals one two subgroups with random structure shows up at once.

- For 25% domain difference, as alpha varies from 0 to 0.4 a core-periphery-with-triangles structure appears. As soon as alpha ranges between 0.4 and 0.9 two core-periphery-with-triangles as intermediate structures appears only to stabilize and form two subgroups showing random structure overtime. When alpha equals one two subgroups with random structure appears at once.
- For 0% domain difference, for no value of alpha the structure breaks. As alpha varies from 0 to 0.9 a core-periphery-with-triangles structure appears. When alpha equals one a random structure shows up.

To summarize the observed network structures in simulation experiment four, Figure 4.9 depicts a plot showing the comparison of all the final structures that emerged out from varying domain difference percentage and the alpha values.



**Figure 4.9** Plotlines Summarizing the Observed Network Structures in Simulation Experiment-4

In Figure 4.9 vertical axis represents the types of network structures that finally appeared in the experiment four. The light grey horizontal gridlines in front of all the named structures represents the respective network structure type. Similarly, the horizontal gridlines that lie in between two such named gridlines, represent the possibility of a final structure that can be either of the two structures, the one that represents the gridline just above the considered gridline or just below the considered gridline. Horizontal axis plots the values of alpha for with the results were collected. Each dot's vertical level represents the structure that emerged at its corresponding alpha value on the horizontal axis. The three plot lines are for different domain conditions. The blue plotline

represents the final network structures that emerged when the domain difference percentage was 50%, i.e., when the node population had 50% of nodes having one kind of node characteristic and other 50% having another kind of node characteristic. The yellow plotline represents the final network structures that emerged when the domain difference percentage was 25%, i.e., when the node population had 25% of nodes having one kind of node characteristic and other 75% having another kind of node characteristic. The grey plotline represents the final network structures that emerged when the domain difference percentage was 0%, i.e., when the characteristics of all the nodes were same.

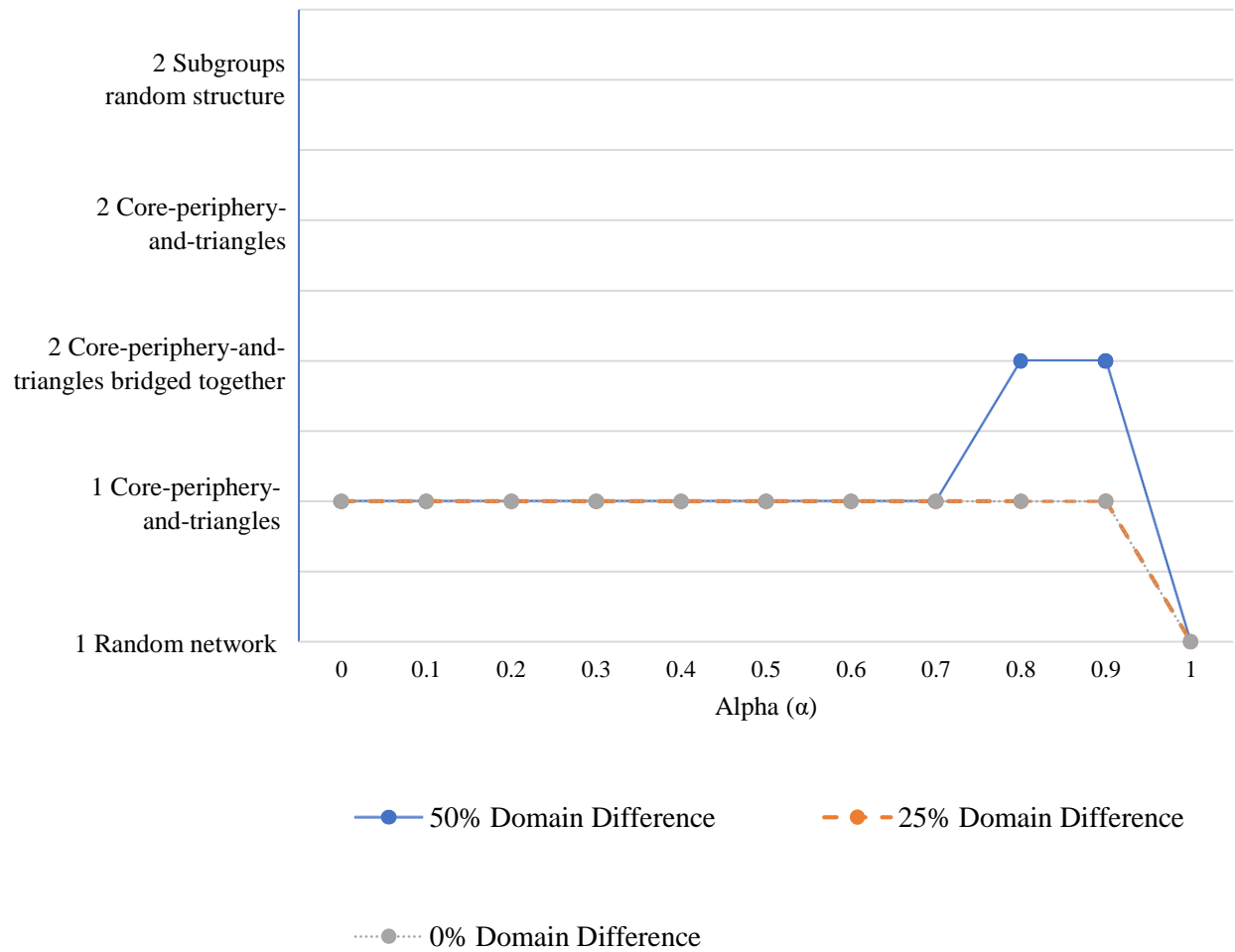
#### 4.2.5 Experiment 5: Small Sized Network using Degree Centrality and Heterophily

Experiments five to eight were analogous to experiment one to four in all terms except the nodes' homophily seeking characteristic was replaced by nodes' heterophily seeking characteristic. This was accomplished by using equation 3 mentioned in the Chapter-3 Methodology.

$$U_{ijt} = (1-\alpha) p_{jt-1} + \alpha |y_{it-1} - y_{jt-1}| \quad (3)$$

This experiment investigated if nodes seeking heterophily via node characteristics similarity versus popularity seeking via degree centrality influence the evolution of AEC project team social network structure overtime. This was conducted over a small sized network.

To summarize the observed network structures in simulation experiment five, Figure 4.10 depicts a plot showing the comparison of all the final structures that emerged out from varying domain difference percentage and the alpha values.



**Figure 4.10** Plotlines Summarizing the Observed Network Structure in Simulation Experiment-5

In Figure 4.10 vertical axis represents the types of network structures that finally appeared in the experiment three. The light grey horizontal gridlines in front of all the named structures represents the respective network structure type. Similarly, the horizontal gridlines that lie in between two such named gridlines, represent the possibility of a final structure that can be either of the two structures, the one that represents the gridline just above the considered gridline or just below the considered gridline. Horizontal axis plots the values of alpha for with the results were collected. Each dot's vertical level represents the structure that emerged at its corresponding alpha value on the horizontal axis. The three plot lines are for different domain conditions. The blue plotline

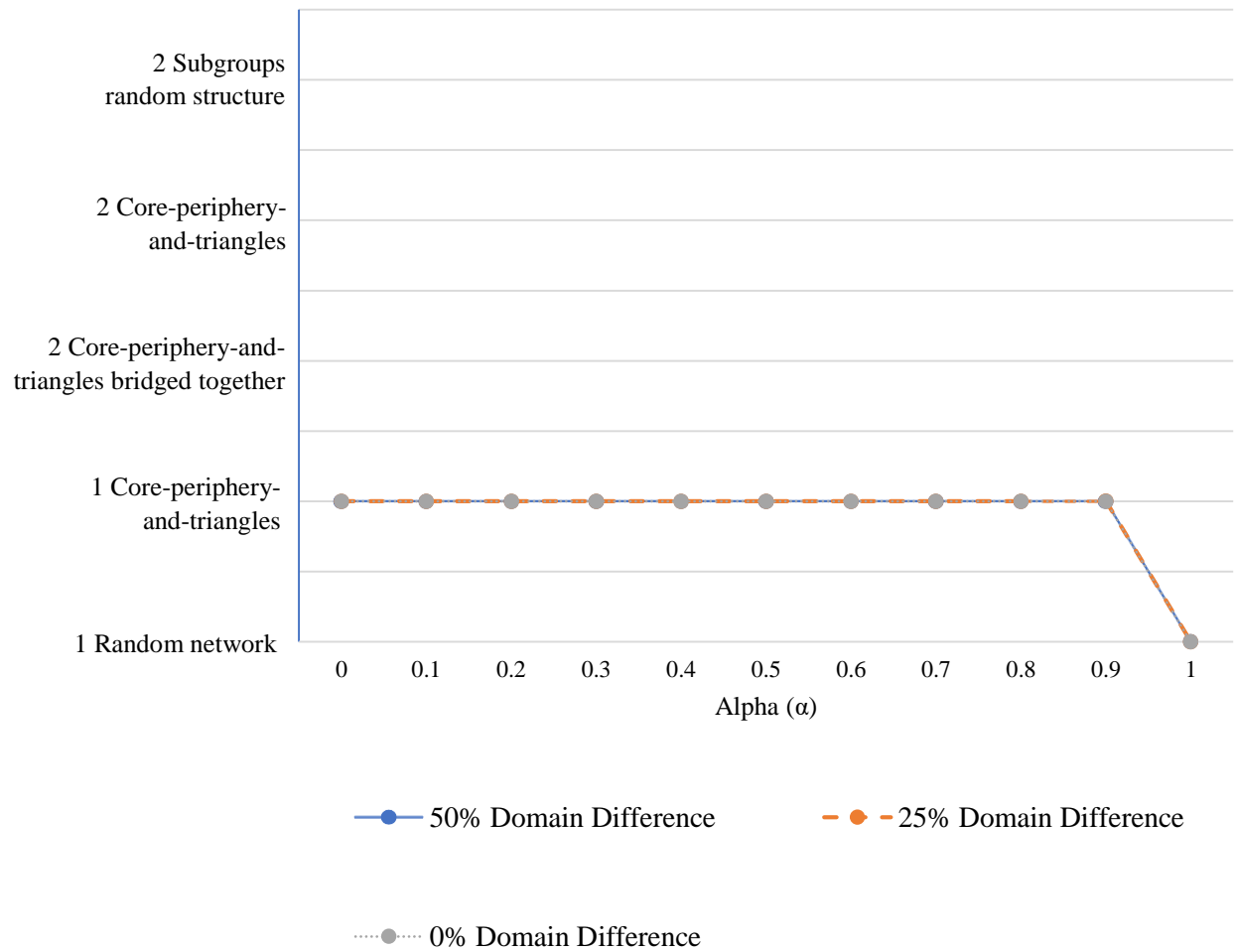


represents the final network structures that emerged when the domain difference percentage was 50%, i.e., when the node population had 50% of nodes having one kind of node characteristic and other 50% having another kind of node characteristic. The yellow plotline represents the final network structures that emerged when the domain difference percentage was 25%, i.e., when the node population had 25% of nodes having one kind of node characteristic and other 75% having another kind of node characteristic. The grey plotline represents the final network structures that emerged when the domain difference percentage was 0%, i.e., when the characteristics of all the nodes were same.

#### 4.2.6 Experiment 6: Real-world Sized Network using Degree Centrality and Heterophily

This experiment investigated if nodes seeking heterophily via node characteristics similarity versus popularity seeking via degree centrality influence the evolution of AEC project team social network structure overtime. This was conducted over a real-world sized network.

To summarize the observed network structures in simulation experiment six, Figure 4.11 depicts a plot showing the comparison of all the final structures that emerged out from varying domain difference percentage and the alpha values.



**Figure 4.11** Plotlines Summarizing the Observed Network Structure in Simulation Experiment-6

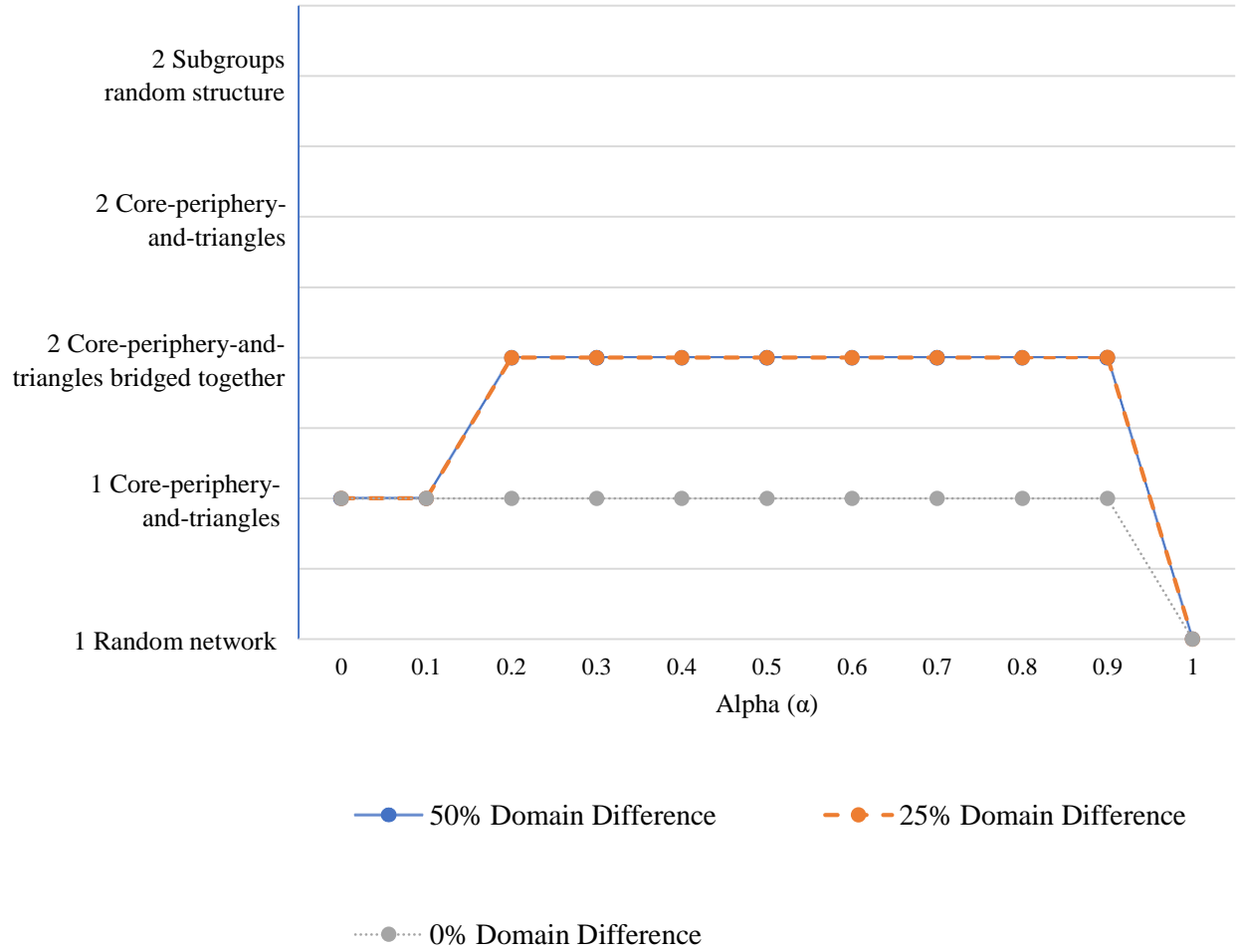
In Figure 4.11 vertical axis represents the types of network structures that finally appeared in the experiment three. The light grey horizontal gridlines in front of all the named structures represents the respective network structure type. Similarly, the horizontal gridlines that lie in between two such named gridlines, represent the possibility of a final structure that can be either of the two structures, the one that represents the gridline just above the considered gridline or just below the considered gridline. Horizontal axis plots the values of alpha for with the results were collected. Each dot's vertical level represents the structure that emerged at its corresponding alpha value on the horizontal axis. The three plot lines are for different domain conditions. The blue plotline

represents the final network structures that emerged when the domain difference percentage was 50%, i.e., when the node population had 50% of nodes having one kind of node characteristic and other 50% having another kind of node characteristic. The yellow plotline represents the final network structures that emerged when the domain difference percentage was 25%, i.e., when the node population had 25% of nodes having one kind of node characteristic and other 75% having another kind of node characteristic. The grey plotline represents the final network structures that emerged when the domain difference percentage was 0%, i.e., when the characteristics of all the nodes were same.

#### 4.2.7 Experiment 7: Small Sized Network using Eigenvector Centrality and Heterophily

This experiment investigated if nodes seeking heterophily via node characteristics similarity versus popularity seeking via eigenvector centrality influence the evolution of AEC project team social network structure overtime. This was conducted over a small sized network.

To summarize the observed network structures in simulation experiment seven, Figure 4.12 depicts a plot showing the comparison of all the final structures that emerged out from varying domain difference percentage and the alpha values.



**Figure 4.12** Plotlines Summarizing the Observed Network Structure in Simulation Experiment-7

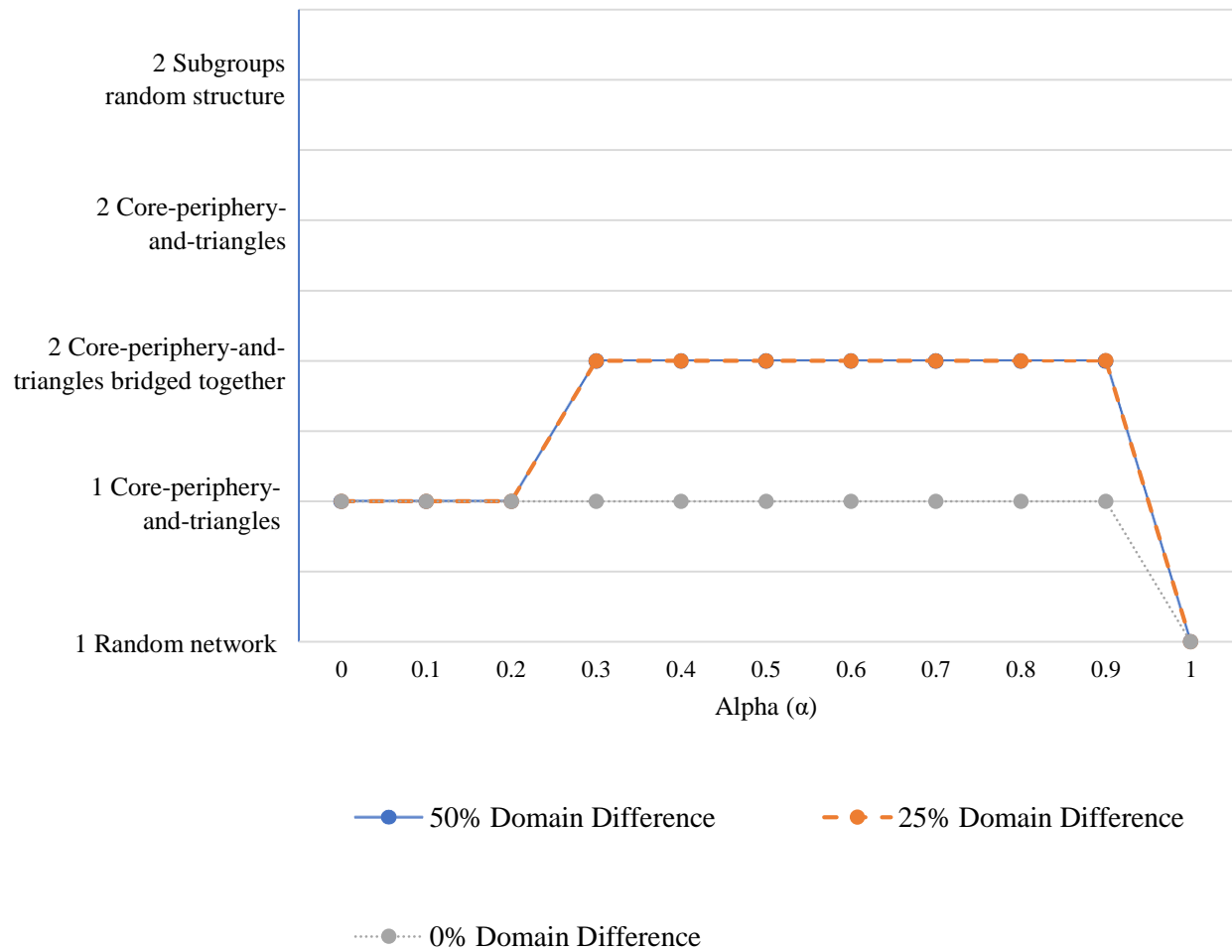
In Figure 4.12 vertical axis represents the types of network structures that finally appeared in the experiment three. The light grey horizontal gridlines in front of all the named structures represents the respective network structure type. Similarly, the horizontal gridlines that lie in between two such named gridlines, represent the possibility of a final structure that can be either of the two structures, the one that represents the gridline just above the considered gridline or just below the considered gridline. Horizontal axis plots the values of alpha for with the results were collected. Each dot's vertical level represents the structure that emerged at its corresponding alpha value on the horizontal axis. The three plot lines are for different domain conditions. The blue plotline

represents the final network structures that emerged when the domain difference percentage was 50%, i.e., when the node population had 50% of nodes having one kind of node characteristic and other 50% having another kind of node characteristic. The yellow plotline represents the final network structures that emerged when the domain difference percentage was 25%, i.e., when the node population had 25% of nodes having one kind of node characteristic and other 75% having another kind of node characteristic. The grey plotline represents the final network structures that emerged when the domain difference percentage was 0%, i.e., when the characteristics of all the nodes were same.

#### 4.2.8 Experiment 8: Real-world Sized Network using Eigenvector Centrality and Heterophily

This experiment investigated if nodes seeking heterophily via node characteristics similarity versus popularity seeking via eigenvector centrality influence the evolution of AEC project team social network structure overtime. This was conducted over a real-world sized network.

To summarize the observed network structures in simulation experiment eight, Figure 4.13 depicts a plot showing the comparison of all the final structures that emerged out from varying domain difference percentage and the alpha values.



**Figure 4.13** Plotlines Summarizing the Observed Network Structure in Simulation Experiment-8

In Figure 4.13 vertical axis represents the types of network structures that finally appeared in the experiment three. The light grey horizontal gridlines in front of all the named structures represents the respective network structure type. Similarly, the horizontal gridlines that lie in between two such named gridlines, represent the possibility of a final structure that can be either of the two structures, the one that represents the gridline just above the considered gridline or just below the considered gridline. Horizontal axis plots the values of alpha for with the results were collected. Each dot's vertical level represents the structure that emerged at its corresponding alpha value on the horizontal axis. The three plot lines are for different domain conditions. The blue plotline

represents the final network structures that emerged when the domain difference percentage was 50%, i.e., when the node population had 50% of nodes having one kind of node characteristic and other 50% having another kind of node characteristic. The yellow plotline represents the final network structures that emerged when the domain difference percentage was 25%, i.e., when the node population had 25% of nodes having one kind of node characteristic and other 75% having another kind of node characteristic. The grey plotline represents the final network structures that emerged when the domain difference percentage was 0%, i.e., when the characteristics of all the nodes were same.

## 4.2 Summary

To present both theory explanation, and prediction of a realistically sized AEC project social network structure evolution, simulations were run on two different types of initial condition setups, representing small sized and real-world sized AEC project team social network structures. All the experiments suggested that, both node characteristic for seeking homophily/heterophily and popularity seeking correlated to the evolution of AEC project network structure overtime. The results showed that the node and dyadic characteristics and network size influence an AEC social network structure evolution. Moreover, using eigenvector centrality measure to define a node's popularity was found more practical than using degree centrality for the same. The key findings and their applications are further discussed in detail in the next chapter.

## Chapter 5 DISCUSSION AND CONCLUSION

### 5.1 Introduction

Through this chapter the author presents the summary of findings and then discuss their theoretical implications and practical application in AEC domain. The researcher then states the research deliverables. Finally, the author concludes the current chapter by presenting some limitations to this study and a few recommendations for future research.

### 5.2 Summary of Findings

The key findings of this research are as follows:

1. Node (e.g., tiers, roles, expertise similarity, and years working in AEC Industry) and dyadic characteristics (e.g., homophily, heterophily, and popularity seeking) influence an AEC social network structure evolution.
2. An agent-based model can be used to simulate an AEC project team social network structure to predict its evolution.
3. Network size also influence the evolution of AEC project teams social network structures, more specifically the formation of broken/unbroken subgroups as the network evolve.
4. Using eigenvector centrality measure to define a node's popularity is more practical than using degree centrality for the same.



### 5.3 Discussions

The various network structures that emerge out over time by changing the network size, domain difference percentage, and alpha (factor for homophily/heterophily), provides the following theoretical explanation and practical implications of the results for AEC research domain and AEC project teams:

1. Node (e.g., tiers, roles, expertise similarity, and years working in AEC Industry) and dyadic characteristics (e.g., homophily, heterophily, and popularity seeking) influence an AEC social network structure evolution.

Theoretical explanation and implication: As homophily in the model increases, project participants (nodes) increasingly seek connections only with similar other based on either role, tier or, expertise similarity. This creates polarized groups within the organization communication network. Polarization at high value of homophily seeking aligns with the literature (Frank and Fahrback, 1999; Frank and Xu, 2020). The polarization can prove detrimental to the project success as the participants no longer communicate with the knowledgeable (popular) others, as they have traded their connections from popular other to connect to only homophilious others.

On the other hand, as heterophily in the model increases, project participants (nodes) increasingly seek connections only with dissimilar others based on either role, tier or, expertise similarity. Although the simulation reveals two core-periphery-and-triangle structure bridged together at high heterophily value but the structure never polarizes (breaks). This implies that the heterophily seeking characteristic holds back the network structure from breaking apart into detached subgroups. This is just opposite to what homophily as a characteristic have an effect on the network structure, as its increment

breaks the structure in completely detached subgroups in the simulation. In real world both homophily and heterophily characteristics coexist in networks. That is, team members seek connections with similar others (e.g., same roles, and tiers) at the time of team coordination, while they seek connections with dissimilar others when they need access to new information to resolve an issue that is outside of their expertise and knowledge (Parraguez et al. 2015).

Simulation findings of homophily and heterophily seeking characteristic in isolation displayed what we observe in real life when these characteristics have a combined effect on evolution of a network structure. That is, the observation of sparsely connected final subgroups as the final structure in the empirical study (Garcia et al., 2020a and b). Here, subgroups are seen potentially due to individuals' homophily seeking but they are not seen as completely detached subgroups potentially due to the presence of heterophily seeking characteristic.

Practical applications: AEC project team leaders should be extra cautious about the attributes of the people they select or manage for each project. Based on how they want the team communication network to perform they should choose some team players with similar attributes. Team members with similar attributes (e.g., similar roles, expertise similarity, and years working in AEC industry) will catalyze and increase cohesion among the group members. However, a few members having different/varied expertise knowledge should also be a part of the team as their presence can insure that the network do not break into completely independent subgroups. The presence of varied expertise creates an inflow of new knowledge within the team. This is because the probability of circulation of redundant information (the knowledge that is common to each team member) decreases

when a team has members with different node attributes (e.g., different tiers, different roles, expertise dissimilarity, and years working in AEC Industry). For instance, a team containing members of different types of roles like architects and construction managers will have a better chance of resolving an issue about a construction change order than a team containing only one type of role like only architects. This is true as the team with members from a common role will mostly have the knowledge that is common to each team member thus the knowledge sharing become redundant for the team members. On the contrary the diverse by role team have members that have varied knowledge thus the knowledge sharing is fruitful and often result in the resolution of the issue at hand.

2. An ABM can be used to simulate an AEC project team social network structure to predict its evolution.

Theoretical explanation and implication: As the investigator found similarity between the simulation resultant network structures and the empirical study network structure evolutions (Garcia et al., 2020a and b), furthermore, the simulation models passed all the research quality tests (Richiardi et al., 2006; Sargent, 2013), it was concluded that ABM simulation is well equipped to emulate AEC project team social network structure evolution.

Practical applications: Research in AEC domain can use the developed model to simulate any other project teams' initial/intermediate social network structure to predict their evolution based on node and dyadic characteristics. They can also study and compare the social network evolution results produced by the current model with the results from an actual longitudinal study over the same initial social network structure setting.

3. Network size also influence the evolution of AEC project teams social network structures, more specifically the formation of broken/unbroken subgroups as the network evolves.

Theoretical explanation and implication: With an increase in network size the probability of bridging ties also increases. The bridging ties are responsible for not letting a network structure break into completely separate subgroups. Thus, with increased number of bridging ties it becomes relatively rare that a network evolves into completely separated subgroups. This aligns with the final evolved structure observed in the empirical study (Garcia et al., 2020a and b).

Practical applications: Big organizational team (comparable to the team size containing 80 members or more) have a rare chance of polarization. Project team leaders should be extra cautious in forming a small team out of members having varied attributes (e.g., different tiers, different roles, expertise dissimilarity, and years working in AEC Industry) because with a lower possibility of a bridging member (those members that keeps a network structure from breaking apart completely) found within the team, the team can break apart overtime and the common goal (e.g., project goals, issue resolutions during project delivery, within budget and on time delivery of project, etc.) could never be achieved in time in such a scenario. For instance, in a scenario where during a project delivery a member connected with two members, an architect and a contractor, can mediate and resolve contention between the two parties over a type of solution. The probability of the presence/emergence of such a mutually connected member increases with an increase in size of the connected group.

4. Using eigenvector centrality measure to define a node's popularity is more practical than using degree centrality for the same.

Theoretical explanation and implication: Literature suggests that, while assigning a popularity value to a node (member), degree centrality measure only consider the number of connected nodes (members) to the node in consideration. Thus, the calculation takes into account of only level one connections. The measure does not consider the importance of the possible deeper levels of connections, i.e., further connections of the connections to the considered node and so on. Eigenvector, however, take these factors into account. It provides a relative importance rank to each node within the network structure accordingly. Moreover, the research found more number of instance where the simulation provided final network structure matched to that found in the empirical study when eigenvector was considered as a measure of popularity instead of degree centrality. Thus it was concluded that eigenvector centrality more effectively represents a team member's popularity as compared to degree centrality.

Practical applications: Research in AEC domain can further study the effects of studying the node characteristics of popularity seeking in terms of eigenvector centrality to explore the effects either by using ABM methodologies or other modelling techniques.

Clear from the theoretical explanation above, degree centrality measure cannot entirely represent the reason for a member being popular (important) within a social network. Thus, AEC teams should be cautious in using the findings of such research that uses degree centrality as a measure of a team member's popularity (importance). In real world, connecting to a popular member means connecting to an important team member to gain new knowledge. For example: A lower organizational tier member such as a project

engineer connecting to a higher organizational tier member such as a senior construction manager to gain new knowledge. Here, senior construction manager is likely the popular node in the network stemming from its high centrality (eigenvector centrality) in the network (receiving/ disseminating information to the whole team). Thus, the research that uses degree centrality to grade if the members popular or not, assumes that the most important (popular) member in the organizational network is the one connected with maximum number of other members within the organization (the member with the highest degree centrality), which is an incorrect way to grade the importance of a member. The measure does not consider the importance of the possible deeper levels of connections, i.e., further connections of the connections to the considered member and so on. Eigenvector, however, take these factors into account. It provides a relative importance rank to each member within the network structure accordingly.

## 5.4 Conclusion

Working towards the goal of research, “exploration of the evolution of network structures of AEC project teams based on team member/node characteristics (i.e., expertise, roles, and tiers) and dyadic (homophily, heterophily, and popularity seeking) characteristics,” three objectives were established:

- (1) Review of literature and study of empirical data to identify node characteristics key to project team networks and their structures: This objective is accomplished via a literature review presented in Chapter 2. Several nodes, dyad, and network level characteristics and structures were studied. SNA studies both in AEC and Non-AEC domain were gleaned to find the existing methodology used in this research. It was discussed that literature till now has suggested several nodes (i.e., expertise, roles, and tiers) and dyadic (i.e., homophily, heterophily, and popularity seeking) characteristics that can influence the evolution of AEC project social network structure.
- (2) The second objective ‘developing and using an ABM approach to study the evolution of AEC project team networks through a computer-based simulation’ was fulfilled in Chapter-3 Methodology. Using selection model part from Frank and Xu (2020) and simulation initial condition reference point from Garcia et al. (2020a and b) an ABM was developed.
- (3) The third and final objective of ‘developing theoretical and practical implications’ were described in the aforementioned Discussion subsection of the current Chapte-5 Discussion and Conclusion.

The principal deliverable of this research is to present the effects of node and dyadic characteristics over an AEC project team social network evolution overtime. Furthermore, this deliverable is complemented by the following ones:

- 1) Verification of trends observed in an empirical study (Garcia et al., 2020a and b) showing that the network structures are dynamic, and they can evolve from core-periphery to cohesive subgroups in early stages of design during project delivery;
- 2) Practical applications in AEC domain future research and AEC project teams for improved communication and management of AEC project networks (section 5.2, and 5.3);
- 3) Insights of variables that can have an impact on AEC network structure (section 5.2, and 5.3); and
- 4) An ABM simulation based Netlogo model that can be used for future research (section 3.4).

Main limitation of this research includes using a single empirical study, that was an AEC project delivered by Construction Management at Risk (CMR) project delivery method, to base the initial condition of the simulation experiments. Thus, the result can only conclude the findings under this domain.

Although node and dyadic characteristics were considered but external factors like events that can influence a social network structure evolution were not considered in the developed model to make it as parsimonious as possible. This was done to reduce the noise and understand the most important influencers in the resultant evolved networks, however since we now know how the model performs these external factors could be included in the model by extending it easily.



The author presents the following recommendations for future research:

- 1) Future ABM simulation-based approach should take into account several empirical studies having AEC projects delivered via a different project delivery method to compare the network evolution within such delivery method. By performing such a study the researcher can analyze if and how project delivery method can influence the project teams' social network structure evolution.
- 2) In addition, the current model is yet in need of development in several aspects. One area of improvement is to enable agent's 'memory' and 'learning capabilities' from its interactions. Such capability will offer agents a new dependency in terms of each interaction and next decision made.
- 3) Moreover, the current model can be extended by populating an event/events within the model. The event/events, same as in real life, can initiate an interaction between nodes holding different types of node characteristics. It would be interesting to observe the evolution of network structure based on type of event and similar/dissimilar node characteristic.
- 4) Furthermore, it will be interesting to see if we can trade agents' homophily seeking characteristics versus their heterophily seeking characteristics at once, within a model. The author believes that such a simulation experiment might provide interesting insights, as in real world both these characteristics (homophily and heterophily seeking) are present simultaneously.

## APPENDICES

## APPENDIX A: Netlogo Code for Experiments

### Code for Experiments 1 and 2:

```
extensions [matrix]

globals [infinity average-path-length]

breed[core turtle1]

breed[periphery turtle2]

turtles-own [ indegree domain information self-identity safety behavior behavior0 new distance-from-
other-turtles utility degree node-cc neighborhood out-degree] ;; values and variables owned by turtles

links-own [information2 information3 old] ;; used as indicator for comparison


to setup
  ca

  setup-core-nodes

  setup-periphery-nodes

  setup-core-periphery-network

  set infinity 99999

  find-path-lengths ; creating a list for each turtle that tells the distance of all the rest of the turtles from itself

  calculate-utility ; using eq 6 from the paper

  ;ask turtles

  ;[
    ;set indegree count in-link-neighbors

    ;set indegree0 indegree

    ;set deviance-from-mean abs (behavior - mean [behavior] of turtles)
  ]

  ;calculate-exposure ;

  ;set average-path-length 1 ; setting the starting point of calculation of avg path length

  reset-ticks

end
```

```

to setup-core-nodes
  set-default-shape turtles "star"
  create-core number-of-core-nodes
  ask core [set color white]
  ask core [
    ; for visual reasons, we want core in center
    setxy (random-xcor * 0.01) (random-ycor * 0.01)
    set size 1.3
    set domain 0
    if domain-difference-percentage = 25
    [
      Let u 0
      while [u < (number-of-core-nodes / 4)]
      [ ask turtle1 u [ set domain 1 ]
        set u u + 1]
    ]
    if domain-difference-percentage = 50
    [
      Let v 0
      while [v < (number-of-core-nodes / 2)]
      [ ask turtle1 v [ set domain 1 ]
        set v v + 1]
    ]

    ;set information n-values 3 [random 15];; set initial information each turtle has
    ;set information lput n-values 2 [random -15] information
    ;set information reduce sentence information ;; eliminate the redundant information
    ;set information shuffle information
    set self-identity 1 - k-parameter * alpha
    set safety alpha
  ]

```

```

;set behavior random-normal (behavior-mean + 2 * behavior-sd) behavior-sd
; set behavior behavior + 0.5 * behavior-sd
]
end
to setup-periphery-nodes
  set-default-shape turtles "circle"
  create-periphery number-of-periphery-nodes
  ask periphery [set color white]
  ask periphery
  [
    ; for visual reasons, we don't put any nodes *too* close to the edges
    setxy (random-xcor * 0.95) (random-ycor * 0.95)
    set domain 0
    if domain-difference-percentage = 25
    [
      Let u 0
      while [u < (number-of-periphery-nodes / 4)]
      [ ask turtle2 (number-of-core-nodes + u) [ set domain 1 ]
        set u u + 1]
    ]
    if domain-difference-percentage = 50
    [
      Let v 0
      while [v < (number-of-periphery-nodes / 2)]
      [ ask turtle2 (number-of-core-nodes + v) [ set domain 1 ]
        set v v + 1]
    ]

;set information n-values 2 [random 15];; set initial information each turtle has
;set information lput n-values 3 [random -15] information

```

```

;set information reduce sentence information ;; eliminate the redundant information

;set information shuffle information

;set self-identity 1 - k-parameter * alpha

set safety alpha

;set behavior random-normal (behavior-mean + 2 * behavior-sd) behavior-sd

;set behavior behavior + 0.5 * behavior-sd

]

end

to setup-core-periphery-network

ask turtles with [domain = 1] [set color red]

let num-links1 (p-within-core * number-of-core-nodes * (number-of-core-nodes - 1)) ;number of links
among Core

let num-links2 (p-within-periphery * number-of-periphery-nodes * (number-of-periphery-nodes - 1))
;number of links among periphery

let num-links3 (p-between-core-periphery * 2 * number-of-core-nodes * number-of-periphery-nodes )
;number of links between core & periphery

; if average-node-degree > min [number-of-sub1 - 1 number-of-sub2 - 1]

; [set average-node-degree min [number-of-nodes - 1 number-of-sub2 - 1]

; stop]

while [count links < num-links1 ]

[
ask one-of core

[
let choice (one-of other core with [not in-link-neighbor? myself ] ; "myself" here refers to "one-of other
core"

)

if choice != nobody [ create-link-to choice ] ;; "choice" is "one-of other core"

]
]

```

```

]
while [ (num-links1 <= count links) and (count links < num-links1 + num-links2) ]
[
  ask one-of periphery
  [
    let choice (one-of other periphery with [not in-link-neighbor? myself ]
      )
    if choice != nobody [ create-link-to choice ]
  ]
]
while [(count links >= num-links1) and (count links < num-links1 + num-links3)]
[
  ask one-of turtles
  [
    ifelse breed = core

    [ let choice (one-of other turtles with [(breed = periphery) and (not in-link-neighbor? myself ) ])
      if choice != nobody [ create-link-to choice ]
    ]

    [
      let choice (one-of other turtles with [(breed = core) and (not in-link-neighbor? myself ) ])
      if choice != nobody [ create-link-to choice ]
    ]
  ]
]

if p-within-core != 0 ; this is to make sure every core turtle is connected to at least one other core turtle
[
  ask core with [count out-link-neighbors = 0] [create-link-to one-of other core]
]

```

if p-between-core-periphery != 0 ; this is to make sure every periphery turtle is connected to at least one core turtle

```
[  
  ask periphery with [count out-link-neighbors = 0] [create-link-to one-of other core]  
]
```

repeat 2 ; lay out core

```
[  
  layout-spring core links 0.6 0.5 1 ; turtle-set link-set spring-constant spring-length repulsion-  
  constant  
]
```

repeat 100 ; lay out periphery

```
[  
  layout-spring periphery links 0.6 5 50 ; turtle-set link-set spring-constant spring-length repulsion-  
  constant  
]
```

ask links

```
[  
  set information2 n-values 2 [random 0]  
  set information2 remove 0 information2  
  set information3 n-values 2 [random 0]  
  set information3 remove 0 information3  
  set old 0  
] ;; set information2 and information3 as empty list
```

end

to find-path-lengths ; creating a list for each turtle that tells the distance of all the rest of the turtles from itself

```
ask turtles  
[ set distance-from-other-turtles []  
]  
let i 0  
let j 0
```



```

let k 0

let node1 one-of turtles

let node2 one-of turtles

let node-count count turtles

;; initialize the distance lists
while [i < node-count]
[
  set j 0
  while [j < node-count]
  [
    set node1 turtle i
    set node2 turtle j
    ifelse i = j ;; zero from a node to itself
    [
      ask node1
      [ set distance-from-other-turtles lput 0 distance-from-other-turtles]
    ]
    [ ;; 1 from a node to it's neighbor
      ifelse [ in-link-neighbor? node1 or out-link-neighbor? node1] of node2
      [
        ask node1 [set distance-from-other-turtles lput 1 distance-from-other-turtles]
      ]
      [
        ask node1 [set distance-from-other-turtles lput infinity distance-from-other-turtles]
      ]
    ]
    set j j + 1
  ]
  set i i + 1
]

```

```

set i 0
set j 0
let dummy 0
while [k < node-count]
[
  set i 0
  while [i < node-count]
  [
    set j 0
    while [j < node-count]
    [
      ;; alternate path length through kth node
      set dummy ( (item k [distance-from-other-turtles] of turtle i) +
                  (item j [distance-from-other-turtles] of turtle k)) ;
      ;; is the alternate path shorter?
      if dummy < (item j [distance-from-other-turtles] of turtle i)
      [
        ask turtle i [
          set distance-from-other-turtles replace-item j distance-from-other-turtles dummy ;; replace distance
if shorter
        ]
      ]
      set j j + 1
    ]
    set i i + 1
  ]
  set k k + 1
]
end

```

to calculate-utility ;; this calculate a list for each turtle, containing utility for connecting to each other turtle

```
ask turtles [ set utility []]
```

```
let m 0
```

```
while [m < count turtles]
```

```
[
```

```
  ask turtle m
```

```
  [
```

```
    let n 0
```

while [n < length distance-from-other-turtles] ;Reports the number of items in the given list, or the number of characters in the given string.

```
  [
```

```
    ifelse item n distance-from-other-turtles = 0 ;; distance to another turtle is 0 or infinity
```

```
    [ let p -999999
```

```
      set utility lput p utility
```

```
    ]
```

```
  [ ifelse item n distance-from-other-turtles > 10000
```

```
    [ let p count [in-link-neighbors] of turtle n * (1 - safety) - safety * (abs ([domain] of turtle m - [domain] of turtle n)) + random-float 0.00000001
```

```
      set utility lput p utility ; utility connecting to itself or turtle it has no access to = minus infinity
```

```
    ]
```

```
  [
```

```
    let p ((1 - safety) * tau * count [in-link-neighbors] of turtle n + (item n distance-from-other-turtles - 1) * (1 - safety) * (1 - tau )) - safety * (abs ([domain] of turtle m - [domain] of turtle n)) + random-float 0.00000001 ;; utility function, consisted of reducing maximal path length and homophily
```

```
      set utility lput p utility
```

```
    ]
```

```
  ]
```

```
  set n n + 1
```

```
]
```

```
]
```

```
set m m + 1
```

```

]
end

to go
;calculate-exposure
;ask turtles [set behavior0 behavior]
find-path-lengths
calculate-utility
;information-seeking ; coming from paper eq 7 ; social influence model eq
;;change-ties

calculate-apl; calculation of average path length for plotting
;find-neighborhood
;calculate-cc ;
;calculate-global-clustering-coefficient

change-path-length
ask turtles [set indegree count in-link-neighbors] ; to plot indegree of turtles
;set deviance-from-mean abs (behavior - mean [behavior] of turtles)
;]
;calculate-correlation
;calculate-sum-score
; influence
;ask turtles
;[
;elseifelse show-information?
;[set label length information]
;[set label ""]
;]
; if min [length information] of turtles = max [length information] of turtles [ stop ]

```

```

if ticks = 50 [
  ; nw:set-context turtles links
  ;ask turtles [ set ec nw:eigenvector-centrality]
  ;calculate-skewness
  ; stop
  ; nw:set-context turtles links
  ; nw:save-matrix "xx.csv"
  ;export-network
  stop
]
; if ticks > 50 [ask turtles [set safety 0.9]]
tick
end

```

to calculate-apl

let num-connected-pairs sum [length remove infinity (remove 0 distance-from-other-turtles)] of turtles;  
removes "zero" then "infinity" from "distance-from-other-turtle" list. then "length" gives the number of  
items in the list

;; In a connected network on N nodes, we should have  $N(N-1)$  measurements of distances between pairs,

;; and none of those distances should be infinity.

;; If there were any "infinity" length paths between nodes, then the network is disconnected.

;; In that case, calculating the average-path-length doesn't really make sense.

ifelse ( num-connected-pairs != (count turtles \* (count turtles - 1) )) ;

[ let connected? true

set average-path-length infinity

;; report that the network is not connected

set connected? false

]

[

set average-path-length (sum [sum distance-from-other-turtles] of turtles) / (num-connected-pairs) ; first sum all the link distances of a turtle then multiply with the number of turtles. then divide by number of connected pairs.

```
]
end
```

to change-path-length

```
ask turtles [set out-degree count out-link-neighbors]
```

```
ask links [die]
```

```
let m 0
```

```
while [ m < count turtles ]
```

```
[
```

```
ask turtle m
```

```
[
```

```
; let n 0
```

```
; let o count out-link-neighbors
```

```
; while [n < count out-link-neighbors]
```

```
;[ let x item n sort out-link-neighbors
```

```
  ;let xx random-float 1
```

```
  ;if ((0.8 ^ ([old] of link m [who] of x)) < xx ) ; no new information provided in 3 ticks
```

```
  ;[
```

```
  ; ask link m [who] of x [die]
```

```
; ]
```

```
;set n n + 1
```

```
; ]
```

```
let p 0 ;; number of ties haven't been deleted( which means these ties provide new information in past 3 ticks)
```

```
let q 0
```

```
while [p < out-degree]
```

```

[if item q (sort-by > utility) > -999999 and (not out-link-neighbor? turtle (position (item q (sort-by > utility))
utility)) ; turtle with max utility and the one to which the ego-turtle is not connected to

[create-link-to turtle (position (item q (sort-by > utility)) utility)] ;;position (item q (sort-by > utility))
utility is to find number of turtle with the max utility to the ego

;ask links with [information2 = 0]

;[set information2 []

;set information3 []

;set old 0]

set p count out-link-neighbors ;; remain number of neighbors as constant

set q q + 1 ]

]

set m m + 1]

if ticks mod 3 = 0

[repeat 100 [

  layout-spring turtles links 0.6 3 10 ;0.6 3 30 ; 0.6 0.5 1 ; 0.6 5 25 ; spring-constant  spring-length
repulsion-constant

  ;layout-spring turtles links 0.3 (world-width / (sqrt number-of-nodes)) 1

  display]]

end

```

### Code for Experiments 3 and 4:

```
extensions [nw]

globals [infinity average-path-length]

breed[core turtle1]

breed[periphery turtle2]

turtles-own [ indegree domain information self-identity safety behavior behavior0 new distance-from-
other-turtles utility degree node-cc neighborhood out-degree] ;; values and variables owned by turtles

links-own [information2 information3 old] ;; used as indicator for comparison

to setup

  ca

  setup-core-nodes

  setup-periphery-nodes

  setup-core-periphery-network

  set infinity 99999

  find-path-lengths ; creating a list for each turtle that tells the distance of all the rest of the turtles from itself

  calculate-utility ; using eq 6 from the paper

  ;ask turtles

  ;[

    ;set indegree count in-link-neighbors

    ;set indegree0 indegree

    ;set deviance-from-mean abs (behavior - mean [behavior] of turtles)

  ;]

  ;calculate-exposure

  ;set average-path-length 1 ; setting the starting point of calculation of avg path length

  reset-ticks

end

to setup-core-nodes

  set-default-shape turtles "star"

  create-core number-of-core-nodes

  ask core [set color white]
```



```

ask core [
; for visual reasons, we want core in center
setxy (random-xcor * 0.01) (random-ycor * 0.01)
set size 1.3
set domain 0
if domain-difference-percentage = 25
[
Let u 0
while [u < (number-of-core-nodes / 4)]
[ ask turtle1 u [ set domain 1 ]
set u u + 1]
]
if domain-difference-percentage = 50
[
Let v 0
while [v < (number-of-core-nodes / 2)]
[ ask turtle1 v [ set domain 1 ]
set v v + 1]
]

;set information n-values 3 [random 15];; set initial information each turtle has
;set information lput n-values 2 [random -15] information
;set information reduce sentence information ;; eliminate the redundant information
;set information shuffle information
set self-identity 1 - k-parameter * alpha
set safety alpha
;set behavior random-normal (behavior-mean + 2 * behavior-sd) behavior-sd
; set behavior behavior + 0.5 * behavior-sd
]
end

```

```

to setup-periphery-nodes
  set-default-shape turtles "circle"
  create-periphery number-of-periphery-nodes
  ask periphery [set color white]
  ask periphery
  [
    ; for visual reasons, we don't put any nodes *too* close to the edges
    setxy (random-xcor * 0.95) (random-ycor * 0.95)
    set domain 0
    if domain-difference-percentage = 25
    [
      Let u 0
      while [u < (number-of-periphery-nodes / 4)]
      [ ask turtle2 (number-of-core-nodes + u) [ set domain 1 ]
        set u u + 1]
    ]
    if domain-difference-percentage = 50
    [
      Let v 0
      while [v < (number-of-periphery-nodes / 2)]
      [ ask turtle2 (number-of-core-nodes + v) [ set domain 1 ]
        set v v + 1]
    ]

    ;set information n-values 2 [random 15];; set initial information each turtle has
    ;set information lput n-values 3 [random -15] information
    ;set information reduce sentence information ;; eliminate the redundant information
    ;set information shuffle information
    ;set self-identity 1 - k-parameter * alpha
    set safety alpha
  ]

```

```

;set behavior random-normal (behavior-mean + 2 * behavior-sd) behavior-sd

;set behavior behavior + 0.5 * behavior-sd

]
end

to setup-core-periphery-network

  ask turtles with [domain = 1] [set color red]

  let num-links1 (p-within-core * number-of-core-nodes * (number-of-core-nodes - 1)) ;number of links
  among Core

  let num-links2 (p-within-periphery * number-of-periphery-nodes * (number-of-periphery-nodes - 1))
  ;number of links among periphery

  let num-links3 (p-between-core-periphery * 2 * number-of-core-nodes * number-of-periphery-nodes )
  ;number of links between core & periphery

  ; if average-node-degree > min [number-of-sub1 - 1 number-of-sub2 - 1]
  ; [set average-node-degree min [number-of-nodes - 1 number-of-sub2 - 1]
  ; stop]

  while [count links < num-links1 ]
  [
    ask one-of core
    [
      let choice (one-of other core with [not in-link-neighbor? myself ] ; "myself" here refers to "one-of other
      core"

      )

      if choice != nobody [ create-link-to choice ] ;; "choice" is "one-of other core"
    ]
  ]

  while [ (num-links1 <= count links) and (count links < num-links1 + num-links2) ]
  [
    ask one-of periphery
    [

```

```

    let choice (one-of other periphery with [not in-link-neighbor? myself ]
    )
    if choice != nobody [ create-link-to choice ]
  ]
]
while [(count links >= num-links1) and (count links < num-links1 + num-links3)]
[
  ask one-of turtles
  [
    ifelse breed = core

    [ let choice (one-of other turtles with [(breed = periphery) and (not in-link-neighbor? myself ) ])
      if choice != nobody [ create-link-to choice ]

    ]
    [
      let choice (one-of other turtles with [(breed = core) and (not in-link-neighbor? myself ) ])
      if choice != nobody [ create-link-to choice ]
    ]
  ]
]
if p-within-core != 0 ; this is to make sure every core turtle is connected to at least one other core turtle
[
  ask core with [count out-link-neighbors = 0] [create-link-to one-of other core]
]

if p-between-core-periphery != 0 ; this is to make sure every periphery turtle is connected to at least one
core turtle
[
  ask periphery with [count out-link-neighbors = 0] [create-link-to one-of other core]
]

```

```

]
repeat 2 ; lay out core
[
  layout-spring core links 0.6 0.5 1 ; turtle-set link-set spring-constant spring-length repulsion-
constant
]
repeat 100 ; lay out periphery
[
  layout-spring periphery links 0.6 5 50 ; turtle-set link-set spring-constant spring-length repulsion-
constant
]
ask links
[
  set information2 n-values 2 [random 0]
  set information2 remove 0 information2
  set information3 n-values 2 [random 0]
  set information3 remove 0 information3
  set old 0
] ;; set information2 and information3 as empty list
end

to find-path-lengths ; creating a list for each turtle that tells the distance of all the rest of the turtles from
itself
ask turtles
[ set distance-from-other-turtles []
]
let i 0
let j 0
let k 0
let node1 one-of turtles
let node2 one-of turtles
let node-count count turtles

```

```

;; initialize the distance lists
while [i < node-count]
[
  set j 0
  while [j < node-count]
  [
    set node1 turtle i
    set node2 turtle j
    ifelse i = j ;; zero from a node to itself
    [
      ask node1
      [ set distance-from-other-turtles lput 0 distance-from-other-turtles]
    ]
    [ ;; 1 from a node to it's neighbor
      ifelse [ in-link-neighbor? node1 or out-link-neighbor? node1] of node2
      [
        ask node1 [set distance-from-other-turtles lput 1 distance-from-other-turtles]
      ]
      [
        ask node1 [set distance-from-other-turtles lput infinity distance-from-other-turtles]
      ]
    ]
    set j j + 1
  ]
  set i i + 1
]
set i 0
set j 0
let dummy 0
while [k < node-count]

```

```

[
  set i 0
  while [i < node-count]
  [
    set j 0
    while [j < node-count]
    [
      ;; alternate path length through kth node
      set dummy ( (item k [distance-from-other-turtles] of turtle i) +
                  (item j [distance-from-other-turtles] of turtle k))
      ;; is the alternate path shorter?
      if dummy < (item j [distance-from-other-turtles] of turtle i)
      [
        ask turtle i [
          set distance-from-other-turtles replace-item j distance-from-other-turtles dummy ;; replace distance
if shorter
        ]
      ]
      set j j + 1
    ]
    set i i + 1
  ]
  set k k + 1
]
end

to calculate-utility ;; this calculate a list for each turtle, containing utility for connecting to each other turtle
ask turtles [ set utility []]
let m 0
while [m < count turtles]
[

```

```

ask turtle m
[
  let n 0

  while [n < length distance-from-other-turtles] ;Reports the number of items in the given list, or the
number of characters in the given string.
  [
    ifelse item n distance-from-other-turtles = 0 ;; distance to another turtle is 0 or infinity
    [ let p -999999
      set utility lput p utility
    ]
    [ ifelse item n distance-from-other-turtles > 10000
      [
        let ec [nw:eigenvector-centrality] of turtle n
        ifelse [ec] of turtle n = false
        [
          let p 0 * (1 - safety) - safety * (abs ([domain] of turtle m - [domain] of turtle n)) + random-float
0.00000001

          set utility lput p utility ; utility connecting to itself or turtle it has no access to = minus infinity
        ]
        [
          let p [nw:eigenvector-centrality] of turtle n * (1 - safety) - safety * (abs ([domain] of turtle m -
[domain] of turtle n)) + random-float 0.00000001

          set utility lput p utility ; utility connecting to itself or turtle it has no access to = minus infinity
        ]
      ]
    ]
    [
      let ec [nw:eigenvector-centrality] of turtle n
      ifelse [ec] of turtle n = false
      [
        let p ((1 - safety) * tau * 0 + (item n distance-from-other-turtles - 1) * (1 - safety) * (1 - tau )) -
safety * (abs ([domain] of turtle m - [domain] of turtle n)) + random-float 0.00000001 ;; utility function,
consisted of reducing maximal path length and homophily

```



```

        set utility lput p utility
    ]
    [
        let p ((1 - safety) * tau * [nw:eigenvector-centrality] of turtle n + (item n distance-from-other-turtles
- 1) * (1 - safety) * (1 - tau )) - safety * (abs ([domain] of turtle m - [domain] of turtle n)) + random-float
0.00000001 ;; utility function, consisted of reducing maximal path length and homophily
        set utility lput p utility
    ]
]
set n n + 1
]
]
set m m + 1
]
end

```

```

to go
;calculate-exposure
;ask turtles [set behavior0 behavior]
find-path-lengths
calculate-utility
;information-seeking ; coming from paper eq 7 ; social influence model eq
;;change-ties
calculate-apl; calculation of average path length for plotting
;find-neighborhood
;calculate-cc
;calculate-global-clustering-coefficient

change-path-length
ask turtles [set indegree count in-link-neighbors] ; to plot indegree of turtles

```

```

        ;set deviance-from-mean abs (behavior - mean [behavior] of turtles)
    ;]
;calculate-correlation
;calculate-sum-score
;influence
;ask turtles
;[
    ;ifelse show-information?
    ;[set label length information]
    ;[set label ""]
;]
; if min [length information] of turtles = max [length information] of turtles [    stop ]
if ticks = 50 [
    ; nw:set-context turtles links
    ;ask turtles [ set ec nw:eigenvector-centrality]
    ;calculate-skewness
    ; stop
    ; nw:set-context turtles links
    ; nw:save-matrix "xx.csv"
    ;export-network
    stop
]
; if ticks > 50 [ask turtles [set safety 0.9]]
tick
end

```

to calculate-apl

let num-connected-pairs sum [length remove infinity (remove 0 distance-from-other-turtles)] of turtles;  
removes "zero" then "infinity" from "distance-from-other-turtle" list. then "length" gives the number of  
items in the list

```

;; In a connected network on N nodes, we should have N(N-1) measurements of distances between pairs,
;; and none of those distances should be infinity.
;; If there were any "infinity" length paths between nodes, then the network is disconnected.
;; In that case, calculating the average-path-length doesn't really make sense.
ifelse ( num-connected-pairs != (count turtles * (count turtles - 1) ) ) ;
[ let connected? true
  set average-path-length infinity
  ;; report that the network is not connected
  set connected? false
]
[
  set average-path-length (sum [sum distance-from-other-turtles] of turtles) / (num-connected-pairs) ; first
  sum all the link distances of a turtle then multiply with the number of turtles. then divide by number of
  connected pairs.
]
]
End

to change-path-length
ask turtles [set out-degree count out-link-neighbors]
ask links [die]
let m 0
while [ m < count turtles ]
[
  ask turtle m
  [
    ; let n 0
    ; let o count out-link-neighbors
    ; while [n < count out-link-neighbors]
    ;[ let x item n sort out-link-neighbors
    ;let xx random-float 1
    ;if ((0.8 ^ ([old] of link m [who] of x)) < xx ) ; no new information provided in 3 ticks
    ;[

```

```

; ask link m [who] of x [die]
; ]
;set n n + 1
; ]

let p 0 ;; number of ties haven't been deleted( which means these ties provide new information in past 3
ticks)

let q 0
while [p < out-degree]

[if item q (sort-by > utility) > -999999 and (not out-link-neighbor? turtle (position (item q (sort-by > utility))
utility)) ; turtle with max utility and the one to which the ego-turtle is not connected to

[create-link-to turtle (position (item q (sort-by > utility)) utility)] ;;position (item q (sort-by > utility))
utility is to find number of turtle with the max utility to the ego

;ask links with [information2 = 0]

;[set information2 []

;set information3 []

;set old 0]

set p count out-link-neighbors ;; remain number of neighbors as constant

set q q + 1 ]
]

set m m + 1]

if ticks mod 3 = 0
[repeat 100 [

layout-spring turtles links 0.6 3 10 ;0.6 3 30 ; 0.6 0.5 1 ; 0.6 5 25 ; spring-constant spring-length
repulsion-constant

;layout-spring turtles links 0.3 (world-width / (sqrt number-of-nodes)) 1

display]]
end

```

## Code for Experiments 5 and 6:

extensions [matrix]

globals [infinity average-path-length]

breed[core turtle1]

breed[periphery turtle2]

turtles-own [ indegree domain information self-identity safety behavior behavior0 new distance-from-other-turtles utility degree node-cc neighborhood out-degree] ;; values and variables owned by turtles

links-own [information2 information3 old] ;; used as indicator for comparison

to setup

ca

setup-core-nodes

setup-periphery-nodes

setup-core-periphery-network

set infinity 99999

find-path-lengths ; creating a list for each turtle that tells the distance of all the rest of the turtles from itself

calculate-utility ; using eq 6 from the paper

;ask turtles

:[

  ;set indegree count in-link-neighbors

  ;set indegree0 indegree

  ;set deviance-from-mean abs (behavior - mean [behavior] of turtles)

;]

;calculate-exposure ;

;set average-path-length 1 ; setting the starting point of calculation of avg path length

reset-ticks

end

```

to setup-core-nodes
  set-default-shape turtles "star"
  create-core number-of-core-nodes
  ask core [set color white]
  ask core [
    ; for visual reasons, we want core in center
    setxy (random-xcor * 0.01) (random-ycor * 0.01)
    set size 1.3
    set domain 0
    if domain-difference-percentage = 25
    [
      Let u 0
      while [u < (number-of-core-nodes / 4)]
      [ ask turtle1 u [ set domain 1 ]
        set u u + 1]
    ]
    if domain-difference-percentage = 50
    [
      Let v 0
      while [v < (number-of-core-nodes / 2)]
      [ ask turtle1 v [ set domain 1 ]
        set v v + 1]
    ]

    ;set information n-values 3 [random 15];; set initial information each turtle has
    ;set information lput n-values 2 [random -15] information
    ;set information reduce sentence information ;; eliminate the redundant information
    ;set information shuffle information
    set self-identity 1 - k-parameter * alpha
    set safety alpha
  ]

```

```

;set behavior random-normal (behavior-mean + 2 * behavior-sd) behavior-sd
; set behavior behavior + 0.5 * behavior-sd
]
end

to setup-periphery-nodes
  set-default-shape turtles "circle"
  create-periphery number-of-periphery-nodes
  ask periphery [set color white]
  ask periphery
  [
    ; for visual reasons, we don't put any nodes *too* close to the edges
    setxy (random-xcor * 0.95) (random-ycor * 0.95)
    set domain 0
    if domain-difference-percentage = 25
    [
      Let u 0
      while [u < (number-of-periphery-nodes / 4)]
      [ ask turtle2 (number-of-core-nodes + u) [ set domain 1 ]
        set u u + 1]
    ]
    if domain-difference-percentage = 50
    [
      Let v 0
      while [v < (number-of-periphery-nodes / 2)]
      [ ask turtle2 (number-of-core-nodes + v) [ set domain 1 ]
        set v v + 1]
    ]

;set information n-values 2 [random 15];; set initial information each turtle has
;set information lput n-values 3 [random -15] information

```

```

;set information reduce sentence information ;; eliminate the redundant information

;set information shuffle information

;set self-identity 1 - k-parameter * alpha

set safety alpha

;set behavior random-normal (behavior-mean + 2 * behavior-sd) behavior-sd

;set behavior behavior + 0.5 * behavior-sd

]

end

to setup-core-periphery-network

ask turtles with [domain = 1] [set color red]

let num-links1 (p-within-core * number-of-core-nodes * (number-of-core-nodes - 1)) ;number of links
among Core

let num-links2 (p-within-periphery * number-of-periphery-nodes * (number-of-periphery-nodes - 1))
;number of links among periphery

let num-links3 (p-between-core-periphery * 2 * number-of-core-nodes * number-of-periphery-nodes )
;number of links between core & periphery

; if average-node-degree > min [number-of-sub1 - 1 number-of-sub2 - 1]

; [set average-node-degree min [number-of-nodes - 1 number-of-sub2 - 1]

; stop]

while [count links < num-links1 ]

[
ask one-of core

[
let choice (one-of other core with [not in-link-neighbor? myself ] ; "myself" here refers to "one-of other
core"

)

if choice != nobody [ create-link-to choice ] ;; "choice" is "one-of other core"

]
]

```



```

]
while [ (num-links1 <= count links) and (count links < num-links1 + num-links2) ]
[
  ask one-of periphery
  [
    let choice (one-of other periphery with [not in-link-neighbor? myself ]
      )
    if choice != nobody [ create-link-to choice ]
  ]
]
while [(count links >= num-links1) and (count links < num-links1 + num-links3)]
[
  ask one-of turtles
  [
    ifelse breed = core

    [ let choice (one-of other turtles with [(breed = periphery) and (not in-link-neighbor? myself ) ])
      if choice != nobody [ create-link-to choice ]
    ]

    [
      let choice (one-of other turtles with [(breed = core) and (not in-link-neighbor? myself ) ])
      if choice != nobody [ create-link-to choice ]
    ]
  ]
]

if p-within-core != 0 ; this is to make sure every core turtle is connected to at least one other core turtle
[
  ask core with [count out-link-neighbors = 0] [create-link-to one-of other core]
]

```

if p-between-core-periphery != 0 ; this is to make sure every periphery turtle is connected to at least one core turtle

```
[  
  ask periphery with [count out-link-neighbors = 0] [create-link-to one-of other core]  
]
```

repeat 2 ; lay out core

```
[  
  layout-spring core links 0.6 0.5 1 ; turtle-set link-set spring-constant spring-length repulsion-  
  constant  
]
```

repeat 100 ; lay out periphery

```
[  
  layout-spring periphery links 0.6 5 50 ; turtle-set link-set spring-constant spring-length repulsion-  
  constant  
]
```

ask links

```
[  
  set information2 n-values 2 [random 0]  
  set information2 remove 0 information2  
  set information3 n-values 2 [random 0]  
  set information3 remove 0 information3  
  set old 0  
] ;; set information2 and information3 as empty list
```

end

to find-path-lengths ; creating a list for each turtle that tells the distance of all the rest of the turtles from itself

ask turtles

```
[ set distance-from-other-turtles []
```

```
]
```

let i 0

let j 0

```

let k 0
let node1 one-of turtles
let node2 one-of turtles
let node-count count turtles
;; initialize the distance lists
while [i < node-count]
[
  set j 0
  while [j < node-count]
  [
    set node1 turtle i
    set node2 turtle j
    ifelse i = j ;; zero from a node to itself
    [
      ask node1
      [ set distance-from-other-turtles lput 0 distance-from-other-turtles]
    ]
    [ ;; 1 from a node to it's neighbor
      ifelse [ in-link-neighbor? node1 or out-link-neighbor? node1] of node2
      [
        ask node1 [set distance-from-other-turtles lput 1 distance-from-other-turtles]
      ]
      [
        ask node1 [set distance-from-other-turtles lput infinity distance-from-other-turtles]
      ]
    ]
    set j j + 1
  ]
  set i i + 1
]

```

```

set i 0
set j 0
let dummy 0
while [k < node-count]
[
  set i 0
  while [i < node-count]
  [
    set j 0
    while [j < node-count]
    [
      ;; alternate path length through kth node
      set dummy ( (item k [distance-from-other-turtles] of turtle i) +
                    (item j [distance-from-other-turtles] of turtle k)) ;
      ;; is the alternate path shorter?
      if dummy < (item j [distance-from-other-turtles] of turtle i)
      [
        ask turtle i [
          set distance-from-other-turtles replace-item j distance-from-other-turtles dummy ;; replace distance
if shorter
        ]
      ]
      set j j + 1
    ]
    set i i + 1
  ]
  set k k + 1
]
end

```

to calculate-utility ;; this calculate a list for each turtle, containing utility for connecting to each other turtle

```
ask turtles [ set utility []]
```

```
let m 0
```

```
while [m < count turtles]
```

```
[
```

```
  ask turtle m
```

```
  [
```

```
    let n 0
```

while [n < length distance-from-other-turtles] ;Reports the number of items in the given list, or the number of characters in the given string.

```
  [
```

```
    ifelse item n distance-from-other-turtles = 0 ;; distance to another turtle is 0 or infinity
```

```
    [ let p -999999
```

```
      set utility lput p utility
```

```
    ]
```

```
  [ ifelse item n distance-from-other-turtles > 10000
```

```
    [ let p count [in-link-neighbors] of turtle n * (1 - safety) + safety * (abs ([domain] of turtle m - [domain] of turtle n)) + random-float 0.00000001
```

```
      set utility lput p utility ; utility connecting to itself or turtle it has no access to = minus infinity
```

```
    ]
```

```
  [
```

```
    let p ((1 - safety) * tau * count [in-link-neighbors] of turtle n + (item n distance-from-other-turtles - 1) * (1 - safety) * (1 - tau )) + safety * (abs ([domain] of turtle m - [domain] of turtle n)) + random-float 0.00000001 ;; utility function, consisted of reducing maximal path length and homophily
```

```
      set utility lput p utility
```

```
    ]
```

```
  ]
```

```
  set n n + 1
```

```
]
```

```
]
```

```
set m m + 1
```

```

]
end

to go
;calculate-exposure
;ask turtles [set behavior0 behavior]
find-path-lengths
calculate-utility
;information-seeking ; coming from paper eq 7 ; social influence model eq
;;change-ties

calculate-apl; calculation of average path length for plotting
;find-neighborhood
;calculate-cc ;
;calculate-global-clustering-coefficient

change-path-length
ask turtles [set indegree count in-link-neighbors] ; to plot indegree of turtles
      ;set deviance-from-mean abs (behavior - mean [behavior] of turtles)
;]
;calculate-correlation
;calculate-sum-score
; influence
;ask turtles
;[
      ;ifelse show-information?
      ;[set label length information]
      ;[set label ""]
;]
; if min [length information] of turtles = max [length information] of turtles [ stop ]

```

```

if ticks = 50 [
  ; nw:set-context turtles links
  ;ask turtles [ set ec nw:eigenvector-centrality]
  ;calculate-skewness
  ; stop
  ; nw:set-context turtles links
  ; nw:save-matrix "xx.csv"
  ;export-network
  stop
]
; if ticks > 50 [ask turtles [set safety 0.9]]
tick
end

```

to calculate-apl

let num-connected-pairs sum [length remove infinity (remove 0 distance-from-other-turtles)] of turtles;  
removes "zero" then "infinity" from "distance-from-other-turtle" list. then "length" gives the number of  
items in the list

;; In a connected network on N nodes, we should have  $N(N-1)$  measurements of distances between pairs,

;; and none of those distances should be infinity.

;; If there were any "infinity" length paths between nodes, then the network is disconnected.

;; In that case, calculating the average-path-length doesn't really make sense.

ifelse ( num-connected-pairs != (count turtles \* (count turtles - 1) )) ;

[ let connected? true

set average-path-length infinity

;; report that the network is not connected

set connected? false

]

[

set average-path-length (sum [sum distance-from-other-turtles] of turtles) / (num-connected-pairs) ; first sum all the link distances of a turtle then multiply with the number of turtles. then divide by number of connected pairs.

```
]
end
```

to change-path-length

```
ask turtles [set out-degree count out-link-neighbors]
```

```
ask links [die]
```

```
let m 0
```

```
while [ m < count turtles ]
```

```
[
```

```
ask turtle m
```

```
[
```

```
; let n 0
```

```
; let o count out-link-neighbors
```

```
; while [n < count out-link-neighbors]
```

```
;[ let x item n sort out-link-neighbors
```

```
  ;let xx random-float 1
```

```
  ;if ((0.8 ^ ([old] of link m [who] of x)) < xx ) ; no new information provided in 3 ticks
```

```
  ;[
```

```
  ; ask link m [who] of x [die]
```

```
  ; ]
```

```
;set n n + 1
```

```
; ]
```

```
let p 0 ;; number of ties haven't been deleted( which means these ties provide new information in past 3 ticks)
```

```
let q 0
```

```
while [p < out-degree]
```



```

[if item q (sort-by > utility) > -999999 and (not out-link-neighbor? turtle (position (item q (sort-by > utility))
utility)) ; turtle with max utility and the one to which the ego-turtle is not connected to

[create-link-to turtle (position (item q (sort-by > utility)) utility)] ;;position (item q (sort-by > utility))
utility is to find number of turtle with the max utility to the ego

;ask links with [information2 = 0]

;[set information2 []

;set information3 []

;set old 0]

set p count out-link-neighbors ;; remain number of neighbors as constant

set q q + 1 ]

]

set m m + 1]

if ticks mod 3 = 0

[repeat 100 [

  layout-spring turtles links 0.6 3 10 ;0.6 3 30 ; 0.6 0.5 1 ; 0.6 5 25 ; spring-constant  spring-length
repulsion-constant

  ;layout-spring turtles links 0.3 (world-width / (sqrt number-of-nodes)) 1

  display]]

end

```

## Code for Experiments 7 and 8:

```
extensions [nw]

globals [infinity average-path-length]

breed[core turtle1]

breed[periphery turtle2]

turtles-own [ indegree domain information self-identity safety behavior behavior0 new distance-from-
other-turtles utility degree node-cc neighborhood out-degree] ;; values and variables owned by turtles

links-own [information2 information3 old] ;; used as indicator for comparison

to setup
  ca

  setup-core-nodes

  setup-periphery-nodes

  setup-core-periphery-network

  set infinity 99999

  find-path-lengths ; creating a list for each turtle that tells the distance of all the rest of the turtles from itself

  calculate-utility ; using eq 6 from the paper

  ;ask turtles

  ;[
    ;set indegree count in-link-neighbors

    ;set indegree0 indegree

    ;set deviance-from-mean abs (behavior - mean [behavior] of turtles)
  ;]

  ;calculate-exposure

  ;set average-path-length 1 ; setting the starting point of calculation of avg path length

  reset-ticks
end

to setup-core-nodes
  set-default-shape turtles "star"

  create-core number-of-core-nodes

  ask core [set color white]
```

```

ask core [
; for visual reasons, we want core in center
setxy (random-xcor * 0.01) (random-ycor * 0.01)
set size 1.3
set domain 0
if domain-difference-percentage = 25
[
  Let u 0
  while [u < (number-of-core-nodes / 4)]
  [ ask turtle1 u [ set domain 1 ]
    set u u + 1]
]
if domain-difference-percentage = 50
[
  Let v 0
  while [v < (number-of-core-nodes / 2)]
  [ ask turtle1 v [ set domain 1 ]
    set v v + 1]
]

;set information n-values 3 [random 15];; set initial information each turtle has
;set information lput n-values 2 [random -15] information
;set information reduce sentence information ;; eliminate the redundant information
;set information shuffle information
set self-identity 1 - k-parameter * alpha
set safety alpha
;set behavior random-normal (behavior-mean + 2 * behavior-sd) behavior-sd
; set behavior behavior + 0.5 * behavior-sd
]
end

```

```

to setup-periphery-nodes
  set-default-shape turtles "circle"
  create-periphery number-of-periphery-nodes
  ask periphery [set color white]
  ask periphery
  [
    ; for visual reasons, we don't put any nodes *too* close to the edges
    setxy (random-xcor * 0.95) (random-ycor * 0.95)
    set domain 0
    if domain-difference-percentage = 25
    [
      Let u 0
      while [u < (number-of-periphery-nodes / 4)]
      [ ask turtle2 (number-of-core-nodes + u) [ set domain 1 ]
        set u u + 1]
    ]
    if domain-difference-percentage = 50
    [
      Let v 0
      while [v < (number-of-periphery-nodes / 2)]
      [ ask turtle2 (number-of-core-nodes + v) [ set domain 1 ]
        set v v + 1]
    ]

    ;set information n-values 2 [random 15];; set initial information each turtle has
    ;set information lput n-values 3 [random -15] information
    ;set information reduce sentence information ;; eliminate the redundant information
    ;set information shuffle information
    ;set self-identity 1 - k-parameter * alpha
    set safety alpha
  ]

```

```

;set behavior random-normal (behavior-mean + 2 * behavior-sd) behavior-sd

;set behavior behavior + 0.5 * behavior-sd

]
end

to setup-core-periphery-network

  ask turtles with [domain = 1] [set color red]

  let num-links1 (p-within-core * number-of-core-nodes * (number-of-core-nodes - 1)) ;number of links
  among Core

  let num-links2 (p-within-periphery * number-of-periphery-nodes * (number-of-periphery-nodes - 1))
  ;number of links among periphery

  let num-links3 (p-between-core-periphery * 2 * number-of-core-nodes * number-of-periphery-nodes )
  ;number of links between core & periphery

  ; if average-node-degree > min [number-of-sub1 - 1 number-of-sub2 - 1]

  ; [set average-node-degree min [number-of-nodes - 1 number-of-sub2 - 1]

  ; stop]

  while [count links < num-links1 ]
  [
    ask one-of core
    [
      let choice (one-of other core with [not in-link-neighbor? myself ] ; "myself" here refers to "one-of other
      core"

      )

      if choice != nobody [ create-link-to choice ] ;; "choice" is "one-of other core"
    ]
  ]

  while [ (num-links1 <= count links) and (count links < num-links1 + num-links2) ]
  [
    ask one-of periphery
    [

```

```

    let choice (one-of other periphery with [not in-link-neighbor? myself ]
    )
    if choice != nobody [ create-link-to choice ]
  ]
]
while [(count links >= num-links1) and (count links < num-links1 + num-links3)]
[
  ask one-of turtles
  [
    ifelse breed = core

    [ let choice (one-of other turtles with [(breed = periphery) and (not in-link-neighbor? myself ) ])
      if choice != nobody [ create-link-to choice ]

    ]
    [
      let choice (one-of other turtles with [(breed = core) and (not in-link-neighbor? myself ) ])
      if choice != nobody [ create-link-to choice ]
    ]
  ]
]
if p-within-core != 0 ; this is to make sure every core turtle is connected to at least one other core turtle
[
  ask core with [count out-link-neighbors = 0] [create-link-to one-of other core]
]

if p-between-core-periphery != 0 ; this is to make sure every periphery turtle is connected to at least one
core turtle
[
  ask periphery with [count out-link-neighbors = 0] [create-link-to one-of other core]
]

```

```

]
repeat 2 ; lay out core
[
  layout-spring core links 0.6 0.5 1 ; turtle-set link-set spring-constant spring-length repulsion-
constant
]
repeat 100 ; lay out periphery
[
  layout-spring periphery links 0.6 5 50 ; turtle-set link-set spring-constant spring-length repulsion-
constant
]
ask links
[
  set information2 n-values 2 [random 0]
  set information2 remove 0 information2
  set information3 n-values 2 [random 0]
  set information3 remove 0 information3
  set old 0
] ;; set information2 and information3 as empty list
end

to find-path-lengths ; creating a list for each turtle that tells the distance of all the rest of the turtles from
itself
ask turtles
[ set distance-from-other-turtles []
]
let i 0
let j 0
let k 0
let node1 one-of turtles
let node2 one-of turtles
let node-count count turtles

```

```

;; initialize the distance lists
while [i < node-count]
[
  set j 0
  while [j < node-count]
  [
    set node1 turtle i
    set node2 turtle j
    ifelse i = j ;; zero from a node to itself
    [
      ask node1
      [ set distance-from-other-turtles lput 0 distance-from-other-turtles]
    ]
    [ ;; 1 from a node to it's neighbor
      ifelse [ in-link-neighbor? node1 or out-link-neighbor? node1] of node2
      [
        ask node1 [set distance-from-other-turtles lput 1 distance-from-other-turtles]
      ]
      [
        ask node1 [set distance-from-other-turtles lput infinity distance-from-other-turtles]
      ]
    ]
    set j j + 1
  ]
  set i i + 1
]
set i 0
set j 0
let dummy 0
while [k < node-count]

```



```

[
  set i 0
  while [i < node-count]
  [
    set j 0
    while [j < node-count]
    [
      ;; alternate path length through kth node
      set dummy ( (item k [distance-from-other-turtles] of turtle i) +
                  (item j [distance-from-other-turtles] of turtle k))
      ;; is the alternate path shorter?
      if dummy < (item j [distance-from-other-turtles] of turtle i)
      [
        ask turtle i [
          set distance-from-other-turtles replace-item j distance-from-other-turtles dummy ;; replace distance
if shorter
        ]
      ]
      set j j + 1
    ]
    set i i + 1
  ]
  set k k + 1
]
end

to calculate-utility ;; this calculate a list for each turtle, containing utility for connecting to each other turtle
ask turtles [ set utility []]
let m 0
while [m < count turtles]
[

```

```

ask turtle m
[
  let n 0

  while [n < length distance-from-other-turtles] ;Reports the number of items in the given list, or the
number of characters in the given string.
  [
    ifelse item n distance-from-other-turtles = 0 ;; distance to another turtle is 0 or infinity
    [ let p -999999
      set utility lput p utility
    ]
    [ ifelse item n distance-from-other-turtles > 10000
      [
        let ec [nw:eigenvector-centrality] of turtle n
        ifelse [ec] of turtle n = false
        [
          let p 0 * (1 - safety) + safety * (abs ([domain] of turtle m - [domain] of turtle n)) + random-float
0.00000001

          set utility lput p utility ; utility connecting to itself or turtle it has no access to = minus infinity
        ]
        [
          let p [nw:eigenvector-centrality] of turtle n * (1 - safety) + safety * (abs ([domain] of turtle m -
[domain] of turtle n)) + random-float 0.00000001

          set utility lput p utility ; utility connecting to itself or turtle it has no access to = minus infinity
        ]
      ]
    ]
    [
      let ec [nw:eigenvector-centrality] of turtle n
      ifelse [ec] of turtle n = false
      [
        let p ((1 - safety) * tau * 0 + (item n distance-from-other-turtles - 1) * (1 - safety) * (1 - tau )) +
safety * (abs ([domain] of turtle m - [domain] of turtle n)) + random-float 0.00000001 ;; utility function,
consisted of reducing maximal path length and homophily

```

```

        set utility lput p utility
    ]
    [
        let p ((1 - safety) * tau * [nw:eigenvector-centrality] of turtle n + (item n distance-from-other-turtles
- 1) * (1 - safety) * (1 - tau )) + safety * (abs ([domain] of turtle m - [domain] of turtle n)) + random-float
0.00000001 ;; utility function, consisted of reducing maximal path length and homophily
        set utility lput p utility
    ]
]
set n n + 1
]
]
set m m + 1
]
end

```

```

to go
;calculate-exposure
;ask turtles [set behavior0 behavior]
find-path-lengths
calculate-utility
;information-seeking ; coming from paper eq 7 ; social influence model eq
;;change-ties
calculate-apl; calculation of average path length for plotting
;find-neighborhood
;calculate-cc
;calculate-global-clustering-coefficient

change-path-length
ask turtles [set indegree count in-link-neighbors] ; to plot indegree of turtles

```

```

        ;set deviance-from-mean abs (behavior - mean [behavior] of turtles)
    ;]
;calculate-correlation
;calculate-sum-score
;influence
;ask turtles
;[
    ;ifelse show-information?
    ;[set label length information]
    ;[set label ""]
;]
; if min [length information] of turtles = max [length information] of turtles [    stop ]
if ticks = 50 [
    ; nw:set-context turtles links
    ;ask turtles [ set ec nw:eigenvector-centrality]
    ;calculate-skewness
    ; stop
    ; nw:set-context turtles links
    ; nw:save-matrix "xx.csv"
    ;export-network
    stop
]
; if ticks > 50 [ask turtles [set safety 0.9]]
tick
end

```

to calculate-apl

let num-connected-pairs sum [length remove infinity (remove 0 distance-from-other-turtles)] of turtles;  
removes "zero" then "infinity" from "distance-from-other-turtle" list. then "length" gives the number of  
items in the list

```

;; In a connected network on N nodes, we should have N(N-1) measurements of distances between pairs,
;; and none of those distances should be infinity.

;; If there were any "infinity" length paths between nodes, then the network is disconnected.

;; In that case, calculating the average-path-length doesn't really make sense.

ifelse ( num-connected-pairs != (count turtles * (count turtles - 1) ) ) ;
[ let connected? true
  set average-path-length infinity
  ;; report that the network is not connected
  set connected? false
]
[
  set average-path-length (sum [sum distance-from-other-turtles] of turtles) / (num-connected-pairs) ; first
  sum all the link distances of a turtle then multiply with the number of turtles. then divide by number of
  connected pairs.
]
]
End

to change-path-length
ask turtles [set out-degree count out-link-neighbors]
ask links [die]
let m 0
while [ m < count turtles ]
[
  ask turtle m
  [
    ; let n 0
    ; let o count out-link-neighbors
    ; while [n < count out-link-neighbors]
    ;[ let x item n sort out-link-neighbors
    ;let xx random-float 1
    ;if ((0.8 ^ ([old] of link m [who] of x)) < xx ) ; no new information provided in 3 ticks
    ;[

```

```

; ask link m [who] of x [die]
; ]
;set n n + 1
; ]

let p 0 ;; number of ties haven't been deleted( which means these ties provide new information in past 3
ticks)

let q 0
while [p < out-degree]

[if item q (sort-by > utility) > -999999 and (not out-link-neighbor? turtle (position (item q (sort-by > utility))
utility)) ; turtle with max utility and the one to which the ego-turtle is not connected to

[create-link-to turtle (position (item q (sort-by > utility)) utility)] ;;position (item q (sort-by > utility))
utility is to find number of turtle with the max utility to the ego

;ask links with [information2 = 0]

;[set information2 []

;set information3 []

;set old 0]

set p count out-link-neighbors ;; remain number of neighbors as constant

set q q + 1 ]

]

set m m + 1]

if ticks mod 3 = 0

[repeat 100 [

layout-spring turtles links 0.6 3 10 ;0.6 3 30 ; 0.6 0.5 1 ; 0.6 5 25 ; spring-constant spring-length
repulsion-constant

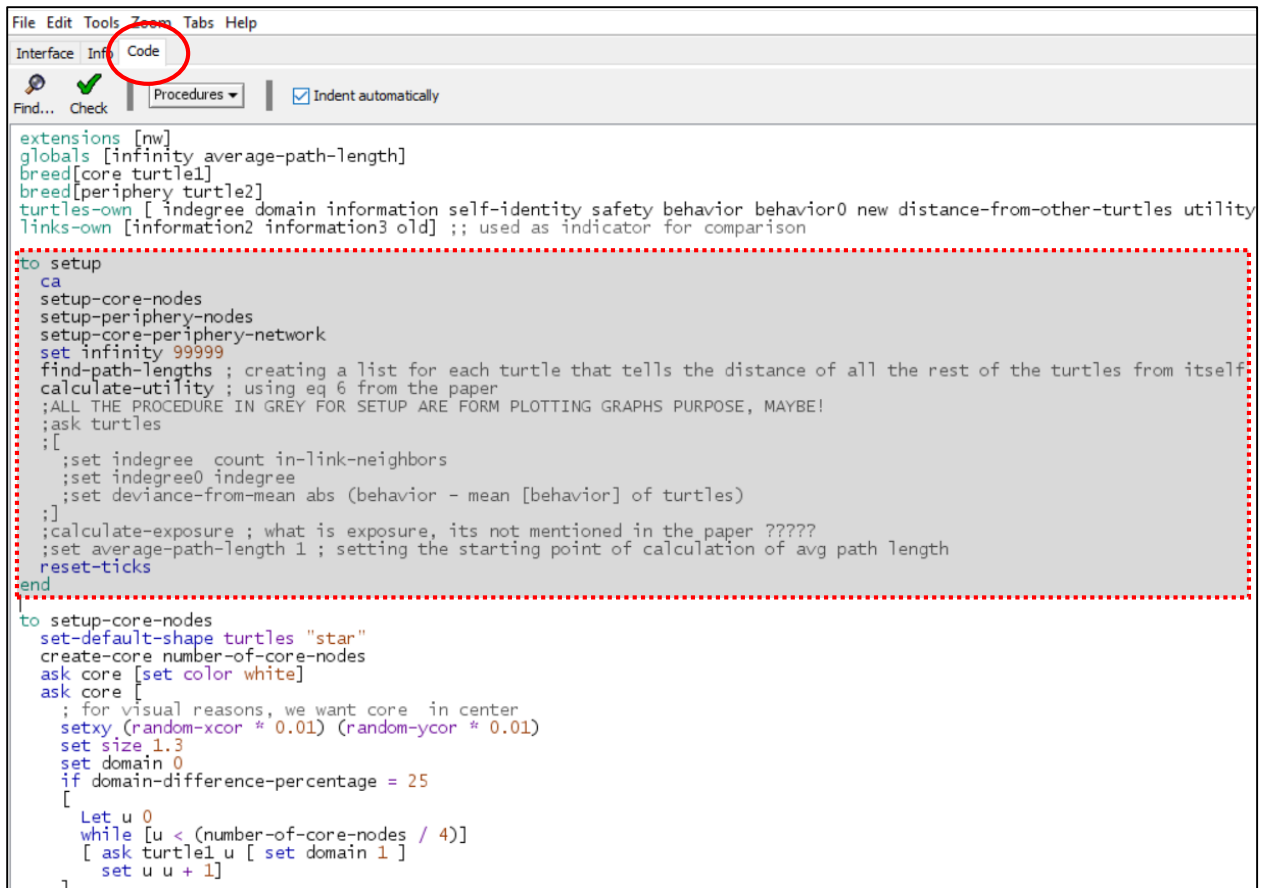
;layout-spring turtles links 0.3 (world-width / (sqrt number-of-nodes)) 1

display]]

end

```

## APPENDIX B: Netlogo Interface

A screenshot of the NetLogo software interface. The 'Code' tab is selected and highlighted with a red circle. The interface includes a menu bar (File, Edit, Tools, Zoom, Tabs, Help), a toolbar with icons for Find, Check, and Procedures, and a checkbox for 'Indent automatically'. The main code area contains several lines of code. A section of code, starting with 'to setup' and ending with 'end', is highlighted in grey and enclosed in a red dotted rectangular boundary. This section includes commands for setting up the simulation environment, such as 'setup-core-nodes', 'setup-periphery-nodes', and 'setup-core-periphery-network'. Below this, there is another procedure 'to setup-core-nodes' which sets the default shape of turtles to 'star' and creates core nodes.

```
File Edit Tools Zoom Tabs Help
Interface Info Code
Find... Check Procedures Indent automatically

extensions [nw]
globals [infinity average-path-length]
breed [core turtle1]
breed [periphery turtle2]
turtles-own [ indegree domain information self-identity safety behavior behavior0 new distance-from-other-turtles utility]
links-own [information2 information3 old] ;; used as indicator for comparison

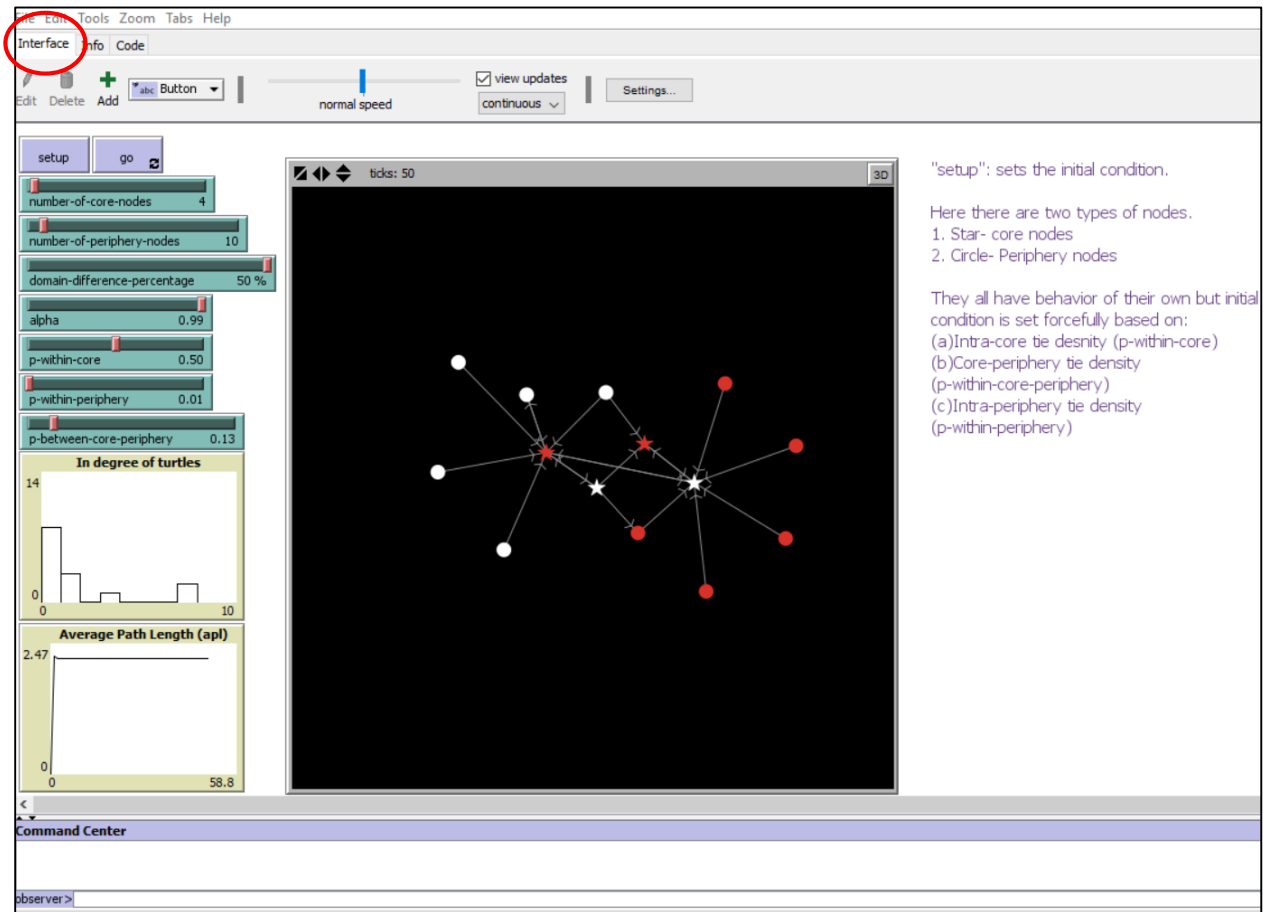
to setup
  ca
  setup-core-nodes
  setup-periphery-nodes
  setup-core-periphery-network
  set infinity 99999
  find-path-lengths ; creating a list for each turtle that tells the distance of all the rest of the turtles from itself
  calculate-utility ; using eq 6 from the paper
  ;ALL THE PROCEDURE IN GREY FOR SETUP ARE FORM PLOTTING GRAPHS PURPOSE, MAYBE!
  ask turtles
  ;[
    ;set indegree count in-link-neighbors
    ;set indegree0 indegree
    ;set deviance-from-mean abs (behavior - mean [behavior] of turtles)
  ;]
  calculate-exposure ; what is exposure, its not mentioned in the paper ?????
  set average-path-length 1 ; setting the starting point of calculation of avg path length
  reset-ticks
end

to setup-core-nodes
  set-default-shape turtles "star"
  create-core number-of-core-nodes
  ask core [set color white]
  ask core [
    ; for visual reasons, we want core in center
    setxy (random-xxcor * 0.01) (random-yycor * 0.01)
    set size 1.3
    set domain 0
    if domain-difference-percentage = 25
    [
      let u 0
      while [u < (number-of-core-nodes / 4)]
      [ ask turtle1 u [ set domain 1 ]
        set u u + 1 ]
    ]
  ]
end
```

**Figure B.1** NetLogo Screenshot Illustrating Code of for A Simulation Model

In the Figure B.1, the highlighted part in grey with dotted boundary is a part of the code known as a “Procedure.” The set of rules defined under such procedures acts as a brain and guides each agent to make learned actions at any time in a running simulation. All such actions are updated and displayed in real time in the “Interface” tab of the NetLogo simulation model. Moreover, the interface tab of the software in itself is a totally customizable area, with all the required buttons such as slider bars, switches, plots, text area, and simulation world screen, etc. The researcher can either use this customizable interface, to observe a part of the simulation result, or to give

instructions, or change attribute values of the agent interacting in the simulation. Figure B.2 presents a screenshot of the interface tab of a NetLogo simulation model.



**Figure B.2** NetLogo Screenshot Illustrating Interface Tab of a Simulation Model

Figure B.2 shows in purple, the “setup” and “go” buttons to set initial condition for the simulation, and run the simulation respectively. Slider bars in green such as “alpha” and “number-of-core-nodes” can be used to control the agents before running a simulation. It also shows histograms and plot charts in the bottom right corner. These updates the viewer in real time about what is happening in the simulation world as the it is running. The black view screen in the center, also known as the “world” within Netlogo, is as the name suggests the simulation world within which agents interact. It also displays the actual movement of the agents while they interact. On the right



side the text in purple is a text display option the NetLogo interface tab provides to the user. This could be used to give a reader a brief insight about the model or specify legends for a viewer to understand what is happening in the interface tab.

As explained earlier agents (nodes/ individuals) in an ABM interact within a model based on a set of rules defined by the coder. If the model represents a phenomenon that evolve overtime, the agents' decision-making process are set such that they interact over and over again following a set rule. In NetLogo program each cycle of this loop is called a "Tick". As the simulation model runs the tick numbers keep on increasing with an increment of 1. This is also displayed in real time just above the simulation world black colored screen on the right corner. In Figure B.2 the tick is displayed as "ticks: 50". This means that when the screenshot of the interface tab was taken the model was in its 50<sup>th</sup> loop of the simulation run.

Just like most of the computing software out there, NetLogo also provides extensions within its coding interface so that specific areas of studies can make a more thorough use of it. One such extension beneficial for simulating network structures on NetLogo is called "[nw]," the network extension. This is also used in the current research agent based NetLogo simulation model.

## REFERENCES

## REFERENCES

- Ahuja, G. (2000). "Collaboration Networks, Structural Holes, and Innovation: A Longitudinal Study". *Administrative Science Quarterly*, 45(3), 425.
- Barabási, A. L. (2016). "Network science: graph theory." *Cambridge university press*, 27.
- Borgatti, S. P., and Everett, M. G. (2000). "Models of core/periphery structures." *Social Networks*, 21(4), 375–395.
- Broekel, T., and Hartog, M. (2013). "Explaining the Structure of Inter-Organizational Networks using Exponential Random Graph Models." *Industry & Innovation*, 20(3), 277–295.
- Burton, R. M., & Obel, B. (2011). "Computational Modeling for What-Is, What-Might-Be, and What-Should-Be Studies—And Triangulation." *Organization Science*, 22(5), 1195–1202.
- Castiglione, F. (2006). "Agent based modeling." *Scholarpedia*, 1(10), 1562.
- Chinowsky, P., Diekmann, J., and Galotti, V. (2008). "Social Network Model of Construction." *Journal of Construction Engineering and Management*, 134(10), 804–812.
- Chinowsky, P. S., Diekmann, J., and O'Brien, J. (2010). "Project Organizations as Social Networks." *Journal of Construction Engineering and Management*, 136(4), 452–458.
- Cohen, W., and Levinthal, D. (1990). "Absorptive Capacity: A New Perspective on Learning and Innovation". *Administrative Science Quarterly*, 35(1), 128.
- Csermely, P., London, A., Wu, L., and Uzzi, B. (2013). "Structure and dynamics of core/periphery networks." *Journal of Complex Networks*, 1(2), 93–123.
- Di Marco, M. K., Taylor, J. E., and Alin, P. (2010). "Emergence and Role of Cultural Boundary Spanners in Global Engineering Project Networks." *Journal of Management in Engineering*, 26(3), 123–132.
- Ding, Z.-kun, Wang, Y.-fei, and Wu, J.-chuang. (2016). "CAS and ABM-based Demolition Waste Management Research in the AEC Industry." *Frontiers of Engineering Management*, 3(1), 18.
- Djomba, J. K., and Zaletel-Kragelj, L. (2016). "A methodological approach to the analysis of egocentric social networks in public health research: a practical example." *Slovenian Journal of Public Health*, 55(4), 256–263.
- Frank, K. A. (1995). "Identifying cohesive subgroups." *Social Networks*, 17(1), 27–56.

- Frank, K. A. (2020). "Ken Frank site - Social Network Resources." *sites.google.com*, <<https://sites.google.com/msu.edu/kenfrank/social-network-resources>> (Mar. 12, 2021).
- Frank, K. A., and Fahrbach, K. (1999). "Organization Culture as a Complex System: Balance and Information in Models of Influence and Selection." *Organization Science*, 10(3), 253–277.
- Frank, K.A., and Xu, R. "Specification, Estimation, and Dynamic Implications of Network Influence." *University of Chicago Center for Spatial Data Science*. November 2020.
- Freeman, L. C. (1977). "A Set of Measures of Centrality Based on Betweenness." *Sociometry*, 40(1), 35–41.
- Frias-Martinez, E., Williamson, G., and Frias-Martinez, V. (2011). "An Agent-Based Model of Epidemic Spread Using Human Mobility and Social Network Information." *2011 IEEE Third International Conference on Privacy, Security, Risk and Trust and 2011 IEEE Third International Conference on Social Computing*.
- Garcia, A.J., Frank, K.A., and Mollaoglu, S.(2020a) (in review). "Multidimensional network structures for knowledge transfer in AEC project teams: From core-periphery to cohesive subgroups via triangles.", *Journal of Management in Engineering*
- Garcia, A. J., Duva, M., Mollaoglu, S., Zhao, D., Frank, K. A., & Benitez, J. (2020b). "Expertise Flows and Network Structures in AEC Project Teams." *Construction Research Congress 2020: Project Management and Controls, Materials, and Contracts* (pp. 95-104). Reston, VA: *American Society of Civil Engineers*.
- Golbeck, J. (2013). "Network Basics." *INST633 - Social Media Analysis*, <<http://www.cs.umd.edu/~golbeck/INST633o/NetworkBasics.shtml>> (Sep 22, 2020)
- Hamill, L. and Gilbert, N. (2009). "Social Circles: A Simple Structure for Agent-Based Social Network Models." *Journal of Artificial Societies and Social Simulation*, 12(2), 3.
- Hansen, M. (1999). "The Search-Transfer Problem: The Role of Weak Ties in Sharing Knowledge across Organization Subunits". *Administrative Science Quarterly*, 44(1), 82.
- Hansen, M. T. (2002). "Knowledge networks: Explaining effective knowledge sharing in multiunit companies." *Organization science*, 13(3), 232-248.
- Hexmoor, H. (2015). "Computational network science: an algorithmic approach." *Morgan Kaufmann Publisher*, Waltham, 1–14.
- Javernick-Will, A. (2012). "Motivating Knowledge Sharing in Engineering and Construction Organizations: Power of Social Motivations." *Journal of Management in Engineering*, 28(2), 193–202.

Kadushin, C. (2012). *Understanding Social Networks: Theories, Concepts, and Findings*. Oxford University Press, 198 Madison Avenue, New York, NY 10016.

Kereri, J., and Harper, C. (2019). "Social Networks and Construction Teams: Literature Review." *Journal of Construction Engineering and Management*, 145(4), 03119001.

Knöri, C., Claudia, B., Althaus, H.J., and Leyk, S.(2007). "Agent based modeling (ABM) for analyzing demand for recycled mineral construction material." M.S. Unpublished Thesis, Swiss Federal Institute of Technology, Zurich.

Lahouti, A. (2013). "Cue-Based Decision-Making in Construction: An Agent-Based Modeling Approach." M.S. Thesis, Michigan State University, East Lansing, MI.

Langton, C. G. (1995). "Artificial life : an overview." *Cambridge, Mass.:* MIT Press.

Laurent, J., and Leicht, R. M. (2019). "Practices for Designing Cross-Functional Teams for Integrated Project Delivery." *Journal of Construction Engineering and Management*, 145(3), 05019001.

Lazarsfeld, P. F., & Merton, R. K. (1954). "Friendship as a social process: A substantive and methodological analysis." *Freedom and control in modern society*, 18(1), 18-66.

Lee, C., Chong, H., Liao, P., and Wang, X. (2018). "Critical Review of Social Network Analysis Applications in Complex Project Management." *Journal of Management in Engineering*, 34(2), 04017061.

Lee, S. K., Kim, H., and Piercy, C. W. (2016). "The Role of Status Differentials and Homophily in the Formation of Social Support Networks of a Voluntary Organization." *Communication Research*, 46(2), 208–235.

Lin, Y. (2014). "Dynamics in organizational problem solving and the leveraging of social capital: An agent-based modeling (ABM) perspective." PhD thesis, University of Washington, Seattle, Washington.

Miller, K. D., & Lin, S.-J. (2010). "Different Truths in Different Worlds." *Organization Science*, 21(1), 97- 114.

Newman, M. E. (2008). "The mathematics of networks." *The new palgrave encyclopedia of economics*, 2(2008), 1-12.

Newman, M. E. J., Watts, D. J., and Strogatz, S. H. (2002). "Random graph models of social networks." *Proceedings of the National Academy of Sciences*, 99(Supplement 1), 2566–2572.

Otte, E., and Rousseau, R. (2002). "Social network analysis: a powerful strategy, also for the information sciences." *Journal of Information Science*, 28(6), 441-453.

Parraguez, P., Eppinger, S., and Maier, A. (2015). "Information Flow Through Stages of Complex Engineering Design Projects: A Dynamic Network Analysis Approach". *IEEE Transactions on Engineering Management*, 62(4), 604-617.

Raoufi, M., and Robinson Fayek, A. (2018). "Fuzzy Agent-Based Modeling of Construction Crew Motivation and Performance." *Journal of Computing in Civil Engineering*, 32(5), 04018035.

Richiardi, M., Leombruni, R., Saam, N. J., and Sonnessa, M. (2006). "A Common Protocol for Agent-Based Social Simulation." *Journal of Artificial Societies and Social Simulation*, 9(1), 15.

Sargent, R. G. (2013). "Verification and validation of simulation models." *Journal of Simulation*, 7(1), 12–24.

Sawhney, A., Walsh, K., & Mulky, A. R. (2003). "Agent-Based Modeling and Simulation in Construction." *Proceedings of the 2003 Winter Simulation Conference*, (pp. 1541-1547), New Orleans.

Sweet, J., and Schneier, M. M. (2017). *Legal Update for Legal Aspects of Architecture, Engineering and the Construction Process*. CI-Engineering.

Szulanski, G. (1996). "Exploring internal stickiness: Impediments to the transfer of best practice within the firm". *Strategic Management Journal*, 17(S2), 27-43.

Tortoriello, M. (2014). "The social underpinnings of absorptive capacity: The moderating effects of structural holes on innovation generation based on external knowledge". *Strategic Management Journal*, 36(4), 586-597.

Wilensky, U. (1999). "NetLogo 5.3.1 User Manual: Programming Guide." *ccl.northwestern.edu*, <<http://ccl.northwestern.edu/netlogo/5.3.1/docs/programming.html>> (Jan. 19, 2021).

Will, M., Groeneveld, J., Frank, K., and Müller, B. (2020). "Combining social network analysis and agent-based modelling to explore dynamics of human interaction: A review." *Socio-Environmental Systems Modelling*, 2, 16325.

William Richard Scott, and Davis, G. F. (2015). *Organizations and organizing : rational, natural, and open systems perspectives*. London ; New York Routledge, Taylor Et Francis Group.

Wolfram, S., and Mallinckrodt, A. J. (1995). "Cellular Automata and Complexity." *Computers in Physics*, 9(1), 55.

Yin, R. K. (2018). *Case study research and applications: design and methods*. Sixth Edition, Sage Publications, Inc, Thousand Oaks, California.

Yuan, Y. C., and Gay, G. (2006). "Homophily of Network Ties and Bonding and Bridging Social Capital in Computer-Mediated Distributed Teams." *Journal of Computer-Mediated Communication*, 11(4), 1062–1084.