



**GENERATING MEANINGFUL INTERACTIONS
BETWEEN NON-PLAYABLE GAME CHARACTERS
FOR ADAPTIVE GAMEPLAY**

MUHTAR AĐKAN ULUDAĐLI

Thesis for the Ph.D. Program in Computer Engineering

Graduate School

Izmir University of Economics

Izmir

2023

**GENERATING MEANINGFUL INTERACTIONS
BETWEEN NON-PLAYABLE GAME CHARACTERS
FOR ADAPTIVE GAMEPLAY**

MUHTAR AĐKAN ULUDAĐLI

THESIS ADVISOR: ASSOC. PROF. DR. KAYA OĐUZ

A Ph.D. Thesis

Submitted to

the Graduate School of Izmir University of Economics

the Department of Computer Engineering

Izmir

2023

ETHICAL DECLARATION

I hereby declare that I am the sole author of this thesis and that I have conducted my work in accordance with academic rules and ethical behaviour at every stage from the planning of the thesis to its defence. I confirm that I have cited all ideas, information and findings that are not specific to my study, as required by the code of ethical behaviour, and that all statements not cited are my own.

Name, Surname:

Muhtar Çağkan ULUDAĞLI

Date:

28.12.2023

Signature:

ABSTRACT

GENERATING MEANINGFUL INTERACTIONS BETWEEN NON-PLAYABLE GAME CHARACTERS FOR ADAPTIVE GAMEPLAY

Uludađlı, Muhtar ađkan

Ph.D. Program in Computer Engineering

Advisor: Assoc. Prof. Dr. Kaya OĐUZ

December, 2023

This thesis presents decision-making methods that are used by non-player characters (NPCs) in computer games; and it proposes a graph generator algorithm to be used by NPC communities. In the thesis, we firstly review the literature for finding out which decision-making methods are used for game NPCs. We list existing methods for industry games, present decision-making frameworks in the literature and also come up with a detailed taxonomy. After reviewing the literature, it can be seen that there are not any appropriate means for generating social networks that can be used by NPC communities in games for adaptive gameplay. Hence, we propose such a generator, AnatoliA, for this purpose. We lay out our key assumptions, present our algorithm and give detailed analysis of our model. Our results show that, AnatoliA outperforms some earlier generators on some of the key metrics. In the final part of the thesis, we also evaluate different use case possibilities of our algorithm and discuss future improvements.

Keywords: Video game, NPC Decision-making, Graph generator, Social networks.

ÖZET

UYARLANABİLİR OYNANIŞ İÇİN OYUNCU OLMAYAN KARAKTERLER ARASINDA ANLAMLI ETKİLEŞİMLER OLUŞTURULMASI

Uludađlı, Muhtar ađkan

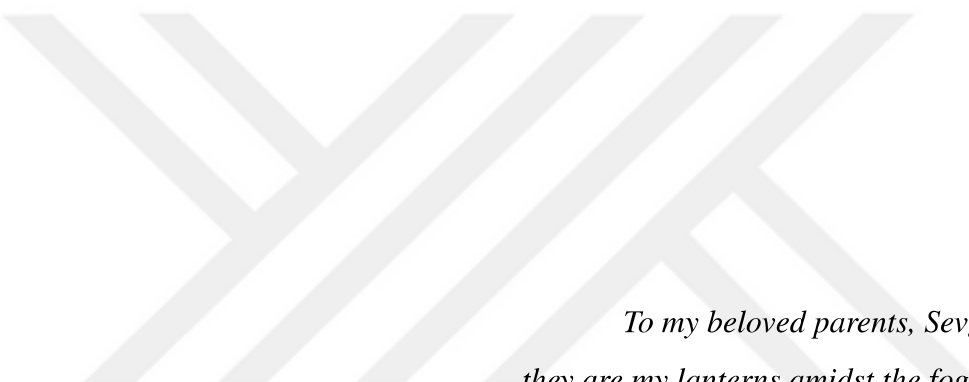
Bilgisayar Mühendisliđi Doktora Programı

Tez Danıřmanı: Do. Dr. Kaya OĐUZ

Aralık, 2023

Bu tez bilgisayar oyunlarında oyuncu olmayan karakterler (OOK'lar) tarafından kullanılan karar verme yöntemlerini sunar, ve bu tür OOK toplulukları tarafından kullanılacak bir izge oluřturucu algoritması önerir. Tezde öncelikle oyunlarda OOK'lar için hangi karar verme yöntemlerinin kullanıldığını bulmak için literatür taraması yapılmıřtır. Bilgisayar oyunları için kullanılan bu yöntemleri tanımlamakta, literatürdeki karar verme yöntemlerini sunmakta ve ayrıca ayrıntılı bir taksonomi oluřturılmaktadır. Literatürü gözden geçirdiğimizde, oyunlarda OOK toplulukları tarafından kullanılacak bir sosyal ađ oluřturmanın uygun bir yolu olmadığını gördük. Biz de bu amaçla AnatoliA adında bir izge oluřturucu yarattık. Tezde, bu algoritma için temel varsayımlarımızı ortaya koyuyor, algoritmamızı sunuyor ve modelimizin ayrıntılı analizini yapıyoruz. Sonuçlarımız, AnatoliA'nın bazı temel ölçümlerde daha önceki bazı izge oluřturuculardan daha iyi performans verdiğini gösteriyor. Tezimizin son bölümü olarak, algoritmamızın farklı kullanım yollarını da deđerlendiriyor ve gelecekteki yapılabilecek iyileřtirmeleri tartıřıyoruz.

Anahtar Kelimeler: Video oyunu, OOK Karar-verme, izge oluřturucu, Sosyal ađlar.



*To my beloved parents, Sevgi and Eşref,
they are my lanterns amidst the fogs of despair..*

ACKNOWLEDGEMENTS

First and foremost, I would like to thank my advisor, Assoc. Prof. Dr. Kaya Oğuz, for his continuous support, mentorship and companionship along the way. Without him, this work would not be completed. And even, without him, my undergraduate thesis project would not be completed; and my field of work, which I am currently satisfied and very happy with, would be different. Throughout these passing years, I have learned so much from him and I will be eternally grateful to him.

Second of all, I would like to thank my precious wife, Damla, for her undying love, support, friendship and partnership throughout this journey. I thank her for bearing all the hardships of life with me, and also for running the house while I was not completely there for her because of this work. Without her, I would be lost already. My sun, my star, the meaning of my life, my joy, my blaze, my paradise, other half of my existence, all my hopes, and my garden of roses, thank you for everything. I love you so much, you are mine, and I will meet you at the end of all time.

Next, I would like to thank my parents, my dear mother, Sevgi and my beloved father, Eşref. They are the reason why I am here and continue to pursue this noble line of work. If they were not always there for me when I need them, I think I would not come this far. I dedicated this thesis to them. I also thank my big brother, Çağrı, and my sister-in-law, Nesrin, for giving me my beautiful niece, Saye Nehir. All my hopes are for the next generation, and I hope that in the future, you will be very successful, prosper, healthy, and most of all, an independent woman. I love you, my little princess.

I would like to thank all of my closest friends. The bad times would be unbearable and the good times would be boring without you. I hope that you will always be around, to laugh, to cry, and to live and to die together.

Lastly, I thank all of my coworkers, my fellow researchers, all the staff of our university, and all of the professors and the teachers whom I have learned so much until now. I am deeply indebted to them for their profound knowledge.

As my last words, I would like to send my eternal love to the heavens for all of my lost ones, especially my grandma and my grandpa, and all the lost ones of all the good people around our world. Life is better because of the dreamers, and thank you very much for teaching me that so early in life. I wish that I will be worthy for you until the end of my life.

TABLE OF CONTENTS

ABSTRACT	iv
ÖZET	v
ACKNOWLEDGEMENTS	vii
TABLE OF CONTENTS	viii
LIST OF TABLES	x
LIST OF FIGURES	xi
CHAPTER 1: INTRODUCTION	1
1.1 Motivation	1
1.2 Preliminaries	2
1.3 Theoretical Foundations	5
1.4 Research Questions	7
1.5 Thesis Layout	8
CHAPTER 2: LITERATURE REVIEW	10
2.1 Literature Review Methodology	10
2.2 Decision-Making Methods of NPCs	11
2.2.1 Finite State Machines	12
2.2.2 Decision Trees	13
2.2.3 Behavior Trees	14
2.2.4 Ruleset- or Logic-based Decision-Making	16
2.2.5 Goal-based Decision-Making	18
2.2.6 Artificial Neural Networks	19
2.2.7 Hybrid Methods	20
2.2.8 Decision-Making Frameworks	20
2.2.9 Review of the Methods	21
2.3 Network Generation Algorithms	21
2.3.1 Network Generation with Communities	23
2.3.2 Network Generation with Communities and Attributes	24
2.4 Summary	26

CHAPTER 3: OUR SOCIAL NETWORK GENERATION ALGORITHM, “AnatoliA”	27
3.1 Preliminaries	27
3.2 Method	28
3.2.1 Assumptions	28
3.2.2 Generation Parameters	29
3.2.3 Algorithm	30
3.3 Results	35
3.3.1 Generated Graphs	37
3.3.2 Assumptions and Model Properties Evaluation	38
3.3.3 Comparison with Other Graph Generators	44
3.3.4 Evaluation with Real Dataset	49
3.4 Discussion	51
CHAPTER 4: APPLICATIONS OF THE PROPOSED ALGORITHM	55
4.1 Further Improvements of Our Method	55
4.2 Previous NPC Networks	57
4.3 A Minor Case Study: AnatoliA vs. Fallout 4 NPC Network	58
4.4 Towards a General NPC Personality Model	62
4.5 The Ultimate Model: Adaptive NPC Community	64
CHAPTER 5: CONCLUSION & FUTURE DIRECTIONS	66
REFERENCES	70
CURRICULUM VITAE	86

LIST OF TABLES

Table 1.	An overview of the literature surveys.	11
Table 2.	The Hybrid Decision-Making Methods.	20
Table 3.	The NPC Decision-Making Frameworks	21
Table 4.	The Decision-Making Methods used in Known Games for NPCs. .	22
Table 5.	The earlier graph generators.	26
Table 6.	Description of generation parameters.	30
Table 7.	Graph model properties and assumptions #1.	42
Table 8.	Graph model properties and assumptions #2.	42
Table 9.	Maximum runtimes of the algorithm.	42
Table 10.	Model properties comparison between graph generators.	45
Table 11.	Similarity evaluation for AnatoliA compared to other generators. .	46
Table 12.	Evaluation results for AnatoliA ground truth communities.	47
Table 13.	Evaluation results for AnatoliA when ground truth communities are not known.	48
Table 14.	Model properties comparison between graph generators using Sinanet benchmark.	50
Table 15.	Community detection comparison between graph generators using Sinanet benchmark.	50
Table 16.	Model properties comparison for AnatoliA using Fallout 4 NPC network.	60
Table 17.	Model properties evaluation with other generators using Fallout 4 NPC network.	61

LIST OF FIGURES

Figure 1.	A sample finite state machine.	12
Figure 2.	A sample decision tree.	14
Figure 3.	Behavior tree details.	15
Figure 4.	An example behavior tree.	15
Figure 5.	A sample schema for rule-based decision-making.	16
Figure 6.	A sample schema for fuzzy-logic based decision-making.	17
Figure 7.	A sample action library and action steps for GOAP.	18
Figure 8.	The overview of AnatoliA algorithm.	31
Figure 9.	Edge generation in AnatoliA.	36
Figure 10.	A graph example with $N = 1250$ nodes and $A = 100$ attributes.	38
Figure 11.	Normal distribution of node-attribute affinity values	39
Figure 12.	AnatoliA vs. LFR for edge degree distributions.	40
Figure 13.	Our Adaptive NPC Community Model.	65

CHAPTER 1 : INTRODUCTION

1.1 *Motivation*

Over the years, computer games have improved substantially, offering players dynamic and vibrant virtual worlds to explore. The actions and interactions of non-player characters (NPCs) are a vital factor in achieving a depth of gameplay experience. The storyline of the game is shaped by NPCs, who take on a variety of roles, including those of allies, enemies, quest givers, and spectators, who add authenticity to the virtual world. Algorithms that enable meaningful interactions between NPCs needs to be developed in order to generate engaging and intelligent NPC behavior.

The absence of sufficient diversity and depth in the area of NPC interaction methods is a major problem. Many current methods place a heavy emphasis on pre-scripted behaviors or basic decision-making models, which can lead to recurring and predictable NPC behavior. To solve this problem, we suggest using each NPC's intrinsic motivation to create an adaptive gameplay experience that increases the diverse and unpredictable nature of NPC activities.

The idea of an NPC-to-NPC (N2N) network, where each node represents a unique NPC and the edges reflect the connections and interactions between NPCs, is crucial to our research. We want to capture the depth and interconnectedness of NPC activities by creating such a network, enabling emergent gameplay and complex social dynamics in the virtual environment.

We carried out a comprehensive review of NPC decision-making techniques that are frequently utilized in academic literature to establish the foundation for our study. This study found a need for innovative methods that go beyond conventional programmed behaviors and add more complex models that take the dynamic nature of gaming settings into account. In order to help NPCs display more complex and contextually relevant behaviors, we attempted to create a social network generation algorithm that is especially suited for NPC decision-making.

The internal attributes particular to each node must also be taken into account while creating the N2N graph, in addition to the interactions between NPCs. These internal characteristics include unique NPC aspects like personality, motives, and ambitions that have a big impact on how they make decisions and interact with other NPCs later

on. We seek to build a more realistic and engaging virtual environment where NPCs have different personalities and interact with one another in sophisticated ways by including these internal attributes into our network generation process.

In summary, this thesis proposes a novel N2N graph generation algorithm for computer games in order to solve the shortcomings of current NPC interaction approaches. We aim to build adaptive gameplay experiences that promote diversity and unpredictability by implementing intrinsic motivation for each NPC. We want to capture the complexity of NPC behaviors through the building of an NPC graph, leveraging internal attributes to support sophisticated decision-making and realistic social interactions. The goal of this study is to improve the field of NPC behavior modeling and assist in the creation of more realistic and compelling video games.

1.2 Preliminaries

In computer science, or more specifically in graph theory, a *graph* can be defined as a discrete structure which includes the nodes, that hold various information in themselves, and the edges, that are the connections between these nodes (Rosen, 2007). In mathematical notion, a simple graph G could be defined as:

$$G = (V, E) \tag{1}$$

where V , is a nonempty set of nodes (can also be called as vertices), and E , is a set of edges. Every edge has two endpoints as nodes, where the edge connects these two nodes together.

There are different graph models. Three most important ones are listed in this section.

The first one is the directed graph. Different from the simple undirected graphs, the edges of a directed graph (which can be called as arcs) have certain directions. The pair of nodes, (u, v) , connected by a directed edge has an order, where u is the starting node and v is the ending node (Rosen, 2007). With this information at hand, directed graph can be denoted as:

$$G_d = (V, E \subseteq \{(x, y) \mid (x, y) \in V^2, x \neq y\}) \tag{2}$$

The second type is the weighted graph. A weighted graph is a graph where each edge is assigned a numerical value (Fletcher et al., 1991). This graph model uses a weight matrix to hold the weight information which can be denoted as:

$$W = [w_{ij}]_{n \times n} \quad (3)$$

where w_{ij} is the weight for the edge between the nodes i and j , and the width and length of the matrix W is n , which is also the number of nodes in the graph. Depending on the context, the weights might reflect different concepts such as costs, lengths, or capacities. With including weight matrix in definition, the weighted graph can be defined as:

$$G_w = (V, E, W) \quad (4)$$

The last type we list here is the attributed graph. Graphs with attributes refer to graph structures where additional information, known as attributes or properties, is associated with the nodes and/or edges of the graph (Newman, 2018). Each node in a graph with attributes can have one or more attributes associated with it. These attributes can reflect a name, a numerical value, a category, a date, or any other important information about the node. Similarly, edges in the graph can be linked with attributes that offer additional information about the relationship between connected nodes.

Essentially, weighted graph is a special kind of edge-attributed graph where it has only one numerical value as the only attribute for the edge. In most cases, we assume the graphs with attributes have only node attributes, but in specific cases or problems, this situation may change. In here, we will define node-attributed graphs. For every node, there may be an attribute list for these graphs. This list can be defined in a matrix form by using the notation:

$$A = [a_{ik}]_{n \times m} \quad (5)$$

where a_{ik} is the k th attribute for node i , n is the number of nodes in the graph and m is the number of attributes per node. By using this matrix, we can denote the graph with node attributes as:

$$G_a = (V, E, A) \quad (6)$$

All of the different graph models we listed above can be combined to create different models. We used undirected unweighted node-attributed graph model for our graph generation algorithm. However, it is easy to change or extend our model with different graph types when needed.

Graphs can be used to denote different networks. Technological networks such as the Internet, power grids or transportation & distribution networks, social and affiliation networks, networks of information such as the World Wide Web and academic citation networks, and natural networks such as biochemical, neural and ecological networks are the few of the many examples that can be represented by computer graphs. From all of these usage scenarios, we use the computer graphs for representing a generated social network in this thesis.

A social network is a network where the nodes are people or occasionally a group of people, and the edges reflect some type of social connection between them, such as friendship or co-working (Newman, 2018). The social networks research is often conducted by sociologists, long before the computer scientists came into the picture. The most studied social network examples by the researchers are the collaborations of scientists (Grossman and Ion, 1995; Newman, 2001), movie actor networks (Amaral et al., 2000), covert or criminal networks (Salganik and Heckathorn, 2004), online communities such as Usenet (Smith, 1999; Lueg and Fisher, 2003), or Facebook (Lewis et al., 2008), and animal networks (Sailer and Gaulin, 1984; Lusseau, 2003).

Most of the time, defining a particular social network is similar to defining a computer graph. The only important thing that must be considered is to select the correct graph type. Is the network is directed or undirected, weighted or unweighted, attributed or non-attributed, or is it used combinations of these, is the real question to answer.

As in these given examples, the researchers mostly studied the networks already created by nature, however, sometimes it is not easy to acquire this kind of information freely. For instance, gathering data from Facebook can be subject to different laws and regulations. In these times, algorithmic generation of synthetic data for social networks is one of the probable solutions.

In computer science and data analysis, the term “graph generation” refers to the act of building or generating graphs. Graph generation can be used for many different

purposes, such as simulating networks, modeling real-world systems, or creating fake data to test apps and algorithms.

There are various methods for generating graphs, and each is appropriate for a particular set of needs and use cases. Some common methods and models for graph generation include random graph generation using models such as Erdős-Rényi model (Erdős and Rényi, 1959) or Watts-Strogatz model (Watts and Strogatz, 1998), preferential attachment model for scale-free networks (Newman, 2018), generation with community structures, and spatial graph model.

In many domains, such as computer networks, data mining, computational biology, and social network analysis, graph generation is a vital tool. Researchers and analysts can create graphs with desired attributes or real-world structures to test and study algorithms, to validate their theories, and to learn more about complex systems and how they behave.

In our graph generation algorithm, we use a model that combines preferential attachment model, community structures, spatial proximity and node-attribute similarity rules together and we create a unique approach to resemble real-world networks in this manner.

1.3 *Theoretical Foundations*

The goal of this section is to lay the theoretical groundwork for research on developing an N2N graph generation algorithm for computer games. We dive into the fundamental ideas of decision-making, NPC definition within the game context, social network generation, and existing methods for constructing such networks in order to allow a thorough comprehension of the proposed research.

NPC behaviour and interaction in computer games centre around decision-making. This process encompasses the selection of actions or behaviours from a range of options available to NPCs, which depend on their internal state and the perceived situation in the game world. The decisions of NPCs are influenced by a range of factors, including their objectives, incentives, convictions, and environmental indications. By implementing decision-making models, NPCs can demonstrate intelligent and adaptable behaviour, thereby enhancing the realism and immersion of the game.

In the context of computer games, NPCs are characters that are under the control of the game's artificial intelligence, rather than the human players. NPCs play different roles and can range from supportive characters who aid the player's progress to adversaries who create challenges and obstacles. These characters add to the story, gameplay, and overall ambience of the game world. Through realistic and captivating behaviour simulations, NPCs increase the player's sense of immersion and control within the virtual world.

Creating a social network within the NPC ecosystem has great potential for promoting complicated and dynamic interactions between NPCs. Social networks represent the associations, links, and dependencies that exist among individuals within a system. Concerning NPC behaviour, a social network can be formed by depicting NPCs as nodes and their interactions as edges, producing a graph-like configuration. This network allows the illustration of social dynamics, influence, and communication patterns among NPCs, which generates contextually appropriate and emerging behaviours.

Several techniques are available to create social networks in the NPC ecosystem. One method is to predefine NPC interactions and relationships based on scripted scenarios or rules. While this technique provides control over NPC behaviour, it frequently causes limited diversity and high predictability. Another method involves using statistical or probabilistic models to create networks based on predetermined attributes or preferences of NPCs. Such models help predict the likelihood of interactions between NPCs, resulting in more behavioural flexibility and variation. Moreover, network creation algorithms may employ machine learning techniques, like reinforcement learning or evolutionary algorithms, to create and modify social networks in response to NPC behaviours and environmental cues.

Our research proposes the development of a new technique for constructing N2N graphs that addresses the limitations of current methods. Our method aims to create an adaptive gameplay environment where NPCs exhibit various unforeseen behaviours by introducing intrinsic motivation for each individual NPC. The algorithm directs each NPC's decision-making process and subsequent interactions within the social network while considering internal characteristics such as personality, motives, and goals that are exclusive to them. Our research aims to advance the field of NPC behaviour

modelling by integrating these underlying concepts and methods, thereby facilitating the development of more realistic and captivating video games.

To conclude, the theoretical foundations of this research encompass the definition of NPCs in games, their employed decision-making techniques, and the development of social networks for these NPCs using existing techniques. By comprehending these essential ideas, a framework can be established for the creation of an N2N graph generation algorithm that encourages contextually rich and adaptable NPC actions in video games.

1.4 Research Questions

This section outlines the research questions that will guide the design of our framework for adaptive and realistic gameplay. Aside from focusing on the specifics of the social network generator and how it is used in NPC communities in games, the questions also encompass the broad objectives of accomplishing human-like interactions and coordination within the framework. The following research questions will be investigated:

- Which NPC decision-making methods are used in game industry? How are these methods used, in which situations are they combined, in which games are they used, which of them are used the most and the least, and how many different game genres are they employed in?
- How can a social network generator that incorporates node attributes and communities be designed to enhance the interactions between NPCs in games?
- How can this generator be compared to other social network generators and real-life networks with respect to performance? What are the key metrics to evaluate such generators and the communities formed by them?
- Are there any known NPC networks with node attributes and communities? Is there a demand for an NPC network generator? Does our network generator create networks similar to in-game NPC networks?
- Is it possible to create a general NPC personality model? Is it possible to create an adaptive NPC community model? Which factors need to be considered to

develop such models?

We want to improve our knowledge of and capabilities for NPC interaction, coordination, and communication in computer games by tackling these research questions. The study will help develop a strong and well-organized framework that supports realistic and adaptable gameplay experiences with an emphasis on human-like interactions. Additionally, research into the social network generation method and how NPC communities use it will provide light on how real-world and social network dynamics are applied to game design and artificial intelligence.

We seek to validate and improve the suggested framework through empirical experiments, simulations, and assessments, assuring its effectiveness and potential for real-world application. The results of this study will make significant contributions to the academic and game development communities by providing novel viewpoints on NPC behavior modeling and increasing the overall gaming experience.

1.5 Thesis Layout

This section details the individual chapters and their corresponding research areas. The thesis structure aims to explore the research topic comprehensively by commencing with an introduction, followed by a literature review, social network generator algorithm, application of the algorithm with NPCs, and ending with a summary of findings and contributions. Below is the suggested layout for the thesis:

1. *Introduction*: The aim of this introductory chapter is to provide an overview of the research focusing on the significance of developing an N2N framework that allows for realistic and adaptive gameplay. It outlines the objectives, research questions and theoretical foundations that serve as the basis for the study. Furthermore, this chapter establishes the context for the following chapters and presents the research methodology adopted.
2. *Literature Review*: Chapter 2 comprehensively reviews previous NPC decision-making methods. The existing academic literature is surveyed in this chapter, examining various techniques and approaches employed in NPC behaviour modelling and decision-making. This chapter identifies limitations and gaps in current methods and highlights the need for novel approaches to enhance NPC

interactions in computer games. In this chapter, the strengths and weaknesses of previous methods are critically analysed, generating a foundation for the development of the proposed framework.

3. *Social Network Generation Algorithm:* Chapter 3 discusses the development and description of the algorithm for generating social networks. This chapter presents the methodology and techniques used to develop a generator that creates social networks with node attributes and communities. The algorithm incorporates assumptions inspired by real-life social network dynamics, such as node placement techniques, the homophily principle, conceptual proximity, and nodes with numeric attributes. The chapter outlines the algorithm's design as well as the implementation details and principles underlying its functionality.
4. *Applications of the Proposed Algorithm:* Chapter 4 examines the implementation of the social network generator algorithm in the context of NPC communities in computer games. This section outlines the potential enhancements that the algorithm can provide and the possibilities of integrating it into gameplay environments. Additionally, it presents early findings and evaluations of the algorithm's efficiency when compared to actual NPC networks. It explores the NPC characteristic model and NPC community model and their potential impacts on future games.
5. *Conclusion:* The thesis concludes with the final chapter, which summarises the significant findings, contributions, and implications of the research. It revisits the research objectives and questions to evaluate the extent to which they have been answered. In this chapter, the significance of the proposed framework for advancing the area of NPC behaviour modelling and its potential impact on the development of more immersive and captivating computer games are discussed. The chapter also outlines possible directions for future research and concludes with closing remarks.

CHAPTER 2 : LITERATURE REVIEW

2.1 *Literature Review Methodology*

NPCs have become an increasingly important part of the overall player experience in the rapidly growing field of computer games. These artificial beings, controlled by their own decision-making processes, add significantly to the story, difficulty and immersion of a game. The need for more sophisticated and dynamic NPC behaviour has become apparent as the gaming industry continues to push the boundaries of realism and engagement. With this in mind, this dissertation begins a thorough investigation of the decision-making processes currently used by NPCs in video games.

We conducted this analysis, because NPCs react and interact based on their own circumstances, and those circumstances require the decision-making techniques to be used to make those actions and interactions happen. By conducting a thorough literature review, we aim to identify gaps for potential innovation, as well as the advantages and disadvantages of current approaches. This review lays the groundwork for the creation of a novel and more immersive decision-making system that promises to enhance the game experience by showcasing the state of the art in NPC decision-making.

It becomes increasingly evident when we examine the intricate details of NPC decision-making in video games that the present situation requires not only more sophisticated decision-making algorithms but also a deeper comprehension of the social dynamics that influence NPC interactions. This necessity has led us to broaden the scope of our investigation beyond just NPC decision-making methods. The development of adaptive NPC communities within the gaming environment has emerged as a crucial element in creating a genuinely immersive and dynamic gameplay experience. In this regard, we identify a notable gap in the existing literature: the absence of a comprehensive social network generation algorithm designed to meet the particular requirements of NPCs and NPC communities. As a result, in an effort to address this shortcoming, our review extends its scope to include the processes involved in the development of social networks with node attributes and communities.

2.2 Decision-Making Methods of NPCs

Decision-making allows NPCs in video games to choose their course of action in relation to both their internal states and how they perceive their surroundings (Millington, 2019). It is possible to develop more convincing NPCs by combining algorithms designed for NPC decision-making, such as behavior trees, in combination with the fundamental algorithms, such as finite state machines or decision trees, which are suited for the majority of games.

A number of surveys on decision-making algorithms contain indirect remarks about how they are used in video games. Some surveys concentrate on a particular algorithm for making decisions, while others include broad artificial intelligence algorithms for video games. The existing surveys with regard to the algorithms addressed are listed in table 1. These studies do not concentrate solely on NPC decision-making algorithms.

Table 1: An overview of the literature surveys. FSM stands for finite state machines, DT stands for decision trees, BT stands for behavior trees, L/R stands for logic-based or rule-based algorithms, GOAP stands for goal oriented action planning, and ANN stands for artificial neural networks.

Title	FSM	DT	BT	L/R	GOAP	ANN
AI in computer games: Survey and perspectives (Cavazza, 2000)	✓			✓		
Current AI in games: A review (Sweetser and Wiles, 2002)	✓	✓		✓		✓
The past, present and future of artificial neural networks in digital games (Charles and McGlinchey, 2004)						✓
The use of Fuzzy Logic for Artificial Intelligence in Games (Pirovano, 2012)				✓		
Behavior Trees for Computer Games (Sekhavat, 2017)	✓		✓			
Building a planner: A survey of planning systems used in commercial video games (Neufeld et al., 2019)					✓	
A Survey: Development and Application of Behavior Trees (Zijie et al., 2021)			✓			
A survey of behavior trees in robotics and ai (Iovino et al., 2022)	✓		✓			
Non-player character decision-making in computer games (Uludağlı and Oğuz, 2023) (Our review article)	✓	✓	✓	✓	✓	✓

In contrast to existing surveys, this review focuses on all available decision-making algorithms with respect to their application for the NPCs in computer games. Some of these are created solely for NPCs and others are decision-making methods also used in other scenarios.

In the first section of this review, we make an attempt to define and categorize the approaches used for NPC decision-making in computer games, and we will also list

the previous studies that include these methods.

2.2.1 Finite State Machines

A finite state machine (FSM) is an abstract machine that has a limited number of fixed states it can be in (Bourg and Seemann, 2004). They were sufficient in the early stages of game industry to give NPCs a way to make decisions. Games like Pac-Man and Sonic used FSMs to control their characters to follow or run away from the player depending on their current state. Due to the elegance of the method and the computational limitations of early computer systems, FSMs were used widely. FSMs are still used in more contemporary games, either alone or in addition to other techniques.

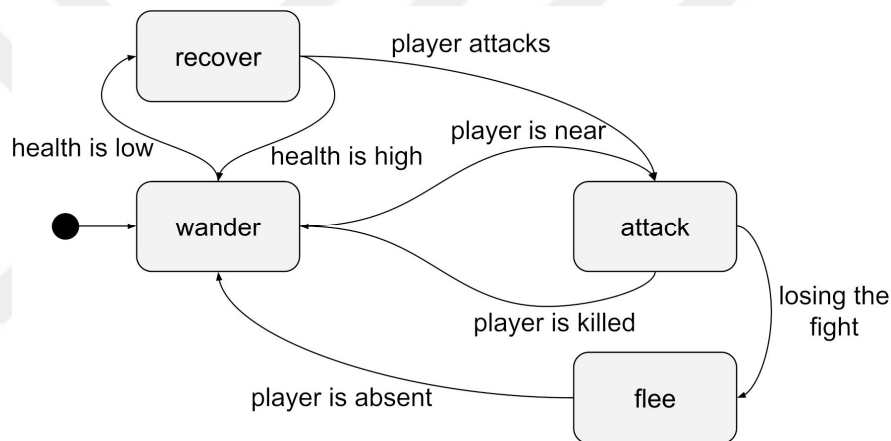


Figure 1: A sample finite state machine for an NPC with four states and seven transitions. In this scenario, when an NPC comes across the player while wandering, it will attack the player. If the player is winning the fight, the NPC runs for its life.

FSMs are a common sight in automata theory. Formally, an FSM can be defined as a 5-tuple $(\Sigma, S, s_0, \delta, F)$ where Σ is the input alphabet, S is a finite set of states, s_0 is the starting state ($s_0 \in S$), δ is the state-transition function as $\delta : S \times \Sigma \rightarrow S$ and F is the finite set of final states ($F \subseteq S$). Their inclusion in video games follows the same rules, such as only being in one state at a time. An FSM contains a set of states and conditions that allow the machine to change states. They don't provide a reject or accept response, unlike their automaton counterparts. Game entities, including NPCs, are given FSMs that allow the entity to behave according to its current state. The machine can change state when a condition is met. The state machine is also destroyed when the entity disappears, such as when an NPC dies. Figure 1 contains a sample

FSM for an imaginary NPC.

FSM usage for NPC decision-making in the literature on is confined (Cavazza, 2000; Nareyek, 2000; Fu and Houlette, 2002). One of the games that implemented FSMs is in the study by Laird (2001), in which used an extension of the FSM concept was used to implement more human-like NPCs. More recent studies have used FSMs for NPCs in various games, including those with a historical theme (Syahputra et al., 2019), in character recognition games (Fathoni et al., 2020), in strategy games with defensive tactics (Fauzi et al., 2019), or 2D character mimicking games (Sindhu et al., 2022).

2.2.2 *Decision Trees*

The algorithmic trees known as decision trees (DTs) can be thought of as a structured set of nested if-then-else rules. Because they are modular and easy to use, they are the simplest way to make decisions.

A DT consists of nodes, each of which can have zero or more child nodes. This is similar to the regular tree abstract data type. Leaf nodes are nodes that don't have any descendants of their own. In a DT, the root node is referred to as the *start node*; nodes with decisions, including the start node, are referred to as *decision nodes*; and the leaf nodes are referred to as the *end nodes* which include actions carried out after they are reached (Millington, 2019).

The basic DT algorithm begins by analyzing decisions at the root. Until an action is performed in a leaf node, each decision moves the algorithm to the next node. Figure 2 shows a sample DT with a total of nine nodes.

DTs are frequently utilized in both industry and academia as the NPC decision-making methods. Mas'udi et al. (2021) employed the NPC decision-making using DTs for a 3D kart-racing game, which is one specific example. Quadir and Khder (2022), in their study, explain the progress made in developing a responsive enemy based on DTs, also employed DTs. For their study, which includes a fighting game, Lie and Istiono (2022) also used DTs.

Black & White (Yannakakis and Togelius, 2018) was another game that used DTs; according to Wexler (2002), who studied this game, the DTs in the game represented the NPCs' beliefs about common object types. To create these DTs, the game uses the



Figure 2: A sample decision tree. If the player is around and healthy, then with a chance of 20 percent, NPC will attack to the player; and with a chance of 80 percent, will evade the player. If the player is not around and the potion is around, then NPC will pick it up.

Iterative Dichotomiser 3 algorithm by Quinlan (1986). This algorithm generates a DT from a dataset by iteratively partitioning features into two or more groups at each step. It applies a top-down greedy approach to construct a DT.

2.2.3 Behavior Trees

A behavior tree (BT) is a tree of hierarchical nodes that regulates an AI agent's decision-making process (Loyall and Bates, 1991; Mateas and Stern, 2002). The actual instructions that govern the AI entity are contained in the tree's leaves. Other branches of the tree contain various kinds of utility nodes that regulate how the AI navigates the tree to reach at the command sequences appropriate to the circumstance. Figure 3 provides a brief categorization and explanation of the BT nodes and tasks' operational principles. Additionally, Figure 4 provides a straightforward illustration of a BT usage situation.

According to Millington (2019), BTs are a common method of shaping the decisions of NPC characters. One of the first modern games to use BTs for NPC decision-making was Halo 2 (Yannakakis and Togelius, 2018), and many other games have since followed suit. The techniques used to represent in-game NPC behavior using BTs are described in Isla (2005). While Johansson and Dell'Acqua (2012) compared BTs with emotional behavior networks in their study, Simonov et al. (2019) incorporated personality traits of NPCs to decision-making to provide a more

Composite Nodes

- • Sequence => behaves like an AND gate. (if all of the child nodes return success, then it is success.)
- ? • Selector => behaves like an OR gate. (if any one of the child nodes return success, then it is success.)

Decorator Nodes

- Inverter => behaves like a NOT gate.
- Succeder => always return success result while processing this branch.
- Repeater => reprocess its children each time they return a result.
- Repeat until Fail => reprocess its children until they return failure result.

Leaf Nodes => nodes that implemented the actions of AI.

Result Types => Success, Failure, Running

Figure 3: Node, task and result types of behavior trees.

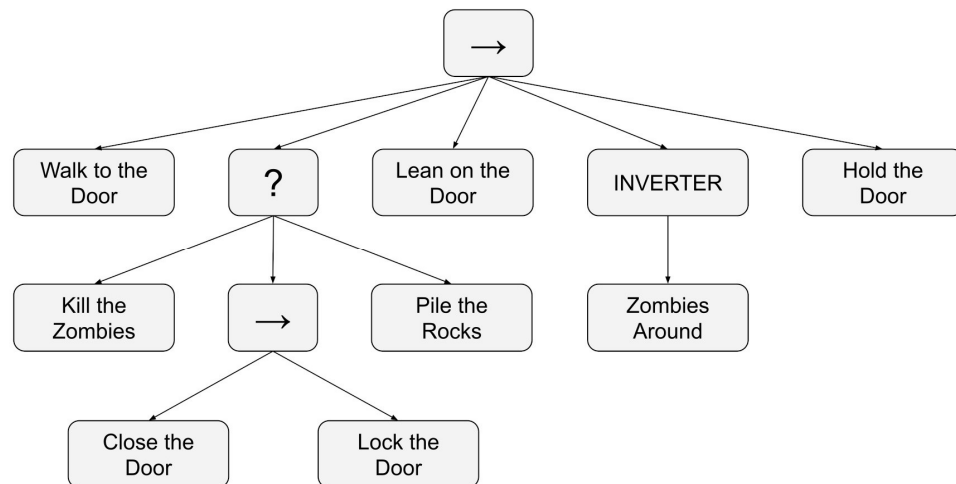


Figure 4: An example behavior tree. If NPC can walk to door, then it must carry out at least one of the actions in the selector branch. If it kills the zombie, then it returns success to the parent selector node. Or if it closed and locks the door, then it returns success to the parent selector node also. If it wants to carry out all of the sequence, then it must lean on the door, and when zombies are not around, it must hold the door.

convincing gaming AI.

The Mafia III video game’s NPC decision-making algorithm was described in Holba and Huber (2021). The issues the creators had when building the AI for The Last of Us were covered in Panwar (2022). Colledanchise and Ögren (2018) explains how they used BTs to build a Pac-Man playing agent from scratch in their book. The use of BTs in AI design for the multiplayer online battle arena (MOBA) gaming genre was mentioned in the Lin et al. (2019).

BTs are also used in various game genres, such as role-playing games (RPGs). One such example is implemented in the research of Rodrigues et al. (2021). BTs can be used in first-person shooter (FPS) games and virtual reality (VR) games. The study by Pyjas et al. (2022) used BTs in a FPS VR game set in the World War II era. In Miyake et al. (2019), the authors explained that they used BTs and FSMs together to create a decision-making tool for NPCs in the game Final Fantasy XV, which they call the *AI Graph Editor*.

2.2.4 Ruleset- or Logic-based Decision-Making

Rule-based systems (RBSs) were used in the early stages of the AI research. According to Millington (2019), using RBSs is not a common strategy because it requires more effort and is less effective than using FSMs or DTs.

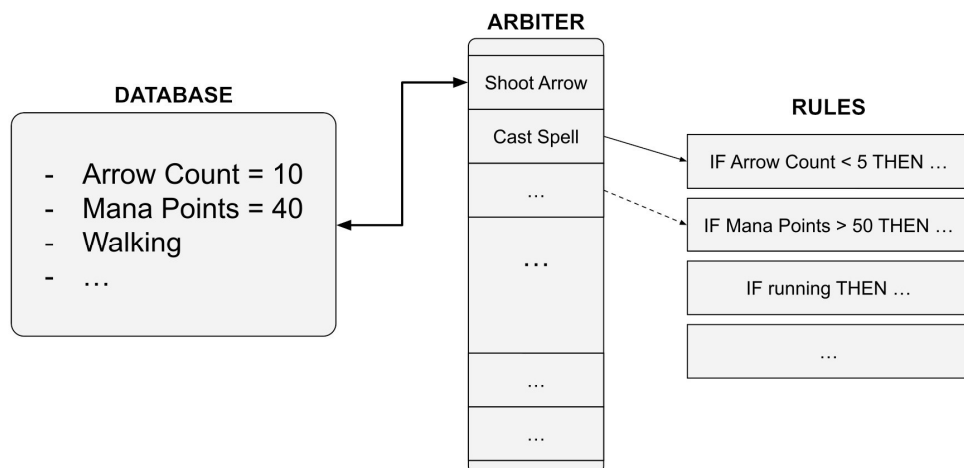


Figure 5: A sample schema for rule-based decision-making. The arbiter is an intermediate process that checks the rules according to the information in the database to complete certain actions. In this example, the first rule will not be fired at this moment to complete *Cast Spell* action, since *Arrow Count* is 10 at the database and the first rule states that it must be less than 5.

The organization structure of RBSs includes three elements: *an information database, a rule set, and between these, an arbiter* as shown in Figure 5.

According to Forgy (1979, 1989), most AI systems use the Rete Matching algorithm to match the appropriate rules with the correct information in the database for RBS. Baldur's Gate and Virtua Fighter (Yue and de Byl, 2006) are two games known to use rule-based decision-making algorithms such as Rete to regulate the actions of NPCs.

The other method of rule-based decision-making is fuzzy logic (FL). FL is a form of logic in which the truth values of the given variables can be real numbers between 0 (false) and 1 (true). These truth values are called *partial truth values*, and their values, such as *half true* or *nearly false*, can be defined with this usage (Novák et al., 1999). The use of FL in game AI was first described in O'Brien (1996) and is widely used in both the academic and industrial game communities. The procedure for using FL is shown in Figure 6.

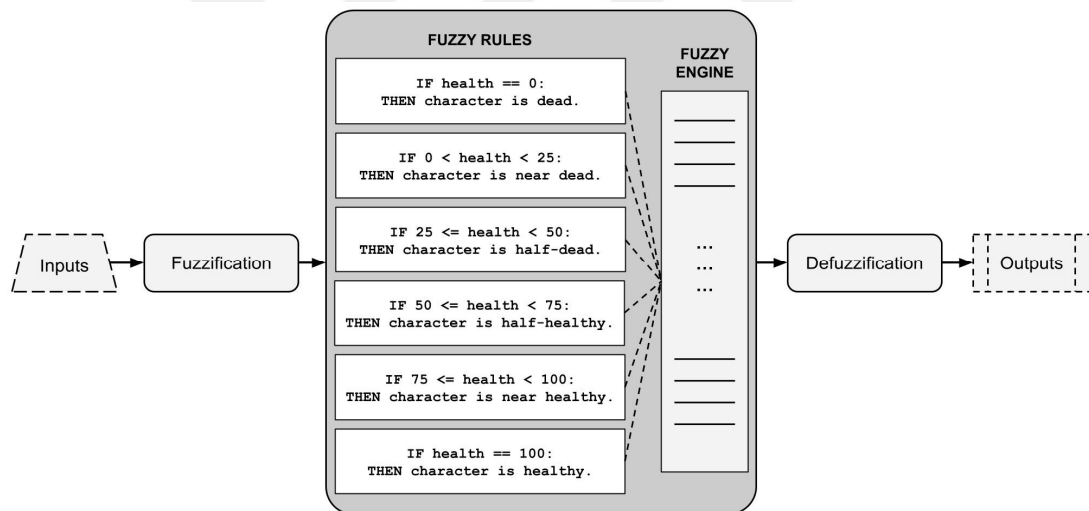


Figure 6: A sample schema for fuzzy-logic based decision-making. The method takes inputs from NPCs or from the environment, fuzzify them, decide what to do according to the fuzzy rules, defuzzify the result that is coming from the fuzzy decision-making engine and gives needed outputs to the NPCs or to the environment.

There are numerous articles about the use of FL in game AI. Shaout et al. (2006) created a Pac-Man clone game in which all enemy decisions are made by FL. Fujii et al. (2008) investigates RBS and FL usage in a car racing game. Van Waveren (2001) also used FL in his thesis to develop the decision-making of NPCs in the game Quake III Arena clone. Li et al. (2004) used belief-desire-intention agent theory for an earlier

arcade game called BattleCity, and used an FL-based framework for enemy NPCs to increase the detail of their decision-making. SoyLucicek et al. (2017) also used FL for drone decisions in a space-themed 2D arcade game called Meteor Escape.

There are more sophisticated applications of FL in NPC decision-making. To control cars in video games, Niewiadomski and Renkas (2014) invented a novel type of fuzzy controller called a hierarchical fuzzy controller. To see if a partner NPC in a poker game could use facial expressions to advise the player character about the opponent's hand, Ohson and Onisawa (2008) built a DT using FL. The researchers integrated the BTs and FL to provide a decision-making approach for the robot agents playing a soccer game simulation in Abiyev et al. (2016).

2.2.5 Goal-based Decision-Making

NPCs in video games may have specific tasks to perform. Goal-oriented behavior occurs when an NPC chooses an option from a list of options that brings the NPC closer to its goal. When multiple goal-oriented actions are performed sequentially to achieve a larger goal, this is the foundation of goal-directed action planning (GOAP) (Millington, 2019; Orkin, 2003). The essential elements and working mechanism of GOAP is described in Figure 7.

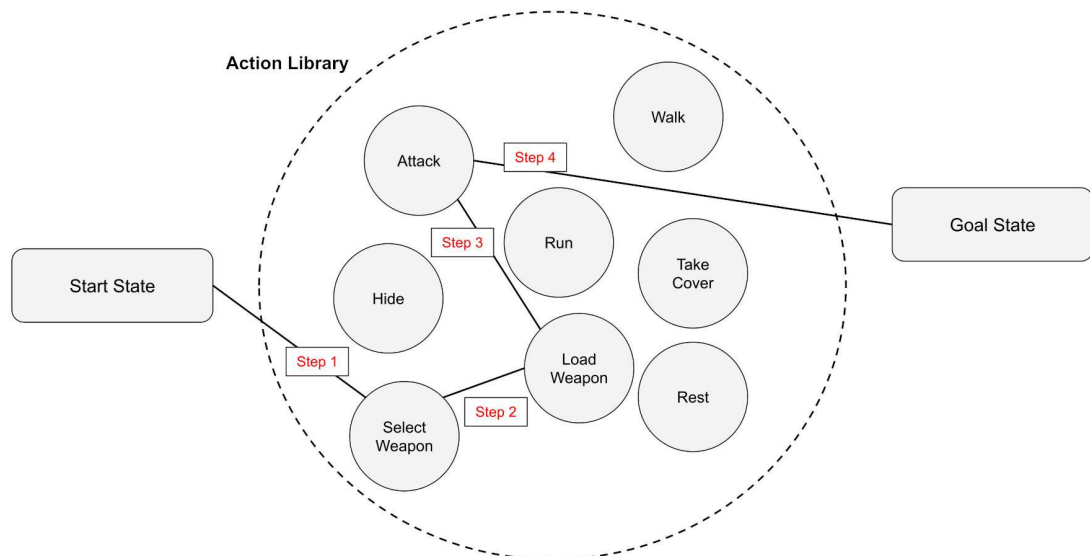


Figure 7: A sample action library and action steps for GOAP. According to this example, NPC selects certain actions with orderly fashion from the library to fulfill its goals. This NPC selects *Select Weapon*, *Load Weapon* and *Attack* action states in this order and reaches its goal state, which we can guess that it may be *Defeat Enemy* state.

Orkin (2006) indicates that the NPC movement in the video game F.E.A.R. was handled by an FSM with only three states. However, they developed a goal-based system that transitions to the appropriate state using the available knowledge for NPC interaction and planning. GOAP was also studied and compared with FSMs by Long (2007). He examined two approaches based on different criteria. To address the shortcomings of GOAP, Sloan et al. (2011) developed a new strategy called Utility-Directed GOAP and compared it to a Sims-like environment. Sloan (2015) expanded on his earlier study by introducing the idea of *smart ambiance*, which allows NPCs to use environmental context knowledge to their advantage.

A real-time tactical space game was created by Studiawan et al. (2018) to test whether GOAP would improve performance in this type of game. Suyikno and Setiawan (2019) studied the use of GOAP, focusing on the hide-and-seek behavior of NPCs in the context of stealth-oriented video games. The study by Johansen et al. (2022) examined the use of domain-independent classical planners for NPC control in an interactive storytelling game engine.

In addition, GOAP can be designed for use in a variety of game engines and development environments. An example is given by Sielicki et al. (2018), who developed GOAP modules in Unreal Engine and observed a severe lack of efficient GOAP tools and frameworks that common developers could use on the most popular game engines. If the system has only one goal, GOAP can also be combined with the Iterative Deepening A* algorithm to solve time complexity problems (Millington, 2019).

2.2.6 Artificial Neural Networks

Artificial neural networks (ANNs) are mathematical structures consisting of several processors connected in various ways. They are artificial, but have some vague similarities to the human neurological system, which influenced their design (Gupta, 2013). ANNs are mostly used in engineering for data operations such as pattern recognition, image processing, and similar tasks because of their ability to analyze large amounts of data.

Artificial neurons (or simply *nodes*) carry information in an ANN, while *edges* connect these nodes by delivering data signals. To determine which neuron is activated,

the weights of all input nodes are determined using a mathematical formula. The output of a neuron is determined from its inputs.

ANNs are typically used as a learning component in machine learning approaches. However, in some situations they can be used as the primary decision-making technique for NPCs in a computer game. Madsen and Adamatti (2013) used ANNs to help NPCs make decisions in a role-playing game. They used ANNs and FSMs to decide the attack behavior of the NPC. Although they did not use a computer game in their study, Jolly et al. (2007) offered an original decision-making strategy for NPCs in a robot football league using ANNs.

2.2.7 Hybrid Methods

Other than these studies, there are also some hybrid decision-making methods that combine different methods together. For instance, Shu and Chaudhari (2008) combines fuzzy Logic with neural networks and Abiyev et al. (2016) combines fuzzy logic with behavior trees. Whole list of hybrid methods can be seen from Table 2.

Table 2: The Hybrid Decision-Making Methods.

Related Study	Method Combination
(El-Nasr et al., 2000)	Decision Tree, Fuzzy Logic
(Orkin, 2006)	Finite State Machine, Goal-Oriented Action Planning
(Ohsone and Onisawa, 2008)	Decision Tree, Fuzzy Logic
(Shu and Chaudhari, 2008)	Artificial Neural Network, Fuzzy Logic
(Florez-Puga et al., 2008)	Behavior Tree, Case Based Reasoning
(Fujii et al., 2008)	Rule-Based System, Fuzzy Logic
(Perez et al., 2011)	Behavior Tree, Grammatical Evolution
(Dey and Child, 2013)	Behavior Tree, Reinforcement Learning
(Madsen and Adamatti, 2013)	Finite State Machine, Artificial Neural Network
(Schwab and Hlavacs, 2015)	Behavior Tree, Genetic Algorithm, Goal-Oriented Action Planning
(Fu et al., 2016)	Behavior Tree, Reinforcement Learning
(Abiyev et al., 2016)	Behavior Tree, Fuzzy Logic
(Sagredo-Olivenza et al., 2017)	Behavior Tree, Programming by Demonstration
(Kopel and Hajas, 2018)	Decision Tree, Artificial Neural Network, Reinforcement Learning
(Partlan et al., 2022)	Behavior Tree, Genetic Algorithm
(Meng and Hyung, 2022)	Behavior Tree, Reinforcement Learning, Simulated Annealing
(Zhu and Feng, 2022)	Artificial Neural Network, Genetic Algorithm
(Widhiyasana et al., 2022)	Artificial Neural Network, Genetic Algorithm

2.2.8 Decision-Making Frameworks

In the field of NPC decision-making, several frameworks have also been established for various goals. Some are used for general decision-making of NPCs, some are

used as personality engines for virtual characters and others, as emotion engines for the appraisal theory. We have listed all of these NPC decision-making frameworks in Table 3.

Table 3: The NPC Decision-Making Frameworks

Framework Type	Framework
General Decision-Making	AI Graph Editor (Miyake et al., 2019), F.E.A.R SDK, GPGOAP, ReGoap (Orkin, 2006)
Personality Engine	Extreme AI (Georgeson and Child, 2016)
Emotion Appraisal Engine	GAMYGDALA (Popescu et al., 2013), FAtiMA (Dias et al., 2014; Mascarenhas et al., 2021), EmoBeT (Belle et al., 2022), EmotionalBT (Dell’Acqua and Costantini, 2022)
Social Interaction	CiF (McCoy et al., 2010, 2011), CiF-CK (Guimaraes et al., 2017)
Communication Network	GVSN (Perrie and Li, 2014)

2.2.9 *Review of the Methods*

Overall, the first part of this survey covers a wide range of studies. We have attempted to collect published video games and the NPC decision-making mechanisms used in such games from the reviewed papers. We provide a bibliography that outlines the decision-making mechanisms used in each game. The list for the NPC decision-making methods used in these video games is shown in Table 4.

Our research suggests that decision-making techniques can be used with NPCs, the environments in which they interact, or with other entities such as communities of NPCs. The decision processes used by NPCs can be useful for spatial movement, planning or acting, controlling vehicles or other devices, or interacting with other game elements such as players or other NPCs. First-person shooter (FPS), strategy, role-playing (RPG), action/adventure, or 2D/platform are the five primary game genres to which the game environments in the study belong.

2.3 *Network Generation Algorithms*

There are extensive surveys covering the topic of graph generation, including (Chakrabarti and Faloutsos, 2006; Bonifati et al., 2020; Xiang et al., 2021). There are also

Table 4: The Decision-Making Methods used in Known Games for NPCs.

Game	Studio/Year	Method	Source
Virtua Fighter	Sega-AM2, 1993	RB	Yue and de Byl (2006)
Close Combat	Atomic Games, 1996	FL	Sweetser and Wiles (2002)
Creatures	Millennium Interactive, 1996	ANN	Yannakakis and Togelius (2018)
Half-Life	Valve, 1998	FSM	Game Source Codes ¹
Baldur's Gate	BioWare, 1998	RB	Yue and de Byl (2006)
Unreal	Epic Games, 1998	FL	Johnson and Wiles (2001)
Police Quest: SWAT 2	Sierra Entertainment, 1998	FL	Sweetser and Wiles (2002); Johnson and Wiles (2001)
Civilization: Call to Power	Activision, 1999	FL	Johnson and Wiles (2001)
The Sims	Maxis, 2000	RB, GOAP	Millington (2019)
Black & White	Lionhead Studios, 2000	DT	Yannakakis and Togelius (2018)
No One Lives Forever 2: A Spy in H.A.R.M.'s Way	Monolith Productions, 2002	FSM, GOAP	Orkin (2003)
Halo 2	Bungie Inc., 2004	BT	Yannakakis and Togelius (2018)
S.W.A.T. 4	Irrational Games, 2005	GOAP	Pittman (2007)
F.E.A.R.	Sierra Entertainment, 2005	GOAP	Orkin (2006)
S.T.A.L.K.E.R.: Shadow of Chernobyl	GSC Game World, 2007	GOAP	Long (2007)
Deus Ex: Human Revolution	Eidos, 2011	GOAP	Sloan (2015)
The Last of Us	Naughty Dog, 2013	DT, BT	Panwar (2022)
Alien: Isolation	SEGA, 2014	BT	Panwar (2022)
Final Fantasy XV	Square Enix, 2016	FSM, BT	Miyake et al. (2019)
Mafia III	2K Games, 2016	DT, BT	Holba and Huber (2021)

various models for generating networks, including *random graphs*, *power-law degree-distributed random graphs*, *small-world model*, and *exponential random graphs*. The works of Aiello et al. (2000, 2001); Nobari et al. (2011) analyze these models. The main focus of this section will be the studies that present graph generation methods with communities and attributed nodes.

2.3.1 Network Generation with Communities

Most of the time, networks with communities are generated to assess the performance of community detection methods. Community detection is a popular research area, as discussed in comprehensive surveys (Lancichinetti and Fortunato, 2009b; Orman and Labatut, 2009; Fortunato, 2010; Fortunato and Hric, 2016; Javed et al., 2018).

The studies of Fortunato (2010) and Fortunato and Hric (2016) use incredibly thorough surveys to explain fundamental ideas like how a community is defined, community structure in actual networks, overlapping communities, and so on. Rossetti and Cazabet (2018) focuses on the dynamic networks that vary over time, while Xie et al. (2013) surveys overlapping community detection in particular. These works can be used to gain more understanding of the community detection topic.

Although many studies have been conducted to discover communities in real networks, approaches for generating synthetic networks with communities are still few in the field. The study by Girvan and Newman (2002) develops one of the first network generators. They developed this generator to evaluate the effectiveness of their recently proposed community discovery method. Some features of real-world networks, such as variable degree distributions on nodes and fixed community sizes, were not present in their generator.

The Lancichinetti-Fortunato-Radicchi (LFR) benchmark, named after its developers, is one of the most well-known network generators for community detection in networks (Lancichinetti et al., 2008). For the degree distributions of the nodes, they generated their network using a power-law. Real-world networks, often known as scale-free networks, have this characteristic. Being a scale-free network means that the few nodes have many edges while the majority of nodes have few edges (de Solla Price, 1965; Barabási and Albert, 1999). To better mimic the properties of real-world networks, they adopted the power-law degree distribution for both the nodes and the

community formation. In addition, directed and weighted graphs with overlapping communities are added to the LFR benchmark (Lancichinetti and Fortunato, 2009a).

Since the networks with communities are created to test the community detection methods, the node attributes are ignored during the generation process.

2.3.2 *Network Generation with Communities and Attributes*

There are studies that generate networks with node attributes without considering communities. DataSynth framework generate property graphs by using their novel graph partitioning algorithm (Prat-Pérez et al., 2017). Attributed graph model utilizes generative graph model (Leskovec et al., 2010) to create node attributes from previously observed networks (Pfeiffer III et al., 2014). Since our approach considers both the generation of networks with node attributes and the generation of communities, this section focuses on the studies that cover both aspects.

The LFR-EA (Elhadi and Agam, 2013) is one of the earliest to generate a network containing both community structure and node characteristics. It is an addition to the LFR benchmark. This technique was developed to find communities using node attributes in networks. Three parameters—total attribute count, domain value count for each attribute, and assignment influence parameter—control the attribute generation on top of LFR network generation.

The idea of homophily arises when node attributes are taken into account in the context of community creation. The easiest way to explain homophily is that it occurs when nodes connect with other nodes that are more similar to them than with nodes that are not (McPherson et al., 2001). As a result, the nodes within a community have a tendency to share more characteristics. The study by Largeron et al. (2015), in which they introduce the Attributed Graph Generator with Community Structure (ANC) algorithm, is one example that produces synthetic networks employing the homophily property. They designed a network generation model with the properties of *local preferential attachment*, *small-world*, *community structure*, *community homogeneity* and *homophily*.

Benyahia et al. (2016) extend ANC for dynamic networks. Dynamic networks are the networks that change and evolve over time. The authors assume that the most of the time the nodes and the edges in a social networks change, new connections are

formed, some nodes join or leave certain communities in the network. Therefore, they created a framework called DANCer, to be able to generate such networks.

In contrast to the previous examples, Nettleton (2016) developed a generator for online social networks. They generated appropriate datasets for predefined graph topologies by varying node attribute values, community structures, and data distributions, but they did not generate the actual network. They used public datasets, such as those from the government and social networks, to create their graphs specifically for their application domain.

acMark, a network generator with cluster labels and node properties, was introduced by Maekawa et al. (2019). The advantages of this method, according to the authors, include variable control over cluster separability, a variety of distributions for attribute values, node degrees, and cluster sizes, and linear time complexity proportional to the number of edges generated. They generated the networks using a Bayesian approach and evaluated the results in terms of graph property distribution management, generation scalability, cluster separability, and unambiguous clustering implementation.

According to Wang et al. (2021), the FastSNG method is the fastest social network generator with attributes and communities. The authors compared their method with previous network generation techniques in terms of time complexity and concluded that it can generate a network with a trillion nodes in a reasonable amount of time.

Citraro and Rossetti (2021) presented X-Mark, a recent network generator that combines structural topology and node attributes to create networks. They compared their approach to previous node-attributed network generators. Their method uses both categorical and continuous attribute value types and produces networks that adhere to both community homogeneity and the homophily principle.

GenCAT (Maekawa et al., 2021), an extension of the previously created acMark generator (Maekawa et al., 2019), is the final generator discussed in this section. GenCAT aims to resolve the problem of inaccurate simulation of relationships among labels (i.e., communities), node attributes, and graph topology that may occur in earlier generators. Through conducting additional tests and evaluations, they expanded their strategy in acMark.

You can overview all the previous generators we survey here with their specified

properties in Table 5.

Table 5: The earlier graph generators. \checkmark indicates that the given generator includes that property, while \times indicates that it is not. *Comm.* means “Communities”, *Attr.* means “Node attributes”, *Param.* means “Adjustable method parameters/properties”.

The Generator	Comm.	Attr.	Param.	Degree Distribution	Properties
G.&N. (Girvan and Newman, 2002)	\checkmark	\times	\times	constant	one of the first generators
LFR (Lancichinetti et al., 2008)	\checkmark	\times	\times	power-law	power-law degree dist.
Overlapp. LFR (Lancichinetti and Fortunato, 2009a)	\checkmark	\times	\times	power-law	overlapping communities
DataSynth (Prat-Pérez et al., 2017)	\times	\checkmark	\checkmark	schema-driven	topology created with synthetic data
AGM (Pfeiffer III et al., 2014)	\times	\checkmark	\checkmark	attribute-correlated	property-to-node matching algorithm
LFR-EA (Elhadi and Agam, 2013)	\checkmark	\checkmark	\checkmark	power-law	LFR with attributes
ANC (Largeron et al., 2015)	\checkmark	\checkmark	\checkmark	clustering method-driven	categorical attributes
DANCer (Benyahia et al., 2016)	\checkmark	\checkmark	\checkmark	clustering method-driven	dynamic graph & communities
acMark (Maekawa et al., 2019)	\checkmark	\checkmark	\checkmark	uniform, normal, power-law	class-preference probability
FastSNG (Wang et al., 2021)	\checkmark	\checkmark	\checkmark	power-law	claimed to be the fastest
X-Mark (Citraro and Rossetti, 2021)	\checkmark	\checkmark	\checkmark	power-law	one of the first attribute-related edge generation
GenCAT (Maekawa et al., 2021)	\checkmark	\checkmark	\checkmark	power-law, normal, input list	class-preference probability
Our method ‘AnatoliA’	\checkmark	\checkmark	\checkmark	currently power-law (adjustable)	modifiable distributions, object-oriented, attribute&proximity-related edge generation

2.4 Summary

Our overarching goal is to use the knowledge gained from these two explorations—the reviews of NPC decision-making methods and social network generation—to design a new social network generator. In contrast to typical methods, this generator establishes a strong emphasis on node attributes, closely replicating the personality traits and characteristics of NPCs in games. We aim to develop an innovative method for establishing NPC networks, enhancing the immersive experience of virtual worlds.

CHAPTER 3 : OUR SOCIAL NETWORK GENERATION ALGORITHM, “ANATOLIA”

3.1 Preliminaries

One important feature of the real networks is that some nodes in the network connect to particular nodes more than they connect to other nodes. By doing so, they create certain groups that can be called communities. A *community* can be defined as a group of nodes that are connected within that group with higher probabilities than being connected to the nodes in outer groups (Fortunato and Hric, 2016). When the definition of the communities is based on the connectivity patterns of the graph, it is often referred as a *structural community*; however, if they are defined according to a common function or a role of the community nodes, it is referred as a *functional community* (Yang and Leskovec, 2015). A more recent and preferable naming convention for functional communities is *meta-community* (Bonifati et al., 2020), and the communities created by our method will be referred as such hereafter.

When real networks are investigated, it can be speculated that the nodes are connected to one another with respect to their similar internal properties. As an example, consider that the people are nodes of a real network, and they form relations if they share similar personality traits, such as thinking of two people can share a bond if they both like cats. The network nodes, in our perspective, also tend to connect to other nodes that are similar to them from the *conceptual distance* aspect. One such example of that can be being in a professional relation with co-workers. Our assumption is that these two factors are the most common features of real world networks when we think about the process of forming relationships. Therefore, they have been considered as the foundational principles of the method in this study.

We propose the *AnatoliA* algorithm which can generate a node-attributed network with meta-communities. These meta-communities are defined by the node attributes, and the edges between the nodes are created with a combination of node attributes and their conceptual node positions. *AnatoliA* is novel in its definition of generalized attributes and the generalized functions to create connections between nodes. Many parameters of the graph generation process can be tweaked and adjusted to meet the requirements of the users.

The following section presents our assumptions and explains our method. Section 3.3 gives our results with respect to the comparisons with similar or alternate methods and a real dataset. The final section concludes Chapter 3 with our final thoughts and the future work that can be done to improve our method.

3.2 *Method*

We explain the design and implementation of our proposed algorithm, AnatoliA in this section.

3.2.1 *Assumptions*

Assumption. 1: *Randomness of node placements* Firstly, our method employs diamond-square algorithm to scatter the nodes pseudo-randomly into the graph. Diamond-square algorithm (Fournier et al., 1982) is a method mostly used for generating heightmaps for 3-dimensional terrains. The algorithm is a tool in computer graphics for creating realistic landscapes.

Most of the graph generators we have surveyed did not deal with creating the topology of the generated graph with using a known method. Furthermore, they used the pre-developed graph topologies for drawing their graphs without considering the effects of the node placements. However, in a real network, the positions of the nodes can be important for different aspects. Therefore, we assume that using a proven algorithm for a similar purpose can be beneficial for placing the nodes in our graph.

Assumption. 2: *Conceptual proximity for forming relations* Communities created in actual places, whether it is an offline or an online place (Lesser et al., 2000). Therefore, the positions of the members of the community is important. We can refer these positions as *conceptual positions*, since these positions can be determined according to being in a workplace, in a professional seminar or in a friendship gathering.

By using this assumption, we accept that the node placements and the *conceptual proximity* between these nodes is one of the most important factors to create relations, hence, the edges of the network. We take benefit of this assumption in our edge generation process.

Assumption. 3: Homophily principle Homophily (McPherson et al., 2001) is best described as creating more edges between similar nodes, by assuming that the similar nodes have more common features for being connected. Using this principle is an effort for generating synthetic networks that resemble real network properties.

We ensure that our generated networks use the homophily principle by forming the edges between the nodes with the help of the similarity of node attributes.

Assumption 4. Node attributes with values for attraction There are earlier studies that generated networks with node attributes as we have covered in our literature review. Though these examples contain node attributes, most of them did not pay attention to the idea of the negative/positive attributes and the effect of these attribute alignments on the edge formation process.

We think that having negative or positive values for node attributes contributes heavily to the edge generation phase of our method.

Assumption 5. Edge generation with power law degree distribution Most real world networks share some common properties. One of these properties is power law degree distribution (Newman, 2018). Some earlier network generators –mostly for social networks– used it for both for node and edge degrees, while some of the examples used it on either one of them.

Since we use a different approach on node generation phase, we decided to use power law degree distribution only on our edge generation phase. In our method, the nodes that have the most attributes and the least conceptual distance to other nodes create more edges, as the nodes with less attributes and more distance to other nodes create less edges.

3.2.2 Generation Parameters

Table 6 lists the five direct parameters that can be used to generate a network using AnatoliA. The number of nodes, N , and the number of attributes for each node, A , are used to create nodes for the graph G that will be created when the algorithm is complete. The parameter k determines the maximum number of nodes a cell can hold before it can be determined how far the nodes can reach to create an edge and if their conceptual distance is within a threshold before an edge is created using the maximum

distance, L , and threshold for proximity, t .

Table 6: Description of generation parameters.

Parameter	Description
N	Number of nodes
A	Number of general attributes
k	Maximum number of nodes in a cell
L	Maximum distance
t	Threshold for proximity

While these parameters can be used to adjust the algorithm for a specific context, it can be customized further. The diamond-square algorithm that is used to create a heightmap has a roughness parameter (ρ) to determine how steep the change between neighbor positions will be. Setting ρ to values closer to 1 creates larger changes between neighbor nodes. The algorithm also allows the function to be used for the generation of attribute values. While a normal distribution is used by default, it is possible to change the probability distribution to fit the requirements of a specific context.

3.2.3 Algorithm

AnatoliA consists of five consecutive phases; *grid generation*, *general attribute generation*, *node generation*, *edge generation* and *resulting graph generation & drawing*. Overview of all the phases is given in Figure 8. The explanations for generation parameters were given in Table 6.

Phase 1. Grid generation In order to place the nodes in a graph topology, we firstly need to create a grid matrix.

The initial step in the method is to use the diamond-square algorithm to create a heightmap that will be used to distribute the initial candidate positions for the nodes in the graph. This algorithm requires a grid of size $(2^N + 1)^2$ because it updates the midpoints as it operates on smaller sizes of grids. Therefore, the size of the grid is set to $m \times m$ where m is bound to the number of nodes N as given in Equation 7.

$$m = 2^{\lceil \log_2(\sqrt{N}-1) \rceil} + 1 \quad (7)$$

-
-
- 1: Generate the graph grid: - *Grid(..)*
Require: N
Return: A matrix of size $(m \times m)$ with uniformly random numbers in entries.
Every value in P is used to determine the number of nodes.

 - 2: Generate general node attributes: - *generate_general_attributes(..)*
Require: A
Return: An attribute list (α) of size A that contains general node attributes (a_1, a_2, \dots, a_A) .

 - 3: Generate the graph nodes: - *generate_nodes(..)*
Require: N, α
Return: A node list ($V \ni \{V_0, V_1, \dots, V_N\}$); where the number of attributes for each node is A , and the affinity level values for every attribute of every node is normally distributed among all the attribute affinity levels.
Every node in V is placed into matrix sequentially by using P .

 - 4: Generate the graph edges: - *generate_edges_with_similar_distance(..)*
Require: V
Return: A dictionary (E) in a key-value pair form, where key is a node from $\{V_0, V_1, \dots, V_N \in V\}$ and the value (also $\in V$) is a list of adjacent nodes.

 - 5: Create the graph: - *Graph(..)*
Require: V, E, α
Return: A graph (G) generated by V & E & α .
-

Figure 8: The overview of AnatoliA algorithm.

This approach creates cells greater than or equal to the number of nodes. Each cell will have some number of candidate positions with respect to the value that has been generated by the diamond-square algorithm for that cell.

The diamond-square algorithm that operates on a grid of size $m \times m$ briefly runs as follows. Initially, the four cells at the corners are set to random values. The algorithm then alternates between the diamond and square steps. In the diamond step, the grid is treated as a square, and the midpoint that lies on the diagonals of the four corners of the square is set to the sum of the average value of the corners and a random value. In the square step, the nodes that form a square are considered and the midpoint that lies in the vertical and horizontal axes of this square is set to the sum of the average corners of the square and a random value. These steps are performed sequentially in smaller squares and diamonds until all cell values have been set.

During realization of the diamond-square algorithm, the values are normalized to integers between 1 and k so that the next step can place a list of candidate positions to each cell. The variable k can be set to any integer value where $k > 1$ to parameterize the generation of the network.

In computer graphics, a heightmap represents a terrain where higher cell values represent higher elevations. AnatoliA treats higher values in the cells as more crowded and places candidate positions around the center of each cell randomly using a uniform distribution. These positions are kept in a list P and since $k \geq 1$, the number of candidate positions in P are greater than the number of nodes.

Phase 2. Generation of general attributes The attributes that the nodes of the graph can have are determined in this phase. The nodes can have A number of attributes where each attribute has an affinity level. The algorithm allows the values for these affinity levels to be set using a custom function so that it is possible to use any required probability distribution. The default function for the affinity levels employs a normal distribution with $\mu = 0$ and $\sigma^2 = 1$. These attributes are placed into a list α of size A where each attribute is denoted with an integer subscript of α . Each node is assigned such a list to decide the affinity levels of their attributes.

Phase 3. Node generation The nodes of the graph are generated with using the node-placeable positions list, P , and the attribute list, α . For every node, we add every

attribute from α . We also determine the values of the particular attributes in the nodes in this phase using the process described in previous phase.

The node is placed into a position in P sequentially, and the node is added into the node list, V , where it is used to generate edges in the next phase.

Phase 4. Edge generation For generation of the graph edges, a map of node adjacency, E , in a key-value pair format ($E = \{k, [v]\}$) is created. All the nodes in V arranged as the keys of the E at first.

The value for a certain key node in E , is actually a list of nodes that have an edge to that certain key node. These node lists for each key node are determined by using a combination of *attribute similarity*, s , and *conceptual proximity*, d , of the nodes. The edges between two nodes, $v_1 \in V$ and $v_2 \in V$, is determined by using a combination of $s(v_1, v_2)$ and $d(v_1, v_2)$.

Similarity of two nodes, $s(v_1, v_2)$, is calculated as follows:

- Affinity levels of same attributes in different nodes are multiplied:

$$s_{\alpha_1}(v_1, v_2) = V_1(\alpha_1) \times V_2(\alpha_1).$$

- This similarity is calculated among every attribute for the node pair:

$$s_{\alpha_2}(v_1, v_2) = V_1(\alpha_2) \times V_2(\alpha_2), \dots, s_{\alpha_i}(v_1, v_2) = V_1(\alpha_i) \times V_2(\alpha_i).$$

- When per-attribute similarity values are calculated, they are summed up to a general similarity between those nodes:

$$s(v_1, v_2) = \sum_i^A(s_{a_i}) = s_{a_1} + s_{a_2} + \dots + s_{a_A}.$$

- If this resulting similarity value, $s(v_1, v_2)$, is bigger than 0, then we can say that the nodes are similar. If the value is less than 0, then we can say that the nodes are dissimilar.

We use the properties of mathematical operations for our benefit in node similarity calculations. When two affinity level values have the same sign (both negative or both positive), the multiplication of these two values are always positive. Similarity value between two nodes are calculated as a summation of these particular values. Therefore, when more values have positive signs, it is likely that the resulting similarity value between the nodes will be bigger than 0. If one attribute is so close to the extreme

values, -1 or +1, then this attribute hugely affects the calculation of similarity value, hence, the edge generation phase between the nodes.

Conceptual proximity of two nodes, d , is a measure of whether these nodes are close enough to form an edge in the given context. It is calculated as follows in our method:

- The maximum distance between any two nodes according to their conceptual positions in the graph, L , is determined at first.
- A proximity value between two nodes, $d(v_1, v_2)$, is calculated:
$$d(v_1, v_2) = |p(v_1) - p(v_2)|/L$$
, where $p(v)$ is the position of node v .
- If this proximity value, $d(v_1, v_2)$ is less than, $t = 0.1 \times L$, then we can say the nodes are conceptually close. If the value is bigger than t , then we can say that they are far from each other.

In literature, this proximity value is determined according to the given context. While there are no consensus on how it can be determined whether two people are in close proximity, we assumed that the 10% of the maximum distance can be considered as a close distance in social relationships and decided to use it for our method. In different contexts, however, this proximity value can be decreased or increased. We evaluate the selection of this value later in Chapter 4.

With calculating these two values, there are two conditions for generating an edge between two nodes:

- If two nodes, v_1 and v_2 , are similar according to their similarity value, $s(v_1, v_2)$, then we can say that these nodes are similar enough to create an edge between one another.
- Moreover, if v_1 and v_2 are in close proximity according to their proximity value, $d(v_1, v_2)$, then we can say that they are close enough to form an edge on that conceptual distance.
- If these two conditions are met together, then an edge, $e_{(v_1, v_2)}$, is generated between v_1 and v_2 .

The values in the edge dictionary E , is filled one by one for every key node by using the properties mentioned in last paragraphs without disrupting the power law. Power law distribution is applied to the edge generation by creating a list of numbers. The numbers in the list start from the value of number of nodes and are divided until they reach one. The sum of the numbers in the list is set to the number of nodes as the last step.

For example, if there are 16 generated nodes, then the divided numbers in order are 8, 4, 2 and 1. The number, 8, is increased by one to 9, for setting the sum as 16. The power law list includes {9, 4, 2, 1}. It means that 1 node has 9 edges, 2 nodes have 4 edges, 4 nodes have 2 edges and 9 nodes have 1 edge. The power-law degree distribution in edge generation is ensured with this way.

The edge dictionary at final, E , with the node list, V are used to generate the resulting graph and draw it.

Since this edge generation phase is the core of our method, we also give it as a detailed algorithm in Figure 9.

Phase 5. Resulting graph generation & drawing For the last phase, a graph $G : (V, E, \alpha)$ is created. A color list is also created randomly for every connected component in G .

By using a dictionary structure, where the keys are the connected components of the resulting graph, and the value corresponds to each key is a list of nodes that the key component contains; the color of the nodes are determined.

As a summary; positions of the nodes are determined at *Phase 1* with respect to *Assumption 1*. The attributes that the nodes have are generated at *Phase 2*. The nodes in G are generated with their names, grid positions and general attributes at *Phase 3*. The edges between the nodes are generated using the *Assumptions 2, 3, 4 and 5* at *Phase 4*. The resulting graph and the connected components that the graph nodes belong to are drawn at their respective positions at *Phase 5*.

3.3 Results

In this section, we firstly give some example graphs that are generated with the help of our method. We also evaluate our generated graphs according to these listed factors:

```

generate_edges_with_similar_distance( $N, D$ )
1:  $max_x = \max(D_x), max_y = \max(D_y)$ .
2:  $L = \sqrt{max_x^2 + max_y^2}$ .
3:  $E = \emptyset$ 
4:  $V' = sorted\{V\} \rightarrow \min D(v_1, \dots, v_N)$ 
5:  $E'_K = \{v'_{s_0} : \{v'_{s_0}, v'_{s_1}, \dots, v'_{s_N}\}, \dots, \{v'_{s_N} : \{v'_{s_0}, v'_{s_1}, \dots, v'_{s_N}\}\}$ 
6:  $div_V = find\_divisors(N)$ 
7:  $div_E = reverse(div_V)$ 
8: for  $x = 0, 1, \dots, length(div_V)$  do
9:   for  $y = 0, 1, \dots, div_V[x]$  do
10:    if  $y \geq length(V')$  then
11:       $v_s = random(V')$ 
12:    else
13:       $v_s = V'[y]$ 
14:    end if
15:     $E' \leftarrow E'_K[v_s]$ 
16:    for  $z = 0, 1, \dots, div_E[x]$  do
17:      if  $in\_circle(v_s, E'[z], L)$  and  $attracted(v_s, E'[z])$  then
18:         $e_s \leftarrow E'[z]$ 
19:      end if
20:    end for
21:     $E[v_s] \leftarrow e_s$ 
22:     $V'.remove(v_s)$ 
23:  end for
24: end for
25: return  $E$ 

```

Figure 9: Edge generation in AnatoliaA.

- One of the first evaluation steps is to find out if our generated graphs are met the assumptions we have in the first place. For that, we present assumption-result evaluation and use some metrics on our generated graphs that are created with different generation parameters.
- We present running times of our algorithm with respect to different generation parameters.
- Secondly, we compare these properties with similar graph generators that are also created with same mutual generation parameters by using different metrics.
- Thirdly, we compare the communities formed by AnatoliA with the communities formed by the similar generators by using attribute-aware community detection (CD) methods.
- We also evaluate our two assumptions concerning the communities in our generated graphs. If we assume that we know the ground-truth communities, which CD method will perform better in that case. And if we assume that ground-truth communities are unknown, what will be the result of evaluation metrics for our method.
- Lastly, we evaluate our model properties and community structure compared to a real graph dataset called “Sinanet” (Jia et al., 2017).

In the next section, we give example generated graphs.

3.3.1 *Generated Graphs*

You can see an example generated graph in Figure 10. Our generated graphs show that there are highly connected one or two nodes that are mostly central to the graph. The vast difference in colors show that there are many connected components in the graphs.

A detailed look on particular nodes shows that some nodes form no edges for connecting other nodes. This may have been arisen due to the fact that those nodes have not any similarity to the nodes that are most close to them. This shows the importance of the node placement approach.

We made a more thorough analysis on generated graphs by looking at the model properties with different generation parameters in the next sections.

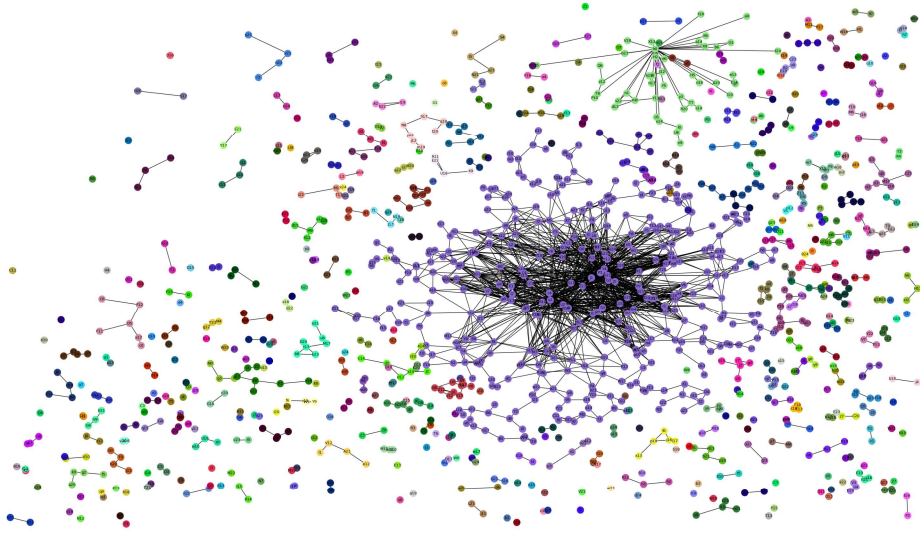


Figure 10: A generated graph example with $N = 1250$ nodes and $A = 100$ general attributes. Our original grid layout approach is used to draw the graph nodes and edges. Connected components are denoted with different colors.

3.3.2 Assumptions and Model Properties Evaluation

We try to evaluate our model according to its own creation rules. Our assumptions are listed in Section 3.2.1. We present these evaluations, and give the measurement metrics used in model properties evaluation in this section.

3.3.2.1 Homophily Measurement

For measuring the homophily in our graph that is aligned with our *Assumption 3*, we employed the measure explained in Easley and Kleinberg (2010), which is also used in ANC (Largeron et al., 2015) generator. We created our own interpretation for this calculation process, calculated Homophily Measure (HM) for our graph and created a new metric called Normalized Homophily Measure (NHM) by normalizing the value of HM. We calculate these measures using the formulas:

$$HM = \sum_{i=1}^A (O_i - E_i) \quad (8)$$

$$NHM = \frac{HM - \min(HM)}{\max(HM) - \min(HM)} \quad (9)$$

where O_i is the ratio of the number of observed edges according to the given attribute i over total edge count, and E_i is the probability of two nodes create edges with the given attribute i is similar among them. While calculating O_i we assumed that if the multiplication of the affinity levels of the same attribute between two nodes is positive, then the edge is created according to that attribute in a certain fraction. E_i is calculated under the assumption that if the both affinity levels of the same attribute for two different nodes are of same mathematical sign (i.e. both positive or both negative), then they can form an edge according to that attribute.

We have listed the homophily of the generated graphs in Table 7 and Table 8 for different number of nodes and for different number of meta-communities.

3.3.2.2 Normal Distribution of Affinity Levels

According to our *Assumption 4*, the affinity level values of the node attributes are distributed using normal distribution. We control how many attributes that each node will have with A parameter.

You can see histogram plot for the frequencies of the attribute values for two example graphs with $N = 500$ and $N = 1000$ while A is constant as 10 from Figure 11.

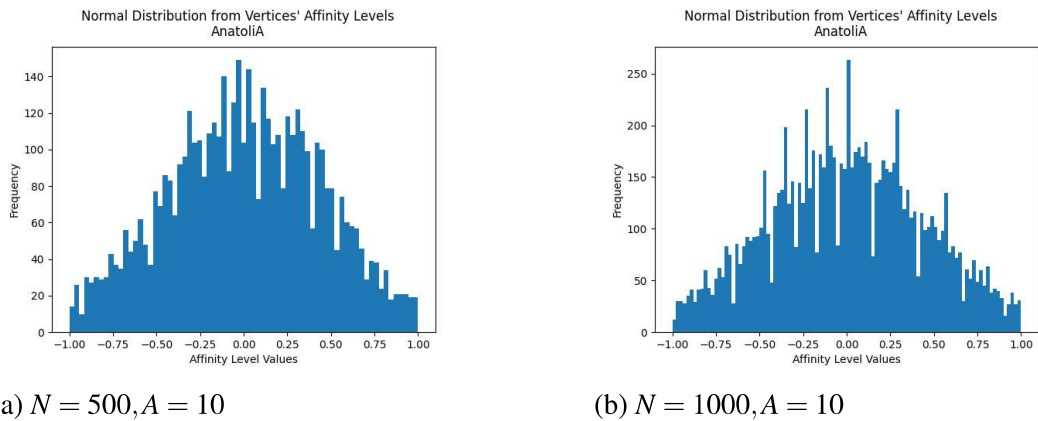


Figure 11: Normal distribution of node-attribute affinity values with two example graphs.

3.3.2.3 Power-Law Edge Generation

According to our *Assumption 5*, edges of our graphs are generated with a power-law degree distribution. We wrote the details of the generation process in Section 3.2.3.

We compared our generation process with LFR benchmark (Lancichinetti et al.,

2008) that also uses power-law degree distribution. You can see the density plots for edge degree distributions between AnatoliA and LFR benchmark from Figure 12.

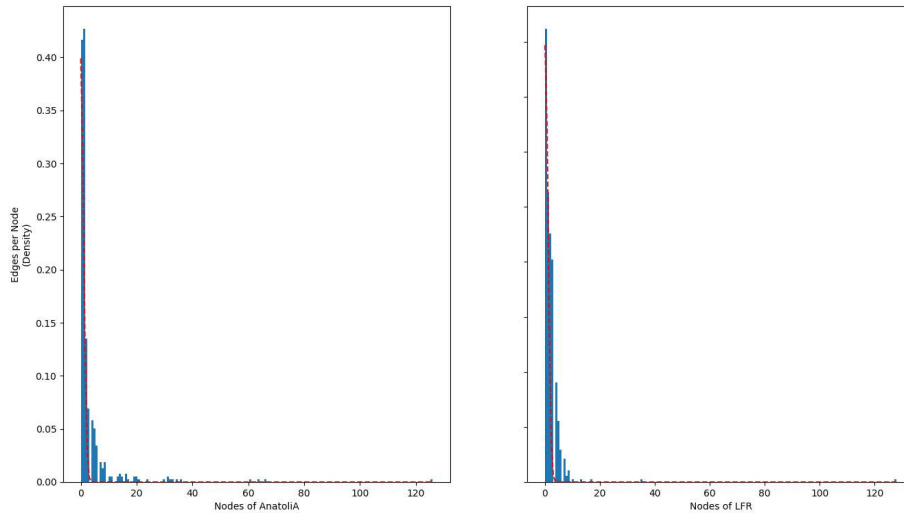


Figure 12: AnatoliA vs. LFR for edge degree distributions. They both use power-law degree distributions.

If we analyze this figure, we can see that while the power-law degree distribution in LFR benchmark is very strict, we created a more relaxed version. In our version, some nodes can create edges to get a varied generation result. As an implication of that, the power-law curve is more smooth than the curve of LFR example which is more steep than ours as you can see from the given plots.

3.3.2.4 *General Properties of Our Model*

In this section, we evaluate the general characteristics of our generation approach. To do that, we firstly explain the model properties briefly.

Average clustering coefficient (CC) (Watts and Strogatz, 1998) is a measure of how tightly the nodes in a network cluster together. The average number of steps along the shortest paths for all potential pairs of nodes is known as *average shortest-path length (SPL)* (or *characteristic path length*) in networks. It is a way to assess how effectively the information transports over a network. *Average degree* gives the average node degree in the whole network. The clustering coefficient, the average shortest path length and the average node degree are three most reliable metrics of a network's topology. The *graph diameter* is the average eccentricity of the nodes in a

graph, which the eccentricity of a node can be defined as the maximum distance from the node to all other nodes.

We also used centrality measures for evaluation. The total percentage of all-pairs shortest paths that pass through a node represents its *betweenness centrality* (Brandes, 2001). The ratio of the average shortest path distance to a node over all reachable nodes is its *closeness centrality* (Freeman, 1978). We used the average of these two centrality values for evaluating the generated graphs.

As the categorization of Xiang et al. (2021) is applied; CC, average degree, betweenness and closeness centrality measures give the characteristics of the node distributions. Edge count, diameter and SPL measures are considered as general graph statistics. There are many more properties that can be covered, however, we consider these properties sufficient for our purpose.

These structural measures give most of the characteristic properties of our generated graphs. We have listed these resulting measurements for our model in two different tables. In Table 7, we listed graphs with variable number of nodes ($250 \leq N \leq 2500$) and constant number of node attributes ($A = 10$). We subsequently listed generated graphs with constant number of nodes ($N = 500$) and variable number of node attributes ($10 \leq A \leq 100$) in Table 8.

We also have a separate result for an example generated graph with $N = 10^4$, $A = 20$, as (*Clustering Coeff.* = 0.121, *Degree* = 4.564, *Betweenness Centrality* = $5.73e^{-5}$, *Closeness Centrality* = 0.011, *Edge count* = 22819, *Conn. Component Count* = 3716, *Diameter* = 40, *SPL* = 8.3, *NHM* = 0.824). Due to the runtime issues, we do not have enough simulation data for that many nodes to calculate the mean values, thus we have listed this result separately. We also evaluate our running times in next section.

3.3.2.5 *Running Time Evaluation*

A computer with these specifications are used for simulating the generation process for evaluating runtimes of our algorithm: Intel-i7 8-core processor, 32GB RAM, NVIDIA 1070 GTX 8GB GPU.

We have listed the maximum running times for our algorithm out of 10 simulations for every different generation parameter used. You can see the results from Table 9.

We have also singular running time results. Our algorithm spent maximum time of

Table 7: Model properties and assumptions. Average (Avg.) measurements come from 10 simulations performed with the same graph properties as $A = 10$. Mean values are given in table.

Measures (Average)	$N = 250$	$N = 500$	$N = 1k$	$N = 2k$	$N = 2.5k$
<i>Clustering Coeff.</i>	0.184	0.189	0.178	0.174	0.138
<i>Degree</i>	3.364	3.848	4.639	5.163	3.729
<i>Betweenness Centrality</i>	$7.15e^{-3}$	$2.25e^{-3}$	$9.96e^{-4}$	$5.61e^{-4}$	$2.04e^{-4}$
<i>Closeness Centrality</i>	0.066	0.049	0.036	0.027	0.015
<i>Edge Count</i>	420	962	2319	5163	4661
<i>Conn. Component Count</i>	57	130	277	574	917
<i>Diameter</i>	18	19	25	28	31
<i>SPL</i>	5.900	5.507	6.126	7.370	7.150
<i>NHM</i>	0.891	0.876	0.862	0.858	0.855

Table 8: Model properties and assumptions. Avg. measurements come from 10 simulations performed with the same graph properties as $N = 500$. Mean values are given in table.

Measures (Average)	$A = 10$	$A = 25$	$A = 50$	$A = 75$	$A = 100$
<i>Clustering Coeff.</i>	0.189	0.171	0.159	0.169	0.155
<i>Degree</i>	3.848	3.771	3.794	3.833	3.836
<i>Betweenness Centrality</i>	$2.25e^{-3}$	$2.48e^{-3}$	$2.34e^{-3}$	$2.43e^{-3}$	$2.43e^{-3}$
<i>Closeness Centrality</i>	0.049	0.048	0.051	0.052	0.050
<i>Edge Count</i>	962	942	949	958	959
<i>Conn. Component Count</i>	130	137	137	137	134
<i>Diameter</i>	19	19	19	19	18
<i>SPL</i>	5.507	5.806	5.504	5.569	5.668
<i>NHM</i>	0.876	0.847	0.832	0.820	0.819

Table 9: Maximum runtimes (in seconds) of the algorithm for given graph properties (out of 10 simulations).

	$A = 10$	$A = 25$	$A = 50$	$A = 75$	$A = 100$
$N = 250$	0.22	0.39	1.27	2.04	4.05
$N = 500$	0.96	1.45	3.20	5.55	9.54
$N = 1000$	7.74	9.02	12.60	17.91	26.48
$N = 2000$	51.16	54.05	65.90	81.98	102.27
$N = 2500$	94.95	105.20	115.65	126.63	141.09

244.11 seconds for ($N = 2000, A = 200$) condition, and spent 860.39 seconds for ($N = 5000, A = 10$) condition.

As it can be seen from these results, the runtime increases as the number of nodes increases. As the number of node attributes increases, there is also an increase in running time. It is evident that the increasing factor of node count effect is more than the effect of attribute count.

When we investigate the source reason of the running time spike with the increasing effect of the node count, we see that the edge generation part, pairwise distance calculation between the nodes, and summation calculations in various points of the algorithm are some of the most notable reasons of it. Since we are doing many calculations to decide if we need to form an edge according to the node attribute values and the distance between the nodes, our algorithm spent a lot of time in that process. We discuss this situation also in Section 3.4.

3.3.2.6 Time Complexity Estimation

AnatoliA has two properties that can affect the time complexity of the method the most, the number of nodes, N , and the number of general attributes, A . The method has five phases, but the most important phase is the edge generation, therefore, we analyze it here.

In this phase, we have three loops inside of each other. Before any loop, *find_divisors* function finds the numbers div_V and div_E . div_V is a list of numbers, and these numbers are found by consecutively halving N until the numbers reach to 1. Because of this halving process, the length of div_V and div_E both converge to $\log(N)$. Since we used $length(div_V)$ as our upper bound in the first outer loop, it completes its execution in $O(\log(N))$ time.

In the middle loop, the upper bound is $div_V[x]$, where in every iteration, $div_V[x]$ is bound to $1, \dots, \frac{N}{16}, \frac{N}{8}, \frac{N}{4}, \frac{N}{2}$. Since the summation of this list of numbers, $\sum_{a=1}^{N/2} a/2 = 1 + \dots + \frac{N}{16} + \frac{N}{8} + \frac{N}{4} + \frac{N}{2}$ converges to N , the contribution of this loop to total time will be $\simeq O(N)$.

The inner loop uses $div_E[x]$ as its upper bound, which is a list with the reversed order of numbers of $div_V[x]$. Because of this reason, the summation of the numbers inside the list also converges to N , and hence, its contribution will also be $\simeq O(N)$.

Inside the inner loop, we have two functions, *in_circle*, which checks if two nodes are close enough to form an edge, and *attracted*, which checks if two nodes are similar enough to form an edge.

in_circle function works as this: it takes the difference of the conceptual positions of two nodes in question and checks if this difference is smaller than our proximity threshold. Since the pairwise distances between all nodes are already calculated before the loops, the operation spends $O(1)$ time.

attracted function, multiplies every attribute affinity level values for two nodes in question and sums them up. Since every node has A number of attributes, and the multiplication&summation operation spends constant time, the whole operation lasts $O(A)$ time. When A is small compared to N , we can take its contribution as constant. In greater values, we also need to take its contribution into account.

When the contribution of every part of the edge generation phase is taken into consideration, we can conclude the worst-case time complexity estimation of AnatoliA as $T(N) = O(N^2 \log(N))$ when $A \ll N$, and $T(N,A) = O(AN^2 \log(N))$ when $A \sim N$. Since all the other phases of our method spend at least $O(N)$ and at most $O(N^2)$ time, we can assume our upper bound comes from mostly our edge generation phase.

3.3.3 Comparison with Other Graph Generators

We compared our graph generator with earlier generators with respect to different evaluation metrics.

Firstly, we compared our algorithm with ANC (Largeron et al., 2015), DANCer (Benyahia et al., 2016), GenCAT (Maekawa et al., 2021) and X-Mark (Citraro and Rossetti, 2021) with respect to their model properties. We selected these generators since they create graphs with communities and node attributes, they use similar approaches with us for their generation process and the source codes or the applications of their algorithms were easy to access and use.

Secondly, we compared our algorithm by detecting its communities using various community detection methods. For that evaluation, we used the same generators we mentioned in previous paragraph. We employed three different evaluation on community detection (CD) methods:

- We used CD methods that take the node attributes into account (i.e. attribute-aware CD methods) while comparing the communities formed by other generators with the communities formed in our method. With this step, we find out which earlier generator is more similar to our approach in community detection aspect.
- We assume that the connected components as our ground-truth communities and compare them with the communities that are found by various CD methods.
- We assume that we do not know the ground-truth communities in our generated graphs and evaluate our communities with some metrics that can be used in these situations.

3.3.3.1 Model Properties Similarity Evaluation

In this section, we try to evaluate similarity of AnatoliA to earlier generators with respect to their model properties. For that reason, we simulate graphs as $N = 500, A = 10$, for every graph generator. You can see the results of this simulation from Table 10.

Table 10: Model properties comparison between graph generators. Average measurements come from 10 simulations performed with the same graph properties for each graph as ($N = 500, A = 10$). Mean values are given in table.

Measures (Average)	ANC	DANCer	GenCAT	X-Mark	AnatoliA
<i>Clustering Coefficient</i>	0.108	0.158	0.547	0.109	0.189
<i>Degree</i>	4.492	8.248	29.558	11.465	3.848
<i>Betweenness Centrality</i>	$7.11e^{-3}$	$4.83e^{-3}$	$2.26e^{-3}$	$3.61e^{-3}$	$2.25e^{-3}$
<i>Closeness Centrality</i>	0.224	0.296	0.476	0.360	0.049
<i>Edge Count</i>	1123	2074	7389	2866	962
<i># of Connected Component</i>	1	1	1	1	130
<i>Diameter</i>	9.3	6.1	4	4.875	19.1
<i>Average SPL</i>	4.539	3.420	2.125	2.796	5.507

From the results, we can concur that, according to the model properties, ANC generator is the one that has the most resemblance to AnatoliA. From eight measures, five of results for ANC is very similar to our method’s results. In one measure, – specifically in clustering coefficient– the most similar generator is DANCer. Also in betweenness centrality measure, the result of GenCAT generator nearly identical to ours.

In one measure (i.e., the connected component count), however, we did not compare any generator with our generator, since all of the other generators produce connected graphs, different than ours.

3.3.3.2 *Community Similarity Comparison*

We selected I-Louvain (Combe et al., 2015) and EVA (Citraro and Rossetti, 2019) attribute-aware CD methods for detecting the communities formed by AnatoliA and other generators. We employed widely used community similarity evaluation metric, Normalized Mutual Index (NMI), for evaluation of community similarities.

You can find the detailed results of our comparisons on Table 11 for attribute-aware CD methods.

It is evident in these results that the most similar generators to our generator are DANCer and GenCAT in detected community aspect. When NMI results of I-Louvain CD method are analyzed, X-Mark is also has a subjectively high result. For EVA CD method, NMI comparison puts the DANCer generator in second place. These results may indicate that DANCer generates mostly similar communities to our method.

Table 11: NMI scores of similarity evaluation for AnatoliA in comparisons with other graph generators using node-attribute aware CD methods. Mean values are given in table.

CD Methods:	<i>I-Louvain</i>	<i>EVA</i>
<i>ANC</i>	0.1	0.485
<i>DANCer</i>	0.159	0.862
<i>X-Mark</i>	0.103	0.558
<i>GenCAT</i>	0.048	0.918

3.3.3.3 *Connected Components as Ground Truth Communities*

As we described before, we assumed that the connected components are the ground-truth communities in our generated graphs. We compared these ground-truth communities formed by AnatoliA to the communities detected by CD algorithms. In this comparison, we also used general CD methods (i.e., CPM (Palla et al., 2005), Louvain (Blondel et al., 2008), and APAL (Doluc a and Oğuz, 2021)) which do not consider the node attributes while detecting the communities. We mainly do that for understanding that if the structural communities detected by general CD methods

will have a similarity score more than the ones that are found by attribute-aware CD methods. We again employ the metric NMI, for similarity evaluation, and also (2) Adjusted Rand Index (ARI) measure.

For both NMI and ARI, the lower and upper bounds are 0 and 1, respectively. Zero means that there are not any similarity between the given community and the community detected by the CD method. One as the result means maximum similarity between two communities, the given and the detected one. You can find our results for this part in Table 12.

Table 12: Evaluation results for AnatoliA ground truth communities compared to the communities found by CD methods. Mean values are given in table. ($N = 500$, $A = 10$).

Measures:	<i>NMI</i>	<i>ARI</i>
<i>CPM</i>	0.785	0.354
<i>Louvain</i>	0.848	0.205
<i>APAL</i>	0.257	0.211
<i>I-Louvain</i>	0.890	0.310
<i>EVA</i>	0.708	0.128

For all of the CD methods in this assumption, we accepted the total connected components as the ground-truth communities of our generated graphs. This assumption comes from the fact that in social relations, being in a community means that being highly connected to other nodes (Yang and Leskovec, 2015). We as people, all know somebody through somebody else. Therefore, knowing a person means that we have some form of acquaintance to the people who knows that person and it creates a weak relation between us and them. Because of that, we used that assumption for this part.

From the results, we can see that according to NMI metric, I-Louvain is the one that finds most similarities. The close second in this metric is Louvain method. In ARI metric, CPM is the one that finds two different community list, ground-truth and detected, as most similar. In this metric, the second score again comes from I-Louvain method.

These results may be an indication that the structural communities formed by our nodes and edges are mostly overlapped with the functional ones created with the effect of the node attributes if we take our ground-truth community assumption as true.

In the next part, we assume that we do not know the ground-truth communities of

our generated graphs.

3.3.3.4 *Unknown Ground Truth Communities*

If ground-truth communities are not present, then different metrics can be used to evaluate the community quality of the graphs. The Silhouette Coefficient (Rousseeuw, 1987) is an example of such evaluation metric, where a higher Silhouette Coefficient score relates to a model with better defined communities. It is a score between -1 and +1, where the positive scores near to +1 means that the graph is densely clustered. Near-zero scores can indicate overlapping communities.

Other than the Silhouette Coefficient, there are two other metrics we used to evaluate the community structures of our graphs. One of them is Calinski-Harabasz score (Caliński and Harabasz, 1974) –also known as the Variance Ratio Criterion–, where a higher score relates to a model with better defined communities. The Davies-Bouldin index (Davies and Bouldin, 1979) can also be used to evaluate the graph models, where a lower index relates to a model with better separation between the communities. The lowest possible score for Davies-Bouldin index is zero, which indicates a better partitioning.

For all of these metrics, the number of clusters to detect must be predetermined to measure the community quality. We used more than one number for the cluster count to evaluate the effects of this change. The mentioned metrics are applied to the results of a K-means clustering algorithm. We used the node attribute values list per graph node for the training instances to cluster with K-means algorithm.

Table 13: Evaluation results for AnatoliA when ground truth communities are assumed to be not known. C_{CC} is the connected component count of the given graph ($N = 500$, $A = 10$). Mean values are given in table.

K-means Label Count =	3	$A/2$	A	C_{CC}
<i>Silhouette Coefficient</i> ($-1 < sc < 1$)	0.075	0.075	0.081	0.090
<i>Calinski & Harabasz Score</i> ($0 \ll ch_score$)	40.422	34.312	26.933	8.464
<i>Davies & Bouldin Index</i> ($0 \leq db_index$)	2.859	2.472	2.117	1.222

It can be seen from Table 13, that the Silhouette Coefficient increases in direct

proportion to the number of predefined number of labels. As it is very close to 0, it can be speculated that the communities in our generated graphs may be defined as overlapping communities. Calinski-Harabasz score and Davies-Bouldin index are decreasing with the increasing number of labels. These two together give us mixed results about our community quality.

3.3.4 Evaluation with Real Dataset

For evaluating our generator further, we used a real network dataset, Sinanet (Jia et al., 2017). It is a microblog user relationship network with 3490 nodes, 30282 edges and 10 numerical attributes per-node.

There are two parts for this evaluation. In the first part, we evaluated the model properties with respect to Sinanet network. In the second part, we evaluated if the communities generated in our model are similar to the ones in Sinanet network.

3.3.4.1 Model Properties Evaluation

For evaluating our method, we generated networks with the same number of nodes and node attributes with Sinanet, where $N = 3490$ and $A = 10$ for every node-attributed graph generator (i.e. ANC, DANCer, GenCAT, X-Mark and AnatoliA) we have. We compared these generators by using maximum mean discrepancy (MMD) over clustering coefficient, degree, betweenness and closeness centrality values, and by using absolute mean difference (AMD) over edge count, connected component count, diameter and average shortest path length values for their model properties. The results for this evaluation is shown on Table 14.

From the results of model evaluation, we can claim that the most similar graphs to Sinanet are generated by GenCAT generator. X-Mark is also very successful on edge count and diameter measures. The most similar graphs on degree measure are generated with DANCer generator.

Nevertheless, our generator AnatoliA is a close contender in some measures, specifically clustering coefficient and betweenness centrality measures as we denote this fact by underlining those scores in the results table. Other than this result, when we look at the connected component count measure, we see that all other generators have an AMD value as 23. This is due to the fact that all other generators create connected

Table 14: Model properties comparison between graph generators using Sinanet benchmark. Measurements come from simulations performed with the same graph properties for each graph as ($N = 3490$, $A = 10$). ANC and DANCer graphs are generated in their respective applications and exported into our generator for comparison.

MMD Over	ANC	DANCer	GenCAT	X-Mark	AnatoliA
<i>Clustering Coefficient</i>	0.043	0.035	$8.39e^{-04}$	0.014	<u>0.004</u>
<i>Degree</i>	0.078	0.013	0.014	0.041	0.119
<i>Betweenness Centrality</i>	$8.53e^{-07}$	$3.67e^{-07}$	$7.15e^{-09}$	$5.91e^{-08}$	<u>$2.55e^{-08}$</u>
<i>Closeness Centrality</i>	0.044	0.026	0.002	0.006	0.233
AMD Over					
<i>Edge Count</i>	19647	14463	38383	8046	19557
<i># of Connected Components</i>	23	23	23	23	999
<i>Diameter</i>	5	1	2	0	38
<i>Average SPL</i>	2.513	1.236	0.215	0.563	7.359

graphs. However, the graphs generated by AnatoliA are disconnected graphs. The AMD over connected component count is higher in our method than in any other generator, as shown in the table, but we still identify it as the most successful one because the Sinanet network is also a disconnected graph.

3.3.4.2 Community Detection Evaluation

To evaluate the communities formed by AnatoliA, we used the attribute-aware CD methods, I-Louvain and EVA, to detect the communities in Sinanet, and compare these detected communities with AnatoliA and same generators from previous section. We used NMI measure for evaluating the community similarity. Results of this comparison is shown on Table 15.

Table 15: NMI scores of community detection comparison between graph generators using Sinanet benchmark. Measurements come from simulations performed with the same graph properties for each graph as ($N = 3490$, $A = 10$). ANC and DANCer graphs are generated in their respective applications and their data are exported into our environment for comparison.

CD Methods:	<i>I-Louvain</i>	<i>EVA</i>
<i>ANC</i>	0.009	0.106
<i>DANCer</i>	0.013	0.147
<i>GenCAT</i>	0.014	0.441
<i>X-Mark</i>	0.009	0.141
<i>AnatoliA</i>	0.065	<u>0.286</u>

From the results, we can say that when the communities are detected by the I-Louvain method, the graphs generated by our method have the most similar communities to Sinanet. When EVA method is used for community detection, AnatoliA is second after GenCAT generator. From these results, we can argue that our method creates networks that have real-world network properties on community aspect.

3.4 Discussion

We designed an algorithm to create synthetic networks with communities and node attributes. Our algorithm can create the edges between the nodes with respect to two aspects from the social context, namely the proximity and the homophily. The number of generation parameters is small. However, since our algorithm uses object oriented programming in every step of the process, all the internal parameters can be tweaked or there can be some additions when needed. As an example, we used power-law degree distribution for the edges and normal distribution for the values of affinity levels in the node attributes. If the users of our algorithm want to change one of these behaviors, only thing they need to do is changing a little part in one of the methods for one of the source code classes. After that change, all the other parts of the algorithm work accordingly. In this sense, AnatoliA is very easy to alter and can be modified with the needs of the users.

To achieve our results, we performed experimental simulations with different generation parameters. We try to estimate the significance of our algorithm by evaluating its model properties and by comparing it with earlier similar network generators such as LFR, ANC, DANCer, X-Mark and GenCAT. We also try to evaluate how the node attributes and edge generation process affects the communities by using both general and attribute-aware CD methods such as CPM, Louvain, APAL, I-Louvain and EVA. We further evaluate our generator with comparing it with a real-world network, Sinanet. We used different measures for graph properties and community assessment and try to devise our own measures such as NHM when there is none applicable.

There are some limitations we can address regarding our research and the opportunities to improve our research more in future.

As a first limitation, our algorithm creates static graphs. However, in most cases, the graphs and the community structures in them change over time or over different factors. One of the articles (Benyahia et al., 2016) we surveyed actually created their generator with dynamic communities in mind. Hence, the merge and split interactions between the communities must also be covered. Moreover, the node attributes are also subject to change if we think of individuals. People tend to dislike some things which they like in the past, or vice versa. Their specific traits can also change with time. An algorithm which also considers these changes can be very beneficial for creating a dynamic graph generator.

We created an algorithm that only work on undirected unweighted graphs. It is a limitation in many aspects, such as the lack of directed information propagation in social networks (Schweimer et al., 2022) can be a missing functionality in a network generator such as ours. If we think that the nodes as people in a social network, they chose to share information with another person in one-way for certain cases. Some sensitive information is kept hidden from the general public, or in this case, the community the person belongs to. We created the affinity levels of the node attributes with this mindset, but there can be some public and some private node attributes by taking this knowledge into consideration. This intimacy between individuals is an aspect yet to be covered in future for better graph generators.

Another limitation can be our node placement algorithm. We used diamond-square algorithm to place the nodes in our generated graphs. Yet, in this usage, we do not place the nodes considering the proximity between nodes, the node similarities, or the power-law edge generation aspect. Placing more central nodes to the correct places in the graph can create more connected graphs, or better community structures. Depending on this same situation, some nodes do not have any edges. Provided that these nodes are placed into the graph with better positions, these problems may be overcome.

One of the most important limitations of our study was the long running time of our algorithm. Since we create a graph generator, time complexity of it is also very important. Our first aim was to create an algorithm which generates graphs with communities and node attributes, and because of that, the running time was the least of our concerns at first. However, we can address this issue by creating a more efficient algorithm in future. Using parallel-processing in the generation of synthetic graphs is

efficient mostly (Bressan et al., 2013), so it can be a solution for our problem, too.

Generating large graphs can also come with other problems such as improper graph visualizations or unsuitable graph layouts. Also, there are not many research examples that show the contributions of the node attributes in their graph visualizations. These problems are also present in our work. We used our grid layout approach for showing the nodes at a better separated positions, however, we accept that our approach has also its own shortcomings.

When we increase the number of general attributes for graph generation, we usually expect the number of connected components to increase as well, since it generally decreases the likelihood of node similarity. However, the number of connected components in the graphs generated by our method does not change significantly with respect to the number of general attributes. We can also evaluate our generator from this perspective.

We can count the conceptual proximity and node similarity calculations as our last limitations. For calculating the proximity, we benefited the physical proximity rules from social context (Stopczynski et al., 2018). The humans can accept themselves near to or far from from each other according to the place they are in. The effect of this assumption must be evaluated with respect to the usage area of our generator. Different use-cases can have their own proximity calculations. We solved this problem in our generator by using an external ratio value for the conceptual proximity; however, a fully different proximity calculation approach may be adopted, since it is a subjective solution. Therewithal, measuring node similarity in accordance with the values of the node attributes may have its disadvantages. One can think that some of the node attributes influences the edge generation phase more than other attributes in the graph. In the future, implementing the node attributes with different importance or giving different weights to them may be a better approach.

One of the most important ideas we consider while we are creating AnatoliA is the broad spectrum of its usage areas. We employed it mostly in social context, however it can be applied to different networks with minimum adjustments. AnatoliA is a robust, easy to use and reliable synthetic graph generator with communities and node attributes and it is implemented in one of the most used programming languages nowadays with object oriented programming principles. We hope that AnatoliA and further iterations

of it can be used as a base network generation solution for various applications in AI, computer games, biology and communications.



CHAPTER 4 : APPLICATIONS OF THE PROPOSED ALGORITHM

4.1 *Further Improvements of Our Method*

We start this chapter by discussing the advancement opportunities of our generation algorithm. These ideas improve our method significantly, in one or more application areas. We briefly debated some of the concepts we discussed here also in Chapter 1 and Section 3.4.

The first concept we wanted to include in our method is edge weight. We can come across with weighted social networks in different contexts (Murase et al., 2014; Li et al., 2020). In these studies, the weights often define the importance of the edges in network. Large or small weight values may indicate that these edges may have different priorities depending on the situation. If our method contains the weight concept, then every node corresponding to an NPC, can have relations with other nodes with different priorities. It is important, since this rule can be present in real social networks (Bellingeri et al., 2023).

Related with the weights, an edge in a network can also have a direction. Directed networks are analyzed in different studies (MacKay et al., 2020; Schweimer et al., 2022). Direction in an edge means that the information or context in a node can only flow or spread out from that node to another, but not vice versa, according to the given direction. The edge directions is also another major concept in networks. Its usage in our method can also be beneficial since there can be one-way relations in NPC networks, such as interacting with in-game merchants, tavern owners, quest-givers and such.

One other concept we can discuss is overlapping communities. It is studied in different works (Xie et al., 2013; Doluca and Oğuz, 2021), in a context where joining more than one community at the same time is significant. We mostly detected the communities in our method according to the edge formations and the node attributes, however, overlapping communities concept can also be covered. NPCs in games can also have multiple communities such as the personal party they are traveling with, a guild that includes other NPCs with the same professions they belong to, a city or a part of a city where they live, and so on.

If we review earlier studies, we can see that there are studies concerning the directed weighted networks with overlapping communities (Lancichinetti and Fortunato, 2009a). Therefore, combining these three concepts is very common in academic research on graphs. They are also commonly combined and used for NPC networks in games. We can also include one other topic in here, which we covered in Chapter 2, which is dynamic networks. In dynamic networks, the nodes, the edges, the attributes or the communities can be subject to change over time or with different environmental effects. The NPCs are not static, hence, they are expected to change their behaviors with time and other effects.

Other than these ideas, we also come up with different ideas using social theories. The most prominent one for us is the usage of different attribute visibility. From social context, we can think that people tend to share their own traits if they are publicly visible or knowable (Lewis et al., 2008). Some people choose not to share their private and sensible information to all of their related people. Therefore, creating two different node attribute visibility, private and public, and use them accordingly in edge formation process can be a better approach for our method. For instance, if two nodes are connected with a weighted edge, and if these two nodes have many mutual public attributes, then one node can choose to share its private attributes with the other one, and according to these private attribute values, the edge weight can be increased. This ensures a stronger connection between two nodes with many common traits, both public and private.

The other important idea we come up with is about the edge formation process in our method. The details of our edge formation method can be read from Chapter 3. If we want to summarize, we have a summation formula that is used to evaluate the similarity between two different nodes with respect to their attribute affinity values. In this formula, we assume that every attribute affects the node similarity equally. However, if we give weights to the node attributes, and calculate the node similarity with proportional to these weights, then the calculation will represent the real world better. If we think about real life relationships, some personal traits affect more than the others when we want to connect with other people, and it should be similar in NPC networks.

The last idea we wanted to include in our method is creating different proximity

calculation options according to context at hand. For now, our method decides that two nodes are close enough to form an edge by using a certain threshold (i.e. 10% distance between the most distant nodes). We decided to use this threshold by assuming that this distance can be considered too close to and/or within personal space of a person for closed spaces. For NPC networks, setting this threshold to a known value can be a sufficient solution. However, creating different options for different usage examples can also be beneficial.

Additionally, we can suggest the inclusion of general social theories and community dynamics into our method. These concepts include small groups and the clique theory (Homans, 1950), small-world networks (Newman, 2018), preferential attachment (de Solla Price, 1965), community homogeneity/heterogeneity, triadic closure (Asikainen et al., 2020), information diffusion/spreading (Al-Taie et al., 2017) and two-step flow of communication hypothesis (Lazarsfeld et al., 1968), strong/weak ties theory (Granovetter, 1973), and reciprocity (Mauss, 2000). We tried to employ the concepts from same group of theories such as homophily and power-law degree distribution. Nonetheless, one or more concepts from these given ones can also be included to correctly identify NPC networks, since most of these theories are directly coming from the real-world social networks.

4.2 *Previous NPC Networks*

In this section, we tried to present earlier NPC social networks that are used or created in academic studies. We do this to identify the key points in earlier networks, the differences between these networks and the networks created with our network generator, and usage possibilities of our generator in NPC networks.

In their study, Lee et al. (2011) exploited social networks to compute automatic reputation. They built a social network by examining the authors' keyword referencing behaviors in a collection of text documents. They also address their social network by using community detection techniques. They discovered a number of unexpected situations, including that the network was scale-free and that it had negative assortativity. Although this work did not include an NPC but a social network of people, we decided to include it; since the next study we cover seems to be a continuation of the earlier work and also has a mutual author.

Brown et al. (2017) employs a similar method of automatic reputation calculation on various social networks, including networks that they generated procedurally and two NPC networks they extracted from the video games The Elder Scrolls 4: Oblivion and Fallout 4. In their article, they investigate a strategy for allowing NPC interactions to propagate reputation from an initial witness point of quest accomplishment to all other NPCs. Their tests indicate that information originating from densely populated areas spreads more rapidly than information produced in smaller quests from remote locations in the game. They shared their algorithm, the network nodes and the cities where those nodes belong to as open-source ¹.

Brown et al. (2020a) used same graphs for a different study, where they employed evolutionary graph compression and diffusion methods for city discovery in RPG games. In this study, they also shared the edge connections of the same graphs as open-source ².

Other than these studies, we could not find any other academic work that shares the NPC social networks they used. It is also evident from this fact that a usable NPC network generator is needed for these types of research problems.

4.3 A Minor Case Study: AnatoliA vs. Fallout 4 NPC Network

In this section, we try to evaluate our generator with respect to NPC social networks we can find as open-source. The researchers from Brown et al. (2020a) shared four different networks from four different computer games. These can be listed as: Fallout: New Vegas, Fallout 4, The Elder Scrolls IV: Oblivion and The Elder Scrolls V: Skyrim. We selected the NPC network from Fallout 4 to compare it with an AnatoliA generated network.

For this evaluation, we used particular assumptions. Fallout 4 NPC network has information about the nodes (i.e. the names of the nodes), the communities of the nodes (i.e. which fictional cities they belong to), and the edges between the nodes (i.e. the connections that are formed when NPCs interact with each other). Since there was no information about node attributes in Fallout 4 network, we could employ two different evaluations:

¹<https://github.com/nikitakraev/inno-thesis/tree/master>

²<http://www.cosc.brocku.ca/~houghten/gamegraphs.html>

- We can assume that there is no attribute information on both networks, and evaluate them accordingly, or,
- We can assume that we accept the player attributes (called as “perks” in the game) from the game as our NPC attributes, and generate the respective node attributes on both social networks in a predefined way.

Since AnatoliA can only generate network nodes with their attributes, we accept the second assumption and try to evaluate differences according to this assumption. For this assumption, we benefit from OpenAI’s GPT3.5, and its application, ChatGPT (Brown et al., 2020b). Using open-sourced dialogues from Fallout 4 Fandom Wiki ³, we create structured prompts to correctly extract NPC attribute affinity level results. These results should be regarded as ambiguous, as there is no way to verify if ChatGPT detects the numerical affinity values with an objective formulation. However, we assume that ChatGPT is among the few cutting-edge chatbots equipped with extensive natural language processing features, capable of subjectively identifying attribute values based on character dialogues.

To evaluate our network generator, we create AnatoliA networks with the same properties as Fallout 4 network. We also get the model properties of Fallout 4 network, and compare them with the model properties of our generated networks. For comparison, we used the absolute mean difference (AMD) between the property values, as we previously did in Section 3.3. The details of the model properties can also be found in the same section. We make this evaluation by changing our proximity threshold to various values, which is used in edge generation phase of our method. We try to understand which proximity value fits better to our needs compared to an actual NPC network.

We choose to compare different proximity values with respect to a reference NPC network, because as we discussed in Section 3.2.3, there is no way to correctly determine “the close proximity”, since it changes from context to context. By using Fallout 4 network, we can check which value corresponds to a close value for proximity between two nodes to form an edge in a game NPC network. To do that, we get the general properties of Fallout 4 network, generate 50 AnatoliA networks for different

³Shared under Creative Commons Attribution-Share Alike License 3.0 (Unported) (CC BY-SA), https://fallout.fandom.com/wiki/Fallout_4_characters

proximity threshold values, and compare the AMD between our mean model property values with values of Fallout 4 NPC network properties. We collected the results of this evaluation in Table 16.

Table 16: Model properties comparison for AnatoliA using Fallout 4 NPC network. Measurements come from 50 simulations performed with the same graph properties for each graph as ($N = 152, A = 7, C = 8$). Subscript values describe the edge generation proximity threshold, less than 0.1, 0.2, 0.3, 0.4 and 0.5, in order.

Fallout Network	AMD Over	A6A _{<0.1}	A6A _{<0.2}	A6A _{<0.3}	A6A _{<0.4}	A6A _{<0.5}
7.50e ⁻⁰²	<i>Clustering Coefficient</i>	5.91e⁻⁰²	1.05e ⁻⁰¹	0.119	0.128	0.137
2.829	<i>Degree</i>	0.971	0.319	0.136	0.041	0.025
4.59e ⁻⁰²	<i>Betweenness Centrality</i>	4.26e ⁻⁰²	4.16e ⁻⁰²	4.12e ⁻⁰²	4.08e⁻⁰²	4.09e ⁻⁰²
0.129	<i>Closeness Centrality</i>	0.094	0.055	0.032	0.020	0.017
215	<i>Edge Count</i>	73.82	24.22	10.36	3.08	1.92
1	<i># Connected Components</i>	55	48	43	41	41
17	<i>Diameter</i>	4.5	7.1	7.7	7.6	8.1
7.889	<i>Average SPL</i>	3.393	4.305	4.566	4.607	4.698

From the results, it is possible to infer that setting the proximity threshold to 0.5 is the best option, as this value is very similar to the property values coming from the Fallout 4 network. This value produces the most similar results for four different properties and is a strong competitor for another property. Another threshold value is 0.1, which produces results that are very similar to the Fallout 4 network in three different properties of the model. This particular value was used to evaluate our generator in Section 3.3. Therefore, it can be argued that using this value as our threshold in previous evaluations was a reasonable choice.

We also compared the model properties of our generator with previous generators, GenCAT (Maekawa et al., 2021) and X-Mark (Citraro and Rossetti, 2021), by using our two successful proximity thresholds, 0.1 and 0.5. The evaluation results can be found in Table 17. These results show that AnatoliA has promising outcomes in most of the properties, and it has more similar values to the Fallout 4 network in most of the given graph properties. The betweenness centrality value is also similar, and the differences to other generators are rather small. Since our method generates disconnected graphs, the only unsuccessful result comes from the number of connected components property.

There are limitations in this evaluation. The first limitation is using ChatGPT to generate attributes in the Fallout 4 network. While we explained our reasons for this

Table 17: Model properties evaluation for AnatoliA with earlier generators using Fallout 4 NPC network. Measurements are the mean values come from 50 simulations performed with the same graph properties for each graph as ($N = 152, A = 7, C = 8$). Subscript values describe the edge generation proximity threshold, less than 0.1 and 0.5, in order.

Fallout Network	AMD Over	A6A _{<0.1}	A6A _{<0.5}	GenCAT	X-Mark
7.50e ⁻⁰²	<i>Clustering Coefficient</i>	0.059	0.137	0.646	0.092
2.829	<i>Degree</i>	0.971	0.025	12.201	8.399
4.59e ⁻⁰²	<i>Betweenness Centrality</i>	<u>0.043</u>	0.166	<u>0.040</u>	0.037
0.129	<i>Closeness Centrality</i>	0.094	0.017	0.395	0.304
215	<i>Edge Count</i>	73.82	1.92	927.26	638.38
1	<i># Connected Components</i>	55	41	0	0
17	<i>Diameter</i>	4.5	8.1	13.8	13
7.889	<i>Average SPL</i>	3.393	4.698	5.962	5.562

in previous paragraphs, it is clearly a significant limitation.

The second limitation pertains to the lack of conceptual node positions in the Fallout 4 network. We examined the effects of different proximity values, but the NPC network data does not include any positions for NPCs in the network. Only additional information available from the network data is the community information to which each NPC belongs. We have decided to omit this viewpoint entirely to avoid introducing another black-box method for generating conceptual positions, similar to what was done with ChatGPT during attribute generation.

In conclusion, these findings can assist us in identifying a more suitable proximity threshold value. However, it is imperative to conduct further evaluations and assessments for confirmation. Perhaps we can explore alternative NPC networks that are appropriate for comparison with the networks generated by AnatoliA. We can also conduct a user study, to find out whether our generator is suitable or not for real-life scenarios of NPC network creation in games. Additionally, we can compare the real communities given by Fallout 4 network with communities detected by attribute-aware community detection methods in our generated networks to examine whether our attribute generation process is appropriate in such cases.

4.4 *Towards a General NPC Personality Model*

The primary goal of this thesis is to improve the realism of NPCs in computer games. To that end, we developed a procedural approach for creating NPC social networks. The resulting graphs of our approach can serve as the foundation for NPC communities. For individual NPCs, we try to lay out the characteristics of a general NPC personality model in this section. Because we want to improve the NPC realism, we assume that the model we propose must be entirely similar to human beings in terms of personal characteristics. However, it is possible to adjust this model to different living creatures that can be positioned as NPCs in games such as supernatural beings, animals or plants.

The first component we want to cover is the *personality engine*. Every NPC has a character, a distinct set of features which defines its inner world of ideas, feelings and thoughts. This engine must consist certain sub-parts for certain character traits. This list of sub-parts can include personal details –e.g., age, gender, nation, affiliation, conceptual location, etc.–, ancestral information –e.g., parents, belief systems, family tree, etc.–, general persona –e.g., optimist, pessimist, creative, etc.–, current mood/emotion –e.g., happy, sad, confused, terrified, etc.–, body features –e.g., tall, short, fat, skinny, handicapped, etc.–, habits, likes/dislikes, and hobbies/phobias.

The second component is the *action engine*. This engine includes the actions and activities that an NPC can do. A limited list of possible actions for a human-like NPC may include walking, running, moving, looking, sitting down and standing up, turning, talking and interacting (with someone), grabbing (something), hitting (to something or to someone), holding (tools), and so on. The action engine can use NPC decision-making methods to perform these actions and interactions, many of which are analyzed in Chapter 2.

The *memory engine* can be proposed as the third component of this model. Humans have both the short-term and long-term memories. Short-term memory can help the NPCs with their consecutive actions and their short-term goals. Long-term memory, on the other hand, can help them to define, form and change their relationships and their long-term objectives. By using this engine, the NPCs can behave and act with prior knowledge on both other NPCs and their environments.

The fourth component can be counted as the *learning engine*. This component must use machine learning (ML) techniques such as reinforcement learning for its benefit. By using ML approaches, NPCs can learn from their mistakes. It must also be responsible for the creativity of the NPCs. Creation of ideas, tools and artifacts must be done with the help of this engine.

The *communication engine* is the fifth component of our model. The NPCs can interact with each other by using a predefined language, body signs, gestures and mimics. This engine provides the dynamic relationships between NPCs.

The final component of our model is the *health engine*, which is responsible for addressing immediate health concerns such as illness, limb loss, loss of eyesight, loss of hearing ability, and other medical issues. Additionally, the health engine is used for life partner choices, transferred genes –i.e., the genetic information inherited from ancestors–, and reproduction –i.e., the genetic information transmitted to the descendants–.

There are existing studies in literature for components of our model, some of which we have already covered in the thesis. For instance, Georgeson and Child (2016) created an NPC personality engine called Extreme AI. In this engine, you may also define some possible actions of NPCs, in addition to its character traits. Perrie and Li (2014) created a social gossiping network as an example of a communication engine. Studies from Popescu et al. (2013) and Hooley et al. (2004) create emotion engines such as GAMYGDALA and EmoBot for in-game NPCs. We did not find any studies concerning some of our model components such as memory engine or health engine.

From our perspective, the various components of a game NPC must work in harmony. An NPC's personality, emotions, and current mood must respond to external stimuli. Additionally, the actions of an NPC must have consequences. If an NPC acquires useful knowledge, it must store this information in its long-term memory. An NPC's learning should be ongoing and cease only upon its demise. Before an NPC dies, it must form meaningful relationships with other NPCs and communicate at varying levels depending on the strength of the relationship. If the NPC's personality and memory encode a desire to reproduce, it should exhibit this behavior. In summary, a general NPC model, like the one we propose, is necessary for creating believable and realistic NPCs in video games.

To achieve this goal, in this thesis, we created the nodes of our network generator with attributes, positioning them as NPC personality traits such as likes/dislikes. We generated relationships between these nodes by using these traits as one of the essential decision criteria. We used the conceptual positions of NPCs as the other prominent decision criteria for forming edges between the nodes. In the next section, we give our ultimate model for our purpose, the adaptive NPC community model.

4.5 *The Ultimate Model: Adaptive NPC Community*

Our ultimate model comprises several aspects. Firstly, it includes our general NPC personality model designed for nodes within the NPC social network. Network edges are created based on interactions between NPC nodes. Certain node attributes are defined as private, while others are defined as public. NPCs can decide to form new relationships or terminate existing ones based on their interactions and the similarity of nodes.

Moreover, the communities in NPC social network must adhere to the rules of social dynamics and network theory of humans (Liu et al., 2017). In both Section 3.2 and Section 4.1, we discuss some of these social rules, including homophily, community homogeneity, small-world networks and power-law degree distribution.

Information flow/diffusion is a crucial concept in social networks (Wu et al., 2004). Employing directed and weighted edges in our network generator may create new possibilities. The NPCs can decide to share their public traits and information with a community, and decide to share their private traits and information with a different community. Sensitive information can be shared with a broader community if individuals possess more talkative or indiscreet personalities.

Other relevant concepts to mention are the theory of evolution and game theory. The term “survival of the fittest” impacts both individual NPCs and their communities. NPCs can have mutual or competing goals. To achieve their goals, they can work cooperatively as a community (Brede, 2011), or NPC communities can expel certain NPCs if they fail to follow the rules of their game. In these scenarios, the theory of evolution and game theory dictate that unsuccessful NPCs will be eliminated, while successful ones may be promoted to leadership positions in order to form their own

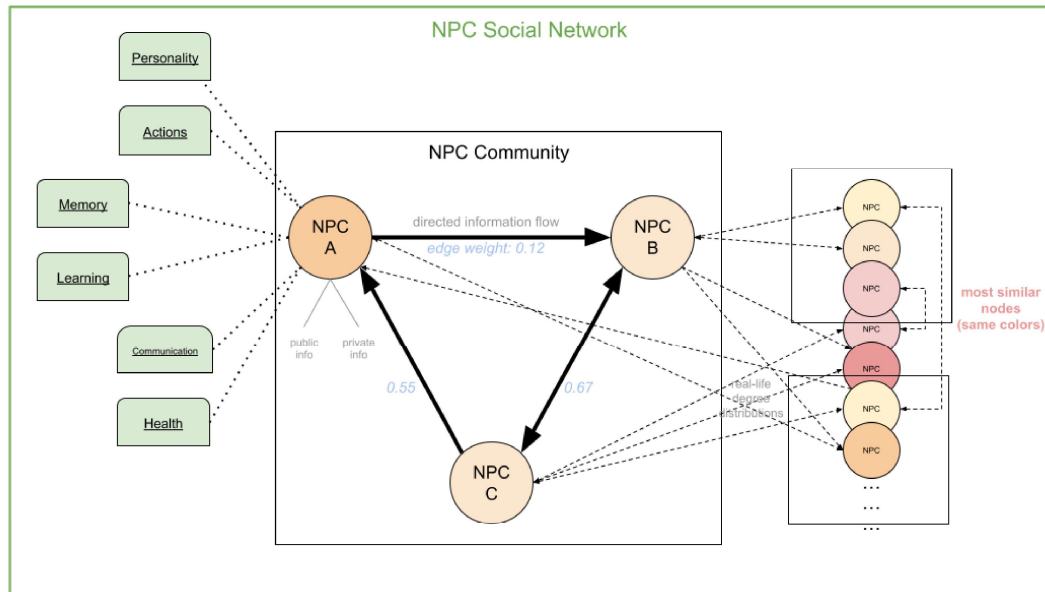


Figure 13: Our Adaptive NPC Community Model.

communities.

We have created a model schema to gather all the concepts mentioned in previous sections regarding the adaptive NPC community model. It can be seen from Figure 13.

In summary, we have developed a social network generator that follows several principles discussed in this chapter, and this generator may serve as an origin point for our ultimate NPC community model.

CHAPTER 5 : CONCLUSION & FUTURE DIRECTIONS

The goal of this thesis was to provide a framework for N2N and N2P coordination, interaction, and communication for realistic and adaptive gameplay in video games. This study has significantly advanced the field of NPC modeling and improved the overall player experience by conducting a thorough literature review of previous NPC decision-making techniques, creating a social network generator algorithm with node attributes and communities, and integrating these components into game environments.

The significance of the proposed framework was established in the first chapter, which also outlined the objectives of the study. It highlighted the shortcomings of current NPC interaction techniques and the need for novel approaches that take into account collaborative, human-like interactions. The theoretical foundations section laid the groundwork by talking about decision-making, the definition of NPCs in the context of games, the creation of social networks, and known methods for doing so. The following chapters had a strong foundation thanks to these principles.

The thorough literature review in Chapter 2 examined previous NPC decision-making processes and previous social network development methodologies. It identified shortcomings and limitations in the field by providing a critical analysis of the methods currently in use. This chapter added to the theoretical understanding of general NPC models by synthesizing the results of academic research, and provided insightful information on how to create more complex and adaptive NPC interactions.

The key element of the proposed framework, the social network generation algorithm, was discussed in Chapter 3. This algorithm incorporated concepts drawn from the dynamics of real-world social networks, including the use of specific node placement strategies, the concept of homophily, conceptual proximity, and the use of numerical node attributes to indicate node affinity. The concept and execution of the algorithm were described in detail, laying the groundwork for building social networks that capture the complexity of NPC interactions. The foundation for Chapter 4 was laid by the algorithm's ability to create diverse and realistic social networks with node attributes and communities.

In Chapter 4, we explore the integration of the generation algorithm into the gaming environment. We propose options to improve our algorithm for better use cases in NPC communities. Our algorithm is evaluated through a minor test case, featuring an NPC

network. We demonstrate the capabilities of our algorithm with respect to this real network. The key concepts of our general NPC model and adaptive NPC community model are shown as a result.

We make comprehensive efforts to address our research questions. A thorough literature review preceded the development of a survey article focusing on NPC decision-making in computer games. The survey provides valuable insights into how these methodologies are applied in industry games and presents tabulated results that detail the combination of methods used in literature. Additionally, a taxonomy table (Uludağlı and Oğuz, 2023) was developed, outlining the decision-making methods used for different game genres and usage environments.

The significance of AnatoliA, our social network generator with node attributes and communities, cannot be overstated as it contributes greatly to the establishment of dynamic NPC communities. Rigorous evaluation against existing generators and real-world data revealed promising outcomes, demonstrating significant efficacy across critical metrics. Correspondingly, our findings have been submitted to a prominent journal.

The endeavor to identify real NPC communities utilized in computer games culminated in the discovery of four different NPC networks, one of which served as the basis for evaluating our methodology through particular criteria. These networks including the one we used did not have node attributes, which led us to create them. This observation highlights the scarcity of available NPC networks and the need for a dedicated NPC network generator that can be used in similar situations.

As a last effort, we propose a general NPC personality model and an adaptive NPC community model to use them in computer games. We could not fully evaluate the models or their underlying principles. However, the lack of similar models in existing literature further emphasizes the significance of delving deeply into this paradigm for future investigations.

We believe that the results of this study have important implications for video game design. The proposed system promotes adaptive gameplay by allowing NPCs to interact and coordinate in a human-like manner. By generating complex social networks that accurately represent the intricacies of real-world social dynamics, “AnatoliA” is used to enrich the game experience. This allows for emergent behaviors,

context-sensitive decision-making, and varied interactions between NPCs and players.

Moreover, our study contributes to NPC modeling by bridging the gap between academic research and practical implementation. We build on existing knowledge of decision-making models and incorporate social network dynamics. This paves the way for more sophisticated and immersive NPC interactions. Our goal is to create social networks that capture the dependencies, influences, and communication patterns among NPCs, resulting in more realistic behaviors.

The successful development of the proposed framework demonstrates its ability to produce engaging, compelling gameplay. Game designers can create NPCs that exhibit complex social interactions, adapt to changing game circumstances, and give players a sense of immersion by using the social network generator algorithm. The framework enables game developers to incorporate emergent narratives and gameplay scenarios in place of scripted behavior, increasing the depth and replayability of video games.

Although NPC behavior modeling has advanced as a result of this research, there are still a number of directions that could be explored in the future. First, it is possible to improve the scalability and effectiveness of network generation in larger game contexts by further refining and optimizing the social network generator algorithm. In addition, by combining machine learning methods with data-driven strategies, NPCs can become more adaptive and responsive, able to learn and change their behavior over time.

Furthermore, the proposed methodology can also be extended to examine how player interactions affect NPC behavior and social networks. The social network generation algorithm could become more dynamic and player-driven by incorporating player choices and behaviors. Additionally, exploring the possibility of NPCs coordinating and communicating between games could lead to the creation of richer, more immersive game universes.

To conclude, this dissertation has addressed the need for a framework for NPC-to-NPC and NPC-to-player interaction, communication, and coordination in computer games that is realistic and adaptive. The study has advanced NPC behavior modeling by conducting a literature review of previous NPC decision-making techniques, creating a social network generation system, and combining these elements. The proposed architecture makes it possible to create NPCs with interactions, adaptability,

and coordination similar to those of humans, improving the overall game experience. The conclusions of the research have applications for game design, providing new opportunities for scenarios that are both immersive and interesting to play. As the gaming industry continues to evolve, the proposed framework provides a stepping stone to more intelligent and dynamic NPCs that will enrich the future of interactive entertainment.



REFERENCES

- Abiyev, R. H., Günsel, I., Akkaya, N., Aytac, E., Çağman, A. and Abizada, S. (2016) *Robot Soccer Control Using Behaviour Trees and Fuzzy Logic*, *Procedia Computer Science*, Vol. 102(August), pp. 477–484.
- Aiello, W., Chung, F. and Lu, L. (2000) *A random graph model for massive graphs*, in ‘Proceedings of the thirty-second annual ACM symposium on Theory of computing’, pp. 171–180.
- Aiello, W., Chung, F. and Lu, L. (2001) *A random graph model for power law graphs*, *Experimental Mathematics*, Vol. 10(1), pp. 53–66.
- Al-Taie, M. Z., Kadry, S., Al-Taie, M. Z. and Kadry, S. (2017) *Information diffusion in social networks*, *Python for Graph and Network Analysis* pp. 165–184.
- Amaral, L. A. N., Scala, A., Barthelemy, M. and Stanley, H. E. (2000) *Classes of small-world networks*, *Proceedings of the national academy of sciences*, Vol. 97(21), pp. 11149–11152.
- Asikainen, A., Iñiguez, G., Ureña-Carrión, J., Kaski, K. and Kivelä, M. (2020) *Cumulative effects of triadic closure and homophily in social networks*, *Science Advances*, Vol. 6(19), p. eaax7310.
- Barabási, A.-L. and Albert, R. (1999) *Emergence of scaling in random networks*, *Science*, Vol. 286(5439), pp. 509–512.
- Belle, S., Gittens, C. and Graham, T. N. (2022) *A Framework for Creating Non-Player Characters That Make Psychologically-Driven Decisions*, in ‘2022 IEEE International Conference on Consumer Electronics (ICCE)’, IEEE, pp. 1–7.
- Bellingeri, M., Bevacqua, D., Sartori, F., Turchetto, M., Scotognella, F., Alfieri, R., Nguyen, N., Le, T., Nguyen, Q. and Cassi, D. (2023) *Considering weights in real social networks: A review*, *Frontiers in Physics*, Vol. 11, p. 242.
- Benyahia, O., Largeron, C., Jeudy, B. and Zaïane, O. R. (2016) *Dancer: Dynamic attributed network with community structure generator*, in ‘Joint European Conference on Machine Learning and Knowledge Discovery in Databases’, Springer, pp. 41–44.

Blondel, V. D., Guillaume, J.-L., Lambiotte, R. and Lefebvre, E. (2008) *Fast unfolding of communities in large networks*, Journal of statistical mechanics: theory and experiment, Vol. 2008(10), p. P10008.

Bonifati, A., Holubová, I., Prat-Pérez, A. and Sakr, S. (2020) *Graph generators: State of the art and open challenges*, ACM Computing Surveys (CSUR), Vol. 53(2), pp. 1–30.

Bourg, D. M. and Seemann, G. (2004) *AI for Game Developers*, O’Reilly Series, O’Reilly Media, Inc.

Brandes, U. (2001) *A faster algorithm for betweenness centrality*, Journal of mathematical sociology, Vol. 25(2), pp. 163–177.

Brede, M. (2011) *Playing against the fittest: A simple strategy that promotes the emergence of cooperation*, Europhysics Letters, Vol. 94(3), p. 30003.

Bressan, S., Cuzzocrea, A., Karras, P., Lu, X. and Nobari, S. H. (2013) *An effective and efficient parallel approach for random graph generation over GPUs*, Journal of Parallel and Distributed Computing, Vol. 73(3), pp. 303–316.

Brown, J. A., Ashlock, D., Houghten, S. and Romualdo, A. (2020a) *Evolutionary Graph Compression and Diffusion Methods for City Discovery in Role Playing Games*, in ‘2020 IEEE Congress on Evolutionary Computation (CEC)’, IEEE, pp. 1–8.

Brown, J., Lee, J. and Kraev, N. (2017) *Reputation systems for non-player character interactions based on player actions*, in ‘Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment’, Vol. 13, pp. 151–157.

Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D. M., Wu, J., Winter, C., Hesse, C., Chen, M., Sigler, E., Litwin, M., Gray, S., Chess, B., Clark, J., Berner, C., McCandlish, S., Radford, A., Sutskever, I. and Amodei, D. (2020b) *Language Models are Few-Shot Learners*.

Caliński, T. and Harabasz, J. (1974) *A dendrite method for cluster analysis*, Communications in Statistics-theory and Methods, Vol. 3(1), pp. 1–27.

- Cavazza, M. (2000) *AI in computer games: Survey and perspectives*, Virtual Reality, Vol. 5(4), pp. 223–235.
- Chakrabarti, D. and Faloutsos, C. (2006) *Graph mining: Laws, generators, and algorithms*, ACM computing surveys (CSUR), Vol. 38(1), pp. 2–es.
- Charles, D. and McGlinchey, S. (2004) *The past, present and future of artificial neural networks in digital games*, in ‘Proceedings of the 5th international conference on computer games: artificial intelligence, design and education. The University of Wolverhampton’, pp. 163–169.
- Citraro, S. and Rossetti, G. (2019) *Eva: Attribute-aware network segmentation*, in ‘International Conference on Complex Networks and Their Applications’, Springer, pp. 141–151.
- Citraro, S. and Rossetti, G. (2021) *X-Mark: a benchmark for node-attributed community discovery algorithms*, Social Network Analysis and Mining, Vol. 11(1), pp. 1–14.
- Colledanchise, M. and Ögren, P. (2018) *Behavior trees in robotics and AI: An introduction*, CRC Press.
- Combe, D., Largeron, C., Géry, M. and Egyed-Zsigmond, E. (2015) *I-Louvain: An Attributed Graph Clustering Method*, in ‘International Symposium on Intelligent Data Analysis’, Springer, pp. 181–192.
- Davies, D. L. and Bouldin, D. W. (1979) *A Cluster Separation Measure*, IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. PAMI-1(2), pp. 224–227.
- de Solla Price, D. J. (1965) *Networks of Scientific Papers*, Science, Vol. 149(3683), pp. 510–515.
- Dell’Acqua, P. and Costantini, S. (2022) *Emotional Behavior Trees for Empathetic Human-Automation Interaction*, in ‘WOA 2022: 23rd Workshop From Objects to Agents’.
- Dey, R. and Child, C. (2013) *QL-BT: Enhancing behaviour tree design and implementation with Q-learning*, in ‘2013 IEEE Conference on Computational Intelligence in Games (CIG)’, IEEE, pp. 1–8.

Dias, J., Mascarenhas, S. and Paiva, A. (2014) *FAtiMA Modular: Towards an Agent Architecture with a Generic Appraisal Framework*, in T. Bosse, J. Broekens, J. Dias and J. van der Zwaan, eds, 'Emotion Modeling: Towards Pragmatic Computational Models of Affective Processes', Springer International Publishing, Cham, pp. 44–56.

Doluca, O. and Oğuz, K. (2021) *APAL: Adjacency Propagation Algorithm for overlapping community detection in biological networks*, Information Sciences, Vol. 579, pp. 574–590.

Easley, D. and Kleinberg, J. (2010) *Networks, crowds, and markets: Reasoning about a highly connected world*, Cambridge university press.

El-Nasr, M. S., Yen, J. and Ioerger, T. R. (2000) *FLAME — Fuzzy Logic Adaptive Model of Emotions*, Autonomous Agents and Multi-agent systems, Vol. 3(3), pp. 219–257.

Elhadi, H. and Agam, G. (2013) *Structure and attributes community detection: comparative analysis of composite, ensemble and selection methods*, in 'Proceedings of the 7th workshop on social network mining and analysis', pp. 1–7.

Erdős, P. and Rényi, A. (1959) *On the evolution of random graphs*, in 'The structure and dynamics of networks', Princeton University Press, pp. 38–82.

Fathoni, K., Hakkun, R. and Nurhadi, H. (2020) *Finite State Machines for Building Believable Non-Playable Character in the Game of Khalid ibn Al-Walid*, in 'Journal of Physics: Conference Series', Vol. 1577, IOP Publishing, p. 012018.

Fauzi, R., Hariadi, M., Nugroho, S. M. S. and Lubis, M. (2019) *Defense behavior of real time strategy games: comparison between HFSM and FSM*, Indonesia Journal of Electrical Engineering and Computer Science, Vol. 13(2), pp. 634–642.

Fletcher, P., Hoyle, H. and Patty, C. (1991) *Foundations of Discrete Mathematics*, PWS-KENT Publishing Company.

Florez-Puga, G., Gomez-Martin, M., Diaz-Agudo, B. and Gonzalez-Calero, P. (2008) *Dynamic Expansion of Behaviour Trees*, in 'Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment', Vol. 4, pp. 36–41.

Forgy, C. L. (1979), *On the efficient implementation of production systems.*, PhD thesis, Carnegie Mellon University.

Forgy, C. L. (1989) *Rete: A fast algorithm for the many pattern/many object pattern match problem*, in J. Mylopoulos and M. Brodie, eds, 'Readings in Artificial Intelligence and Databases', Morgan Kaufmann, San Francisco (CA), pp. 547–559.

Fortunato, S. (2010) *Community detection in graphs*, Physics reports, Vol. 486(3-5), pp. 75–174.

Fortunato, S. and Hric, D. (2016) *Community detection in networks: A user guide*, Physics reports, Vol. 659, pp. 1–44.

Fournier, A., Fussell, D. and Carpenter, L. (1982) *Computer rendering of stochastic models*, Communications of the ACM, Vol. 25(6), pp. 371–384.

Freeman, L. C. (1978) *Centrality in social networks conceptual clarification*, Social networks, Vol. 1(3), pp. 215–239.

Fu, D. and Houlette, R. (2002) *Putting AI in entertainment: An AI authoring tool for simulation and games*, IEEE Intelligent Systems, Vol. 17(4), pp. 81–84.

Fu, Y., Qin, L. and Yin, Q. (2016) *A Reinforcement Learning Behavior Tree Framework for Game AI*, in '2016 International Conference on Economics, Social Science, Arts, Education and Management Engineering', Atlantis Press, pp. 573–579.

Fujii, S., Nakashima, T. and Ishibuchi, H. (2008) *A study on constructing fuzzy systems for high-level decision making in a car racing game*, in '2008 IEEE Congress on Evolutionary Computation, CEC 2008', IEEE, pp. 3626–3633.

Georgeson, J. and Child, C. (2016) *NPCs as People, Too: The Extreme AI Personality Engine*, arXiv preprint arXiv:1609.04879 .

Girvan, M. and Newman, M. E. (2002) *Community structure in social and biological networks*, Proceedings of the national academy of sciences, Vol. 99(12), pp. 7821–7826.

Granovetter, M. S. (1973) *The strength of weak ties*, American journal of sociology, Vol. 78(6), pp. 1360–1380.

Grossman, J. W. and Ion, P. D. (1995) *On a portion of the well-known collaboration graph*, *Congressus Numerantium* pp. 129–132.

Guimaraes, M., Santos, P. and Jhala, A. (2017) *CiF-CK: An architecture for social NPCs in commercial games*, in ‘2017 IEEE Conference on Computational Intelligence and Games (CIG)’, IEEE, pp. 126–133.

Gupta, N. (2013) *Artificial neural network*, *Network and Complex Systems*, Vol. 3(1), pp. 24–28.

Holba, J. and Huber, G. (2021) *Open-world Enemy AI in Mafia III*, in ‘Game AI Pro – Online Edition 2021’, *Game AI Pro*, chapter 16.

Homans, G. C. (1950) *The Human Group*, Harcourt, Brace & World.

Hooley, T., Hunking, B., Henry, M. and Inoue, A. (2004) *Generation of Emotional Behavior for Non-Player Characters-Development of EmoBot for Quake II*, in ‘PROCEEDINGS OF THE NATIONAL CONFERENCE ON ARTIFICIAL INTELLIGENCE’, Menlo Park, CA; Cambridge, MA; London; AAAI Press; MIT Press; 1999, pp. 954–955.

Iovino, M., Scukins, E., Styruđ, J., Ögren, P. and Smith, C. (2022) *A survey of Behavior Trees in robotics and AI*, *Robotics and Autonomous Systems*, Vol. 154, p. 104096.

Isla, D. (2005) *GDC 2005 Proceeding: Handling Complexity in the Halo 2 AI*, in ‘Game Developers Conference 2005’.

URL: <https://www.gamedeveloper.com/programming/gdc-2005-proceeding-handling-complexity-in-the-i-halo-2-i-ai>

Javed, M. A., Younis, M. S., Latif, S., Qadir, J. and Baig, A. (2018) *Community detection in networks: A multidisciplinary review*, *Journal of Network and Computer Applications*, Vol. 108, pp. 87–111.

Jia, C., Li, Y., Carson, M. B., Wang, X. and Yu, J. (2017) *Node attribute-enhanced community detection in complex networks*, *Scientific reports*, Vol. 7(1), p. 2626.

Johansen, N. S., Kær, L. B., Stolberg, J. A. B., Tollund, R. G., Hyldig, N., Oktober, P. and Torralba, A. (2022) *Towards Believable Non-Player Characters with Domain-Independent Planning*, in ‘2022 Workshop on Scheduling and Planning Applications woRKshop’.

Johansson, A. and Dell’Acqua, P. (2012) *Comparing behavior trees and emotional behavior networks for NPCs*, in ‘2012 17th International Conference on Computer Games (CGAMES)’ , IEEE, pp. 253–260.

Johnson, D. and Wiles, J. (2001) *Computer games with intelligence*, in ‘10th IEEE International Conference on Fuzzy Systems.(Cat. No. 01CH37297)’ , Vol. 3, IEEE, pp. 1355–1358.

Jolly, K., Ravindran, K., Vijayakumar, R. and Kumar, R. S. (2007) *Intelligent decision making in multi-agent robot soccer system through compounded artificial neural networks*, Robotics and Autonomous Systems, Vol. 55(7), pp. 589–596.

Kopel, M. and Hajas, T. (2018) *Implementing AI for Non-player Characters in 3D Video Games*, in N. T. Nguyen, D. H. Hoang, T.-P. Hong, H. Pham and B. Trawiński, eds, ‘Intelligent Information and Database Systems’, Springer International Publishing, Cham, pp. 610–619.

Laird, J. E. (2001) *It knows what you’re going to do: Adding anticipation to a Quakebot*, in ‘Proceedings of the fifth international conference on Autonomous agents’, pp. 385–392.

Lancichinetti, A. and Fortunato, S. (2009a) *Benchmarks for testing community detection algorithms on directed and weighted graphs with overlapping communities*, Physical Review E, Vol. 80(1), p. 016118.

Lancichinetti, A. and Fortunato, S. (2009b) *Community detection algorithms: a comparative analysis*, Physical review E, Vol. 80(5), p. 056117.

Lancichinetti, A., Fortunato, S. and Radicchi, F. (2008) *Benchmark graphs for testing community detection algorithms*, Physical review E, Vol. 78(4), p. 046110.

Largerone, C., Mougél, P.-N., Rabbany, R. and Zaïane, O. R. (2015) *Generating attributed networks with communities*, PloS one, Vol. 10(4), p. e0122777.

Lazarsfeld, P. F., Berelson, B. and Gaudet, H. (1968) *The people's choice: How the voter makes up his mind in a presidential campaign*, Columbia University Press.

Lee, J., Duan, Y., Oh, J. C., Du, W., Blair, H., Wang, L. and Jin, X. (2011) *Automatic reputation computation through document analysis: A social network approach*, in '2011 International Conference on Advances in Social Networks Analysis and Mining', IEEE, pp. 559–560.

Leskovec, J., Chakrabarti, D., Kleinberg, J., Faloutsos, C. and Ghahramani, Z. (2010) *Kronecker graphs: an approach to modeling networks.*, Journal of Machine Learning Research, Vol. 11(2).

Lesser, E., Fontaine, M. and Slusher, J. (2000) *Knowledge and communities*, Routledge.

Lewis, K., Kaufman, J., Gonzalez, M., Wimmer, A. and Christakis, N. (2008) *Tastes, ties, and time: A new social network dataset using Facebook. com*, Social networks, Vol. 30(4), pp. 330–342.

Li, P., Yu, J., Liu, J., Zhou, D. and Cao, B. (2020) *Generating weighted social networks using multigraph*, Physica A: Statistical Mechanics and its Applications, Vol. 539, p. 122894.

Li, Y., Musilek, P. and Wyard-Scott, L. (2004) *Fuzzy logic in agent-based game design*, in 'IEEE Annual Meeting of the Fuzzy Information, 2004. Processing NAFIPS'04.', Vol. 2, IEEE, pp. 734–739.

Lie, C. S. K. and Istiono, W. (2022) *How To Make NPC Learn The Strategy In Fighting Games Using Adaptive AI?*, International Journal of Scientific and Technical Research in Engineering (IJSTRE), Vol. 7(4).

Lin, J., He, J. and Zhang, N. (2019) *Application of behavior tree in AI design of MOBA games*, in '2019 IEEE 2nd International Conference on Knowledge Innovation and Invention (ICKII)', IEEE, pp. 323–326.

Liu, W., Sidhu, A., Beacom, A. M. and Valente, T. W. (2017) *Social network theory*, The international encyclopedia of media effects pp. 1–12.

- Long, E. (2007) *Enhanced NPC Behaviour using Goal Oriented Action Planning*, Master's thesis, University of Abertay Dundee, Dundee, Scotland.
- Loyall, B. A. and Bates, J. (1991) *Have a reactive, adaptive architecture for agents*, in 'Citeseer'.
- Lueg, C. and Fisher, D. (2003) *From Usenet to CoWebs: interacting with social information spaces*, Springer Science & Business Media.
- Lusseau, D. (2003) *The emergent properties of a dolphin social network*, Proceedings of the Royal Society of London. Series B: Biological Sciences, Vol. 270(suppl_2), pp. S186–S188.
- MacKay, R. S., Johnson, S. and Sansom, B. (2020) *How directed is a directed network?*, Royal Society open science, Vol. 7(9), p. 201138.
- Madsen, C. A. B. C. W. and Adamatti, D. F. (2013) *Using Artificial Neural Networks in NPC Decision-Making Process*, International Journal of Computer and Information Technology, Vol. 2(6), pp. 1009–1013.
- Maekawa, S., Sasaki, Y., Fletcher, G. and Onizuka, M. (2021) *GenCAT: Generating Attributed Graphs with Controlled Relationships between Classes, Attributes, and Topology*, arXiv preprint arXiv:2109.04639 .
- Maekawa, S., Zhang, J., Fletcher, G. and Onizuka, M. (2019) *General generator for attributed graphs with community structure*, in 'proceeding of the ECML/PKDD Graph Embedding and Mining Workshop', pp. 1–5.
- Mascarenhas, S., Guimarães, M., Santos, P. A., Dias, J., Prada, R. and Paiva, A. (2021) *FAtiMA Toolkit—Toward an effective and accessible tool for the development of intelligent virtual agents and social robots*, arXiv preprint arXiv:2103.03020 .
- Mas'udi, N. A., Jonemaro, E. M. A., Akbar, M. A. and Afirianto, T. (2021) *Development of Non-Player Character for 3D Kart Racing Game Using Decision Tree*, Fountain of Informatics Journal, Vol. 6(2), pp. 51–60.
- Mateas, M. and Stern, A. (2002) *A behavior language for story-based believable agents*, IEEE Intelligent Systems, Vol. 17(4), pp. 39–47.

- Mauss, M. (2000) *The gift: The form and reason for exchange in archaic societies*, WW Norton & Company.
- McCoy, J., Treanor, M., Samuel, B., Tearse, B., Mateas, M. and Wardrip-Fruin, N. (2010) *Comme il Faut 2: A fully realized model for socially-oriented gameplay*, in ‘Proceedings of the Intelligent Narrative Technologies III Workshop’, INT3 ’10, Association for Computing Machinery, New York, NY, USA, pp. 1–8.
- McCoy, J., Treanor, M., Samuel, B., Wardrip-Fruin, N. and Mateas, M. (2011) *Comme il Faut: A System for Authoring Playable Social Models*, in ‘Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment’, Vol. 6.
- McPherson, M., Smith-Lovin, L. and Cook, J. M. (2001) *Birds of a feather: Homophily in social networks*, *Annual review of sociology*, Vol. 27(1), pp. 415–444.
- Meng, F. and Hyung, C. J. (2022) *Research on Multi-NPC Marine Game AI System based on Q-learning Algorithm*, in ‘2022 IEEE International Conference on Artificial Intelligence and Computer Applications (ICAICA)’, IEEE, pp. 648–652.
- Millington, I. (2019) *Artificial Intelligence for Games*, 3 edn, CRC Press.
- Miyake, Y., Shirakami, Y., Shimokawa, K., Namiki, K., Komatsu, T., Tatsuhiro, J., Prasertvithyakarn, P. and Yokoyama, T. (2019) *A Character Decision-Making System for FINAL FANTASY XV by Combining Behavior Trees and State Machines*, in ‘Game AI Pro 360: Guide to Architecture’, CRC Press, p. 339.
- Murase, Y., Török, J., Jo, H.-H., Kaski, K. and Kertész, J. (2014) *Multilayer weighted social network model*, *Physical Review E*, Vol. 90(5), p. 052810.
- Nareyek, A. (2000) *Intelligent agents for computer games*, in ‘International Conference on Computers and Games’, Springer, pp. 414–422.
- Nettleton, D. F. (2016) *A synthetic data generator for online social network graphs*, *Social Network Analysis and Mining*, Vol. 6(1), pp. 1–33.
- Neufeld, X., Mostaghim, S., Sancho-Pradel, D. L. and Brand, S. (2019) *Building a Planner: A Survey of Planning Systems Used in Commercial Video Games*, *IEEE Transactions on Games*, Vol. 11(2), pp. 91–108.

- Newman, M. (2018) *Networks An Introduction*, Oxford university press.
- Newman, M. E. (2001) *The structure of scientific collaboration networks*, Proceedings of the national academy of sciences, Vol. 98(2), pp. 404–409.
- Niewiadomski, A. and Renkas, K. (2014) *Hierarchical fuzzy logic systems and controlling vehicles in computer games*, Journal of Applied Computer Science, Vol. 22(1), pp. 201–212.
- Nobari, S., Lu, X., Karras, P. and Bressan, S. (2011) *Fast random graph generation*, in ‘Proceedings of the 14th international conference on extending database technology’, pp. 331–342.
- Novák, V., Perfilieva, I. and Mockor, J. (1999) *Mathematical Principles of Fuzzy Logic*, Vol. 517, Springer Science & Business Media.
- Ohson, K. and Onisawa, T. (2008) *Friendly partner system of poker game with facial expressions*, in ‘2008 IEEE Symposium On Computational Intelligence and Games’, IEEE, pp. 95–102.
- Orkin, J. (2003) *Applying Goal-Oriented Action Planning to Games*, in S. Rabin, ed., ‘AI Game Programming Wisdom, Vol. 2’, Charles River Media, Inc., USA.
- Orkin, J. (2006) *Three states and a plan: the AI of FEAR*, in ‘Game Developers Conference 2006’, CMP Game Group, San Jose, California, p. 4.
- Orman, G. K. and Labatut, V. (2009) *A comparison of community detection algorithms on artificial networks*, in ‘International conference on discovery science’, Springer, pp. 242–256.
- O’Brien, L. (1996) *Fuzzy Logic in Games*, Game Developer Magazine, Vol. 3(2), pp. 52–55.
- Palla, G., Derényi, I., Farkas, I. and Vicsek, T. (2005) *Uncovering the overlapping community structure of complex networks in nature and society*, Nature, Vol. 435(7043), pp. 814–818.
- Panwar, H. (2022) *The NPC AI of The Last of Us: A case study*, arXiv preprint .

Partlan, N., Soto, L., Howe, J., Shrivastava, S., El-Nasr, M. S. and Marsella, S. (2022) *Evolving Behavior: Towards Co-Creative Evolution of Behavior Trees for Game NPCs*, in ‘Proceedings of the 17th International Conference on the Foundations of Digital Games’, FDG ’22, Association for Computing Machinery, New York, NY, USA.

Perez, D., Nicolau, M., O’Neill, M. and Brabazon, A. (2011) *Evolving Behaviour Trees for the Mario AI Competition Using Grammatical Evolution*, in ‘Applications of Evolutionary Computation’, Springer, Berlin, Heidelberg, pp. 123–132.

Perrie, J. and Li, L. (2014) *Building a Dynamic Social Community with Non Playable Characters*, IEICE TRANSACTIONS on Information and Systems, Vol. 97(8), pp. 1965–1973.

Pfeiffer III, J. J., Moreno, S., La Fond, T., Neville, J. and Gallagher, B. (2014) *Attributed graph models: Modeling network structure with correlated attributes*, in ‘Proceedings of the 23rd international conference on World wide web’, pp. 831–842.

Pirovano, M. (2012) *The use of fuzzy logic for artificial intelligence in games*, University of Milano, Milano .

Pittman, D. L. (2007) *Practical Development of Goal-Oriented Action Planning AI*, Master’s thesis, Southern Methodist University, Dallas, TX, USA.

Popescu, A., Broekens, J. and Van Someren, M. (2013) *GAMYGDALA: An Emotion Engine for Games*, IEEE Transactions on Affective Computing, Vol. 5(1), pp. 32–44.

Prat-Pérez, A., Guisado-Gámez, J., Salas, X. F., Koupy, P., Depner, S. and Bartolini, D. B. (2017) *Towards a property graph generator for benchmarking*, in ‘Proceedings of the fifth international workshop on graph data-management experiences & systems’, pp. 1–6.

Pyjas, G. M., Weinel, J. and Broadhead, M. (2022) *Storytelling and VR: Inducing emotions through AI characters*, Proceedings of EVA London 2022 pp. 198–204.

Quadir, A. M. and Khder, M. A. (2022) *Exploring the Potential of AI-Driven Opponents in Video Games*, in ‘2022 ASU International Conference in Emerging

Technologies for Sustainability and Intelligent Systems (ICETISIS)', IEEE, pp. 464–469.

Quinlan, J. R. (1986) *Induction of decision trees*, Machine learning, Vol. 1(1), pp. 81–106.

Rodrigues, S., Rayat, H. K., Kurichithanam, R. M. and Rukhande, S. (2021) *Shriek: A Role Playing Game Using Unreal Engine 4 and Behaviour Trees*, in '2021 4th Biennial International Conference on Nascent Technologies in Engineering (ICNTE)', IEEE, pp. 1–6.

Rosen, K. H. (2007) *Discrete mathematics and its applications*, The McGraw Hill Companies.

Rossetti, G. and Cazabet, R. (2018) *Community discovery in dynamic networks: a survey*, ACM Computing Surveys (CSUR), Vol. 51(2), pp. 1–37.

Rousseeuw, P. J. (1987) *Silhouettes: a graphical aid to the interpretation and validation of cluster analysis*, Journal of computational and applied mathematics, Vol. 20, pp. 53–65.

Sagredo-Olivenza, I., Gómez-Martín, P. P., Gómez-Martín, M. A. and González-Calero, P. A. (2017) *Trained Behavior Trees: Programming by Demonstration to Support AI Game Designers*, IEEE Transactions on Games, Vol. 11(1), pp. 5–14.

Sailer, L. D. and Gaulin, S. J. (1984) *Proximity, sociality, and observation: the definition of social groups*, American Anthropologist pp. 91–98.

Salganik, M. J. and Heckathorn, D. D. (2004) *Sampling and estimation in hidden populations using respondent-driven sampling*, Sociological methodology, Vol. 34(1), pp. 193–240.

Schwab, P. and Hlavacs, H. (2015) *Capturing the essence: Towards the automated generation of transparent behavior models*, in 'Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment', Vol. 11.

Schweimer, C., Gfrerer, C., Lugstein, F., Pape, D., Velimsky, J. A., Elsässer, R. and Geiger, B. C. (2022) *Generating simple directed social network graphs for information spreading*, in 'Proceedings of the ACM Web Conference 2022', pp. 1475–1485.

- Sekhvat, Y. A. (2017) *Behavior trees for computer games*, International Journal on Artificial Intelligence Tools, Vol. 26(02), p. 1730001.
- Shaout, A., King, B. W. and Reisner, L. A. (2006) *Real-Time Game Design of Pac-Man Using Fuzzy Logic*, Int. Arab J. Inf. Technol., Vol. 3(4), pp. 315–325.
- Shu, F. and Chaudhari, N. S. (2008) *A chaotic behavior decision algorithm based on self-generating neural network for computer games*, in ‘2008 3rd IEEE Conference on Industrial Electronics and Applications’, IEEE, pp. 1912–1915.
- Sielicki, M., Daszuta, M. and Szajerman, D. (2018) *Adaptation and application of Goal Oriented Action Planning in Unreal Engine*, Computer Game Innovations p. 103.
- Simonov, A., Zagarskikh, A. and Fedorov, V. (2019) *Applying Behavior characteristics to decision-making process to create believable game AI*, Procedia Computer Science, Vol. 156, pp. 404–413. presented at 8th International Young Scientists Conference on Computational Science, YSC2019, 24-28 June 2019, Heraklion, Greece.
- Sindhu, R. M., Annabel, L. S. P. and Monisha, G. (2022) *Development of a 2D Game using Artificial Intelligence in Unity*, in ‘2022 6th International Conference on Trends in Electronics and Informatics (ICOEI)’, IEEE, pp. 1031–1037.
- Sloan, C. (2015), *Drive-Based Utility-Maximizing Computer Game Non-Player Characters*, PhD thesis, Technological University Dublin, Dublin, Ireland.
- Sloan, C., Mac Namee, B. and Kelleher, J. D. (2011) *Utility-Directed Goal-Oriented Action Planning: A Utility-Based Control System for Computer Game Agents*, MartinMcGinnity Intelligent Systems Research Centre, University of Ulster .
- Smith, M. A. (1999) *Invisible crowds in cyberspace*, in ‘Communities in Cyberspace’, Routledge London, pp. 195–218.
- Soylucicek, A. E., Bostanci, E. and Safak, A. B. (2017) *A Fuzzy Logic Based Attack Strategy Design for Enemy Drones in Meteor Escape Game*, International Journal of Computer Theory and Engineering, Vol. 9(3), pp. 167–171.
- Stopczynski, A., Pentland, A. and Lehmann, S. (2018) *How physical proximity shapes complex social networks*, Scientific reports, Vol. 8(1), pp. 1–10.

- Studiawan, R., Hariadi, M. and Sumpeno, S. (2018) *Tactical Planning in Space Game using Goal-Oriented Action Planning*, JAREE (Journal on Advanced Research in Electrical Engineering), Vol. 2(1).
- Suyikno, D. A. and Setiawan, A. (2019) *Feasible NPC Hiding Behaviour using Goal Oriented Action Planning in case of Hide-and-Seek 3D Game Simulation*, in '2019 Fourth International Conference on Informatics and Computing (ICIC)', IEEE, pp. 1–6.
- Sweetser, P. and Wiles, J. (2002) *Current AI in games: A review*, Australian Journal of Intelligent Information Processing Systems, Vol. 8(1), pp. 24–42.
- Syahputra, M., Arippa, A., Rahmat, R. and Andayani, U. (2019) *Historical theme game using finite state machine for actor behaviour*, in 'Journal of Physics: Conference Series', Vol. 1235, IOP Publishing, p. 012122.
- Uludağlı, M. Ç. and Oğuz, K. (2023) *Non-player character decision-making in computer games*, Artificial Intelligence Review, Vol. 56(12), pp. 14159–14191.
- Van Waveren, J. M. P. (2001) *The Quake III Arena Bot*, Master's thesis, Delft University of Technology, Delft, Netherlands.
- Wang, B., Wang, C. and Feng, H. (2021) *FastSNG: The Fastest Social Network Dataset Generator*, in 'Companion Proceedings of the Web Conference 2021', pp. 680–684.
- Watts, D. J. and Strogatz, S. H. (1998) *Collective dynamics of 'small-world' networks*, Nature, Vol. 393(6684), pp. 440–442.
- Wexler, J. (2002) *Artificial Intelligence in Games*, Rochester: University of Rochester .
- Widhiyasana, Y., Harika, M., Hakim, F. F. N., Diani, F., Komariah, K. S. and Ramdania, D. R. (2022) *Genetic Algorithm for Artificial Neural Networks in Real-Time Strategy Games*, JOIV: International Journal on Informatics Visualization, Vol. 6(2), pp. 298–305.
- Wu, F., Huberman, B. A., Adamic, L. A. and Tyler, J. R. (2004) *Information flow in social groups*, Physica A: Statistical Mechanics and its Applications, Vol. 337(1-2), pp. 327–335.

Xiang, S., Wen, D., Cheng, D., Zhang, Y., Qin, L., Qian, Z. and Lin, X. (2021) *General graph generators: experiments, analyses, and improvements*, The VLDB Journal pp. 1–29.

Xie, J., Kelley, S. and Szymanski, B. K. (2013) *Overlapping community detection in networks: The state-of-the-art and comparative study*, Acm computing surveys (csur), Vol. 45(4), pp. 1–35.

Yang, J. and Leskovec, J. (2015) *Defining and evaluating network communities based on ground-truth*, Knowledge and Information Systems, Vol. 42(1), pp. 181–213.

Yannakakis, G. N. and Togelius, J. (2018) *Artificial Intelligence and Games*, Springer. <http://gameaibook.org>.

Yue, B. and de Byl, P. (2006) *The State of the Art in Game AI Standardisation*, in ‘Proceedings of the 2006 International Conference on Game Research and Development’, CyberGames ’06, Murdoch University, Murdoch, AUS, pp. 41–46.

Zhu, M. and Feng, L. (2022) *Design and Implementation of NPC AI based on Genetic Algorithm and BP Neural Network*, in ‘Proceedings of the 14th International Conference on Computer Modeling and Simulation’, ICCMS ’22, Association for Computing Machinery, New York, NY, USA, pp. 168–173.

Zijie, W., Tongyu, W. and Hang, G. (2021) *A Survey: Development and Application of Behavior Trees*, in Q. Liang, W. Wang, X. Liu, Z. Na, X. Li and B. Zhang, eds, ‘Communications, Signal Processing, and Systems’, Springer Singapore, Singapore, pp. 1581–1589.

CURRICULUM VITAE

Muhtar Çağkan Uludağlı received his BSc degree in Computer Engineering from the Faculty of Engineering, İzmir Institute of Technology in 2012 and his MSc degree in Game Technologies from the Informatics Institute, Middle East Technical University in 2017. He worked as a software developer at METU Computer Center for over three years during his MSc education. He's been studying at Computer Engineering PhD Program at Izmir University of Economics since July 2017 and he's been working as a research assistant at the same department since April 2018.

His research interests cover the topics such as graph/data science, player-computer interaction, artificial intelligence for games, psychology in game design and virtual/augmented/mixed reality.

List of Publications

1. Muhtar Çağkan Uludağlı, Kaya Oğuz. Generating Complex Social Networks with Node Attributes. *Information Sciences*. 2023, Elsevier, 30 pages. (submitted at 20.07.2023)
2. Uludağlı, M.Ç., Oğuz, K. Non-player character decision-making in computer games. *Artif Intell Rev* (2023). Springer, Volume 56, Number 12, pg. 1-33. <https://doi.org/10.1007/s10462-023-10491-7>
3. Tekgün, E, Uludağlı, MÇ, Akcan, H, Erdeniz, B. Virtual body anthropomorphism increases drift in self-location: Further support for the humanoid shape rule. *Comput Anim Virtual Worlds*. 2022; 33(2):e2038. <https://doi.org/10.1002/cav.2038>
4. ULUDAĞLI, MUHTAR ÇAĞKAN and ACARTÜRK, CENGİZ (2018) "User interaction in hands-free gaming: a comparative study of gaze-voice and touchscreen interface control," *Turkish Journal of Electrical Engineering and Computer Sciences*: Vol. 26: No. 4, Article 24. <https://doi.org/10.3906/elk-1710-128>