West Visayas State University
COLLEGE OF INFORMATION AND COMMUNICATIONS TECHNOLOGY
La Paz, Iloilo City

FENCING AI; NEUROEVOLUTION OF AUGMENTING TOPOLOGIES (NEAT) TOWARDS COMPETITION AND SURVIVAL OF THE FITTEST IN A FENCING SIMULATION

An Undergraduate Thesis

Presented to the Faculty of the

College of Information and Communications Technology

West Visayas State University

La Paz, Iloilo City

In Partial Fulfillment

of the Requirements for the Degree

Bachelor of Science in Computer Science

by

Bryan Kent S. Abesamis

Timothy A. Diosaban

Alyssa Danielle R. Magallanes

Neil Ryan S. Sustiguer

June 2022

FENCING AI; NEUROEVOLUTION OF AUGMENTING TOPOLOGIES (NEAT) TOWARDS COMPETITION AND SURVIVAL OF THE FITTEST IN A FENCING SIMULATION

An Undergraduate Thesis

Presented to the Faculty of the

College of Information and Communications Technology

West Visayas State University

La Paz, Iloilo City

In Partial Fulfillment

of the Requirements for the Degree

Bachelor of Science in Computer Science

by

Bryan Kent S. Abesamis

Timothy A. Diosaban

Alyssa Danielle R. Magallanes

Neil Ryan S. Sustiguer

June 2022

West Visayas State University
COLLEGE OF INFORMATION AND COMMUNICATIONS TECHNOLOGY
Lapaz, Iloilo City

Approval Sheet

FENCING AI; NEUROEVOLUTION OF AUGMENTING TOPOLOGIES (NEAT) TOWARDS COMPETITION AND SURVIVAL OF THE FITTEST IN A FENCING SIMULATION

Bachelor of Science in Computer Science

by

Bryan Kent S. Abesamis

Timothy A. Diosaban

Alyssa Danielle R. Magallanes

Neil Ryan S. Sustiguer

Approved:

DR. BOBBY D. GERARDO
Adviser

DR. MA. LUCHE P. SABAYLE
Chair, Computer Science

DR. MA.BETH S. CONCEPCION
Dean, CICT

June 2022

Acknowledgment

The researchers have gotten a great deal of help and support while creating this study. It could be directly in the composition of this paper or the form of moral support. The researchers would want to express their gratitude.

First and foremost, the researchers wish to express their acknowledgment to Dr. Bobby D. Gerardo, their research adviser, whose skills and experience were invaluable in the production of this paper;

The researchers would like to thank the following consultants for their assistance and guidance: Dr. Ma Luche Sabayle, Dr. Regin Cabacas, Dr. Frank Elijorde, and Sir Mark Solidarios;

Dr. Steven Künzel's assistance and recommendations are also greatly appreciated;

Bitsplash Interactive also offered to assist by providing a free voucher for Graphs and Charts assets which aided in the data visualization of this paper, for which the researchers are thankful;

Finally, the researchers would like to express their deepest appreciation to their families and friends for their unwavering support and encouragement. The researchers

could not have completed this paper without the assistance and guidance of all of the individuals and organizations indicated above, for which they express their heartfelt gratitude.

Bryan Kent S. Abesamis

Timothy A. Diosaban

Alyssa Danielle R. Magallanes

Neil Ryan S. Sustiguer

June 2022

Abesamis, Bryan Kent S.; Diosaban, Timothy A.; Magallanes, Alyssa Danielle R.; Sustiguer, Neil Ryan S.; "Fencing AI; NeuroEvolution of Augmenting Topologies (NEAT) towards Competition and Survival of the Fittest in a Fencing Simulation".Unpublished Undergraduate Thesis, Bachelor ofScience in Computer Science, West Visayas State University, Iloilo City, Philippines, June 2022

Abstract

Neuroevolution is a machine learning technique that uses evolutionary algorithms to create artificial neural networks. In this study, the researchers developed a simulation using the NeuroEvolution of Augmenting Topologies (NEAT) within the Unity3D engine that generates a visualization of how different AI fencers evolve and improve over time. This study highlights the enhancement on NEAT, specifically SharpNEAT, with the addition of two Competitive Agents instead of one, Multi-Objective Optimization techniques, Behaviour Modelling, & Opponent Modelling to show the humanness factor of the study.

The results show that the fencers indeed evolve by developing their behavior strategies by being more aggressive or defensive when adapting to their opponent. The findings amply demonstrate the value of the research's suggested methods and approach, promote its further use, and inspire comparable research for a better understanding

of NeuroEvolution of Augmenting Topologies (NEAT) in competitive learning development.

TABLE OF CONTENTS

List of Figures

List of Tables

List of Appendices

CHAPTER 1 INTRODUCTION TO THE STUDY

Background of the Study

Neuroevolution is a machine learning technique that uses evolutionary algorithms to create artificial neural networks. It is most commonly used in evolutionary robotics and artificial life. It is inspired by the evolution of biological nervous systems in nature. Neuroevolution is to be implemented in this research for creating a smarter AI with NeuroEvolution of Augmenting Topologies (NEAT) in evolution and training. (Lehman, Miikkkulainen, 2013)

NEAT stands for NeuroEvolution of Augmentation of Topologies. It is a method for developing artificial neural networks with a genetic algorithm. NEAT implements the idea that it is most effective to start the evolution of small, simple networks and to allow them to become increasingly complex over generations. That way, just as organisms in nature have increased in complexity since the first cell, so do neural networks in NEAT. This process of continuous development enables the discovery of highly sophisticated and complex neural networks. (Stanley, 2015).

With the help of NEAT, the development and evolution of two AI Fencers are made possible. Through NEAT and its cardinal mixture of architecture in opponent modeling, two

neural nets can learn implicitly and explicitly from each other. Using a mixture-based approach, the two AI Fencers can learn from one another in real-time while taking into account their past generations, which omits the need for a random player or a customized neural network to adapt, satisfy and promote the promise of complexity in NEAT models. Development and evolution are inextricably linked. (Stanley, 2006) Development in the standpoint of NEAT are functions derived from observed gradient patterns present in developing natural organisms while evolution, on the other hand, is the preservation of the regularities while co-existing with variations and anomalies.

In conventional game theory problems, the usual assumption is that decision-makers (players) make their decisions based on a scalar playoff. Usually, the main objective is addressed with the use of a 'fitness score'. (Avigad, Eisendstadt, and Cohen, 2011) For example, in Épée Fencing, the objective is to hit the opponent in any part of the body with the Épée- fitness score is determined by the number of hits the AI lands on each other. If one AI hits the other, it wins a point, if the AI gets hit by its opponent, a point is deducted from its fitness score.

In this study, the researchers employed Evolutionary Multi-objective Optimization (EMO) to address a variety of elements that can affect one's gameplay. A wide variety of problems advocates the simultaneous optimization of several objectives. In many cases, the objectives are defined in incomparable units, and they present some degree of conflict among them (i.e., one objective cannot be improved without deterioration of at least another objective). Let us consider, for example, a shipping company that is interested in minimizing the total duration of its routes to improve customer service. On the other hand, the company also wants to minimize the number of trucks used to reduce operating costs. These objectives are in conflict since adding more trucks reduces the duration of the routes but increases operating costs. In addition, the objectives of this problem are expressed in different measurement units. (James, Martinez, and Coello, 2009)

Under the notion of Fencing AI, Evolutionary Multi-objective Optimization helps to reflect humanness and the nonlinearity of real-life fencers. Winning a fencing game doesn't revolve around merely hitting the enemy. Independent variables such as aggression, cautiousness, stamina, and timing of the player must also be taken into

account to win a round. With EMO, the player can find a favorable balance between these factors in order to refine their playstyle and gain skills or strategies they lack. With the variables considered, fitness scores and weights are modeled within the fuzzy functions of a Multi-Objective Evolutionary Algorithm (MOEA), specifically NSGA-III, in a single run.

Several projects have evolved neural networks to generate images. The Artificial Painter for example uses neural networks with inputs involving orientation and distance from landmark coordinates to either automatically or interactively evolve abstract imagery. Stanley's NEAT infrastructure was used by Fagerlund to evolve complex networks for image generation. Stanley demonstrates the usage of the software for targeted evolution by interactively evolving networks that gradually refine a spaceship design. This approach is replicated in a C# implementation by Ferstl. which adds several interface extensions, in particular giving the user greater control of color. (Lewis, 2008) This concept applies to the study as the distinction of the systems intention to produce visually appealing software with organized data models, seems important to the interpretation of results and

evaluation of success. The questions of stylistic signature and controllably increasing visual diversity can be strongly related to a goal remaining present.

SharpNEAT is the mediator between the evolutionary algorithm and FencingAI, using sockets to be able to run the game simulations in parallel. It gets the configuration from the NEAT module and sets up the game. In this paper, the researchers used uNEATY, a part of a UnityNeat project, which itself is a port of SharpNEAT from pure C# 4.0 to Unity 4x and 5. It is a collection of Unity packages to make it easier to implement NEAT Neuroevolution of Augmenting Topologies neural models with Unity3D. Just like OpenAI's Gym toolkit, the researchers also used the environment to implement the necessary reinforcement learning algorithms. With this simulation and demonstration, this paper showcases the full potential of NeuroEvolution of Augmentation of Topologies (NEAT), where researchers and engineers are not only able to apply different effective optimization techniques but at the same time, show the visualization of the neural and data models of how it grows over time. The application can also be a good reference to start developing neural networks in Unity.

Theoretical Framework

As stated by Than (2018), Darwin's theory of evolution by natural selection is the process by which organisms change over time as a result of changes in heritable physical or behavioral traits. These adaptations allow organisms to survive and reproduce in a changing environment. This model is the basis for NeuroEvolution of Augmentation of Topologies (NEAT). It uses the concept of improving the program generation after generation concerning environmental changes. Since this research primarily uses NEAT, the system architecture is heavily based on this model of evolution.

Based on the research of Gruber(2013) on Nature, Nurture, and Knowledge Acquisition, The first generation of an Artificial Neural Network has to start somewhere. It acts only on the nature of how it is set by the programmer. Nature is acquiring knowledge by being programmed or modeled that way. But an ANN does not improve if it doesn't adapt to environmental changes. Thus nurturance is needed to enhance the program in its next generation by acquiring knowledge through machine learning from data and information in its environment. Further improvements are achieved through countless trials the program experiences

in a changing environment. The next generation has more enhanced intelligence than the previous one.

As claimed by Saumya (2020), ANN's have a predefined model, only the weights of connections can change during training. A human brain has about 86 billion neurons, while an artificial neural network has 10-1000 neurons. Additionally, there is no way to add or delete neurons in an ANN. An Artificial Neural Network's training session consists of random data sets rather than the same set of examples. A human brain, on the other hand, does not typically start or stop learning; instead, it learns by repetition and even while sleeping. In a fencing game environment, initially, all that is needed is two predefined models of first-generation fencers. This will be the baseline for future generations of neural networks.

According to Danny Zhu (2020), NEAT (1) utilizes historical markings to tack genes and avoid competing conventions, so that a meaningful crossover could happen and less topological analysis is required; (2) separates the population into species based on compatibility distance, so that competition is primarily within the same niche and innovation is protected; (3) starts from the simplest

structure and grows only as necessary through mutation, so that it's faster to find a solution.

According to Rodriguez (2019), Collaboration and competition are two of the key pillars of the evolution of human societies and essential to our evolution as species. In a competitive scheme, multiple neural networks in a given generation compete to survive and be updated to the next generation. Compare this to a cooperative scheme where neural networks work together to survive and are updated as one symbiotic team. In a fencing environment, however, two neural networks are faced off against each other with the winning neural network updated to the next generation. The neural network that proceeds to the next generation then faces off with the same neural network as itself, thereby improving generation after generation.

Chen, Locket, and Miikkulainen (2007) expressed that Opponent Modeling in NEAT is vital in competitive games such as Fencing, especially in creating counterstrategies. Opponent Models can be explicit or implicit. Implicit Modeling means a neural net can train against another trained neural net with an already defined 'personality or set of skills' or Explicit, where an AI fighter goes against another AI fighter with 'inputted actions'. The

goal for opponent modeling is to defeat specific opponents while for explicit, is intended to generalize directly to previously unseen opponents without any extra training by enabling recognition of unseen opponents.

Kasimoglu's (2016) study on Interactive Approaches for Multi-Objective Decision-Making Problems showcases that Evolutionary Multi-Objective Algorithms using interactive approaches have become a trend for researches that involve a lot of decision-making. These algorithms are considered to be more efficient for problems with a large number of objectives. Typically, the decision-maker is integrated into the process by expressing his/her preference when generating solutions through an evolutionary algorithm.

Agapitos et. al (2008), Projects related to NEAT that involve advanced game playing experiments use SharpNEAT, a module or mediator between the evolutionary algorithm and Unity. Green(2006) indicates that SharpNEAT is a complete implementation of NEAT written in C# and targeting .NET (on both MS Windows and Mono/Linux).

Objectives

The general objective of this thesis is to develop a simulation using the NeuroEvolution of Augmenting Topologies (NEAT) within the Unity3D engine that generates a visualization of how different AI fencers evolve and improve over time.

The specific objectives of this thesis are:

1. Implement multi-objective technique enhancement on the NEAT Algorithm by using simulated fencing competition as its main learning process.

2. Effectively use multi-objective optimization in Sharp NEAT to find all possible moves/solutions to given constraints.

3. The optimization of the Fencing AI's input and output strategy, find behavioral models better than their counterparts, and aspects such as ‘‘humanness’’ of an AI fighter will also be investigated.

Significance of the Study

In accordance with the aforementioned objectives, this research would become one of the foundations for studies that use NEAT in applications that involve various types of strategy, while considering two or more objectives. The study used Multi-Objective Optimization by addressing different conflicting factors to find an optimal solution for a certain problem. The output of this algorithm would be deemed useful in competitive applications that require achieving a better win/loss ratio because they would be able to train and view models that can supplement their future 'gameplay'. This would also give a visual insight on how their neural net would train itself while taking into account different critical components, that would be highly effective in adapting and addressing several problems.

Definition of Terms

Classic NEAT

As stated by Buckland & LaMothe (2002), NEAT (NeuroEvolution of Augmenting Topologies) is an evolutionary algorithm that creates artificial neural networks, It uses node-based encoding to describe the network structure and connection weights, and avoids the competing convention problem by utilizing the historical data generated when new nodes and links are created. NEAT also attempts to keep the size of the networks it produces to minimum by starting the evolution using a population of networks of minimal topology and adding neurons and connections throughout the run. The researchers utilize this algorithm for the artificial intelligence in the program to improve over time.

In the context of this research NeuroEvolution of Augmenting Topologies (NEAT) is the algorithm used for training the fencers.

SharpNEAT

Green (2006) defines that SharpNEAT provides an implementation of an Evolutionary Algorithm (EA) with the specific goal of evolving neural networks. The EA uses the evolutionary mechanisms of mutation, recombination and

selection to search for neural networks with behaviour that satisfies some formally defined problem. Example problems might be how to control the limbs of a simple biped or quadruped to make it walk, how to control a rocket to maintain vertical flight, or finding a network that implements some desired digital logic (such as a multiplexer).

In the context of this research, It is a port to the Unity game engine which adds an interface to connect the algorithm directly to a scene in Unity.

Machine Learning

Machine learning is a method where information is processed by an artificial intelligence where it is to learn from the data, identify patterns, and to make decisions with minimal human intervention. In the context of the research, machine learning is the method of training artificial neural networks for the fencing game.

Artificial Neural Networks

An artificial neural network is a series of algorithms that mimics the process of how the human brain operates when given data. Within the NEAT algorithm, artificial neural networks are created which hold the different processes for the program to do. In the context of this

research, Artificial Neural Networks are to be trained for fencing using the NEAT algorithm.

Opponent Modeling

Opponent modeling concerns establishing models of the opponent player, and utilizing the models in actual play. In general, an opponent model is an abstracted description of a player or of a player's behavior in a game. The goal of opponent modeling is to improve the capabilities of the artificial player by allowing it to adapt to its opponent and exploit his weaknesses.

The researchers applied Opponent Modeling to implement and design the humanness factor of FencingAI neural nets. It provides the researcher's references to different behavioral models suited for the study.

Artificial Intelligence (A.I.)

Artificial intelligence (AI) is the ability of a computer or a robot controlled by a computer to do tasks that are usually done by humans because they require human intelligence and discernment. In the context of the research, the term artificial intelligence is the fencers within the simulation being fed actions via ANN's.

Fuzzy Functions

Fuzzy functions consist of crisp functions with fuzzy constraints and fuzzifying functions. Maximizing set and minimizing set are also introduced and applied to find the maximum value with a fuzzy domain of the crisp function. In the context of the research fuzzy functions are used in the creation of the Fitn

Produce

uNEATy is a fork of the UnityNEAT project - which is itself a port of SharpNEAT from pure C# 4.0 to Unity 4.x and 5 (using Mono 2.6). It is a collection of Unity packages to make it easier to implement NEAT Neuroevolution of Augmenting Topologies neural models within Unity3d. The researchers also used this in the study to be able to optimally use SharpNEAT within the Unity Game engine.

Unity Asset Store

The Unity Asset Store is home to a growing library of free and commercial assets created both by Unity Technologies and also members of the community. In the context of this study, the assets used by the researchers are purchased on the Asset Store website and are added into their projects using the Package Manager, within the Editor window of Unity.

Mixamo

Mixamo is an online database of characters and mocap animations that anyone can access to be used in art projects, movies, and games. These models and animations can then be exported for use across a wide range of software including Unity, Blender, and Adobe Photoshop. The researchers used the free available fencer model within Mixamo.

Evolutionary Algorithm

An evolutionary algorithm (EA) is an algorithm that uses mechanisms inspired by nature and solves problems through processes that emulate the behaviors of living organisms. In EAs, the solutions play the role of individual organisms in a population. In the context of this research, EA's such as MOEA and NEAT were used by the researchers and then developed the Fencing A.I. 's like the 'solutions'.

Multi-objective Evolutionary Algorithm

Multi-objective EAs, or MOEAs, are used to solve problems that have multiple, and often conflicting, objectives. A central concept for MOEAs, and multi-objective optimization in general, is that of a non-dominated solution. This is a solution that is no worse

than any of the other solutions within the population when all objectives are taken into account, and an MOEA aims to build and maintain a population of non-dominated solutions that cover all trade-offs between the objectives. This is known as a Pareto Optimal Front.

A multi-objective Evolutionary Algorithm was applied to the study as FencingAI focuses on two or more objectives.

Genetic Algorithm

Genetic algorithms are a search-based optimization technique based on the principles of Genetics and Natural Selection by Charles Darwin. In the context of this research, NEAT itself is a genetic algorithm and these kinds of algorithms are used in making or training artificial neural networks where they improve generation by generation.

Non-dominated Sorting Genetic Algorithm III (NSGA-III)

NSGA-III was developed as a means to improve NSGA-II for Multi-Objective Problems that had 4 or more objectives. It highlights the increasing computational effort required and the problem of removing dominated solutions as their fraction of the population decreases as the amount of objectives increases. It improved on these issues by using

reference points to guide the selection of solutions. These reference points are predetermined to ensure diversity in the solutions and are used to assert a potential reference set to approximate distances between the current generation and the Pareto front. This is the technique the researchers used to optimize the 4 objectives in the study.

Fitness Function

In the context of this research, the fitness function is a function that takes the solution as input and generates a fitness score as output to evaluate the quality of each solution which is the fitness score.

Population

In the context of this research, the population is the subset of all candidate solutions to the given problem.

Chromosome

In the context of this research a chromosome is each candidate solution in the population, sometimes referred to as a genome.

Selection

In the context of this research, a selection is a process of filtering and retaining solutions according to fitness scores. Solutions with higher fitness are more likely to be pushed forward into the next generation. A

pair of selected solutions could be chosen as parents to breed and propagate their genes to the subsequent generation via crossover.

Crossover

In the context of this research, a crossover is the way parent genomes aproduce child genomes. Genes of parents are reassembled based on a set of crossover points to formulate new offspring.

Fencing

Fencing is a group of three related combat sports. The three disciplines in modern fencing are the foil, the épée, and the saber (also saber); winning points are made through the weapon's contact with an opponent (Roi & Biachendi, 2008). In the context of the research, fencing would be the game for teaching artificial neural networks to learn how to play and to train them to improve gradually.

Competition

The base definition of competition is the act or process of trying to get or win something (such as a prize or a higher level of success) that someone else is also trying to get or win: the act or process of competing (Merriam-Webster Dictionary). Within the context of this study and within the NEAT algorithm, competition comes up

when the algorithm will look at which iteration of the current generation has the highest fitness score to be picked as a template for the next generation. (Buckland & LaMothe, 2002

Genomes

A genome is an organism's complete set of genetic instructions. In the context of the research, genomes are the set of genes that together code for a (neural network) phenotype.

Genes

A gene is the basic physical and functional unit of heredity. Genes are made up of DNA. In the context of the research, genes hold the information coding (in the current implementation) for a particular aspect (node or connection) of a neural network phenotype. Contains several attributes, varying depending on the type of gene.

Generations

A type or class of objects usually developed from an earlier type. In the context of the research, generations are the iterations for each time the AI completes and renews to another.

Nodes

A node is a basic unit of a data structure, such as a linked list or tree data structure. Also known as a neuron in a neural network. They are three types of nodes: input, hidden, and output. Nodes have one or more attributes, such as an activation function; all are determined by their gene. In the context of the research nodes are the connections between the genomes.

Species

Subdivisions of the population into groups of similar (by the genomic distance measure) individuals (genomes), which compete among themselves but share fitness relative to the rest of the population. This is, among other things, a mechanism to try to avoid the quick elimination of high-potential topological mutants that have an initial poor fitness before smaller "tuning" changes. In the context of the research, the term species are groups of virtual fencers which share similar fitness scores relative to the rest of the population.

NeuroEvolution

Neuroevolution is a machine learning technique that applies evolutionary algorithms to construct artificial neural networks, taking inspiration from the evolution of

biological nervous systems in nature (Lehman & Miikkulainen, 2013). In the context of the research, NEAT makes use of the technique to create and augment various neural networks.

Augmentation

Defined as the action or process of making or becoming greater in size or amount. In the context of the research, the Artificial Neural Networks (ANN) are augmented "Because smaller structures optimize faster than larger structures, and adding nodes and connections usually initially decreases the fitness of the network, recently augmented structures have little hope of surviving more than one generation even though the innovations they represent might be crucial towards solving the task in the long run" (Stanley & Miikkulainen, 2002).

Topology

The dictionary definition of topology is a branch of mathematics concerned with those properties of geometric configurations (such as point sets) which are unaltered by elastic deformations (such as stretching or a twisting) that are homeomorphisms. In the context of the research, topology is the configuration of how all neurons and layers are interconnected, as well as the number of layers (Anderson, James. 1995).

Weight

In the context of the research, the term weights (commonly referred to as w) and the partner term biases (commonly referred to as b) are the learnable parameters of the machine learning model. When the inputs are transmitted between neurons, the weights are applied to the inputs along with the bias.

Mutation

A mutation is defined as the change that occurs in a DNA sequence. In the context of the research, mutation is the process in which the attributes of a gene (or the genes in a genome) are (randomly, with likelihoods determined by configuration parameters) altered.

Fitness Score

In the context of the research, a fitness score is a mathematical function that rewards success. In combat sports like fencing, the fitness score would be the ability of a person to hit the other person. There can be many more variables included within the Fitness Score function, like correct posture, defeated enemies, or the overall time to finish.

Unity

Unity is a free-to-use, cross-platform game engine, meaning it is used to develop for various operating systems and devices, ranging from Windows PCs to PS, to Mobiles. It is one of the most prominent game engines in the industry and is used by millions of people around the globe. With it, creating games ranging from simple 2D side scrollers to hyper-realistic open-world games. In the context of the research, Unity is the game engine used to create and simulate the fencing game.

Unity's Scriptable Render Pipeline

The Scriptable Render Pipeline is a thin API layer that lets you schedule and configures rendering commands using C# scripts. Unity passes these commands to its low-level graphics architecture, which then sends instructions to the graphics API.

Tick/s

In the context of the research, the term tick/s is used to describe as the unit of measure for time, specifically, it refers to a single instance of a repeated action (usually a broad action) in a game or this studies sense within the simulation using the Unity3D game engine,

or the period that action consumes. Ticks are repeated and (mostly) regular.

Behavioral Model/s

A Behavioral Model is specially designed to understand behavior and factors that influence the behavior of a System. It usually describes overall states that a system can have and events that are responsible for a change in the state of a system. In the context of this research behavioral models are what the fencing A.I. can become after a run of the simulation, it can either be Aggressive, Defensive, Balances, and Presumptuous.

Aggressive/Offensive Model/s

Aggressive or Offensive models are the ones that go beyond their zone and meet their enemies head-on. These models prefer to perform riskier plays in the pursuit of points and hits.

Defensive/Cautious Model/s

Defensive or Cautious models showcase a defensive behavior that could either land them a win or loss. These models spend most of their time in their zones preferring to play safe.

Balanced/Stable Model/s

A Balanced or Stable model is the one that performs consistently as both an offensive and defensive model.

Presumptuous Models

A Presumptuous model sticks to one strategy and is overconfident in implementing it, thus resulting in poor performance, fitness score and usually, these are the fencers that lose most of the time.

Scope and Delimitations of the Study

The delimitations/limitations of this research are the characteristics that limit the scope and define the boundaries of the study which are under the control of the Researchers. They will involve four characteristics in the study namely: Fencing, Algorithm, Programming Language, and Graphic assets.

The researchers will use the combat sport of fencing as the basis of the simulation within the program, specifically towards épée, one of its disciplines, following its set of rules and moves exercised.

They will focus on the development of a program that is based on the NeuroEvolution of Augmenting Topologies (NEAT) algorithm wherein two AI Agents perform the combat

sport of fencing against each other. Both would improve over time. They will implement NSGA-III, a multi-objective optimization technique, to the algorithm to address the objectives of the game.

They used the C# programming language as the exclusive language when developing the program and also makes use of the Unity Game Engine to simulate the gameplay and real-time training data models of both fencers.

They also used graphic assets from the Unity Store-fencer models to graphs and various data models.

CHAPTER 2 REVIEW OF RELATED STUDIES

Games and simulations have come a long way especially with their Non-playable Characters that have artificial intelligence integrated within them. Those who have been playing single player games are thankful for the improvement on how these A.I.'s act, especially when there's a repetition of actions within a game or simulation and having a good A.I. that can learn from previous fights to fight against makes the player think of various situations on how to beat the A.I. in different ways.

Especially in the games of combat sports where two combatants have to be physically fit to be tough and powerful enough to beat their opponents but also at the same time mentally fit to think on the go on how they can beat their opponents in the least and efficient moves as possible and the combat sports of fencing is the best thing to simulate an A.I. that is physically powerful and can make great decisions on the field.

In an article written by Roi and Bianchedi (2021), Fencing is an open-skilled combat sport that was admitted to the first modern Olympic Games in Athens. It is mainly practised indoors, with three different weapons: the foil, the sabre and the épée, each contested with different rules.

In this study, the researchers would be using the épée along with its standard rules and gameplay.

According to the Fencing Academy of Boston (2021), A game of fencing takes place on a "strip" or a "piste", about 14 meters long and 2 meters wide. In modern fencing, the equipment (Weapons and armor) are electrically wired from sword to their sleeve which connects to an electronic scoring machine paired with their opponent, and then, next test is that their weapons and cords are functioning correctly by touching their opponent on their target, a metallic garment called a *lamé* (except in the case that epeéists, who do not wear a lamé, test by touching the point to the guard of their opponent's weapon). They return to their respective starting lines 4 meters apart, salute their opponent, put on their mask, and get in the en garde position, facing their opponent and ready to fence. The referee then calls the beginning of the bout. Fencers remain facing the direction of their opponent and are not to leave the strip during the bout.

The epee (pronounced "EPP-pay," meaning sword in French), the descendant of the dueling sword, is similar in length to the foil, but is heavier, weighing approximately

27 ounces, with a larger guard (to protect the hand from a valid hit) and a much stiffer blade. Touches are scored only with the point of the blade, and the entire body, head-to-toe, is the valid target area, imitating an actual duel. In this study, the epee is considered to be the main focus

Opponent Modelling

Lockett et. al (2018) discussed how opponent models are necessary for games where the game state is only partially known to the player since the player must infer the state of the game based on the opponent's actions. This paper presents an architecture and a process for developing neural network game players that utilize explicit opponent models to improve gameplay against unseen opponents. The model is constructed as a mixture over a set of cardinal opponents, i.e. opponents that represent maximally distinct game strategies. The model is trained to estimate the likelihood that the opponent will make the same move as each of the cardinal opponents would in a given game situation. Experiments were performed in the game of Guess It, a simple game of imperfect information that has no optimal strategy for defeating specific opponents. Opponent

modeling is, therefore, crucial to play this game well. Both opponent modeling and game-playing neural networks were trained using NeuroEvolution of Augmenting Topologies (NEAT). The results demonstrate that game-playing provided with the model outperforms networks not provided with the model when played against the same previously unseen opponents. The "cardinal mixture" architecture, therefore, constitutes a promising approach for general and dynamic opponent modeling in game playing.

Opponent modeling is used in the game-playing literature to refer to the process of training computer players to play against specific opponents. General strategies that can defeat all opponents do not exist, e.g., in games where the state space is too large to model, or where aspects of the current game state are not observable.

Van De Herik, Donkers, Spronck (2005) In general, an opponent model is an abstracted description of a player or a player's behavior in a game. There are many different types. For instance, a model can describe a player's preferences, strategy, skill, capabilities, weaknesses, knowledge, and so on. For each type, we may distinguish two different roles in a game program. The first role is to model a (human or computer) opponent in such a way that it

informs the player appropriately in classical two-person games. Such an opponent model can be implicit in the program's strategy or made explicit in some internal description. The task of such an opponent model is to understand and mimic the opponent's behavior, in an attempt either to beat the opponent or to assist the opponent. The second role is to provide an artificial opponent agent for the own agent (program or human player) using the program, or an artificial agent that participates in an online multi-person game. Iteratively, such an opponent agent could bear in itself an opponent model of its opponents. In most cases, the task of an opponent model in this second role is to manifest an interesting and entertaining opponent to human players. Regardless of its internal representation, an opponent model may range from statically defined in the program to dynamically adaptable. Opponent models that are dynamically adapted (or adapt themselves) to the opponent and other elements of the environment are to be preferred.

NEAT and Neuroevolution

Stanley (2015) described NEAT as a method for evolving artificial neural networks with a genetic algorithm. NEAT implements the idea that it is most effective to start evolution with small, simple networks and allow them to

become increasingly complex over generations. That way, just as organisms in nature increased in complexity since the first cell, so do neural networks in NEAT. This process of continual elaboration allows finding highly sophisticated and complex neural networks.

Stanley and Miikkulainen (2002) state that Neuroevolution (NE), the artificial evolution of neural networks using genetic algorithms, has shown great promise in reinforcement learning tasks. NeuroEvolution of Augmenting Topologies (NEAT) is designed to take advantage of an evolving structure as a way of minimizing the dimensionality of the search space of connection weights. If the structure is evolved such that topologies are minimized and grown incrementally, significant performance gains result.

They also clarified that in NEAT, each genome includes a list of connection genes, each of which refers to two-node genes being connected (figure 2). Each connection gene specifies the in-node, the out-node, the weight of the connection, whether or not the connection gene is expressed (an enable bit), and an innovation number, which allows finding corresponding genes during the crossover. NEAT can also evolve networks with any number of inputs or outputs.

Heindrich(2019) states that mutation in NEAT can change both connection weights and network structures. Connection weights mutate as in any NE system, with each connection either perturbed or not. Structural mutations, which expand the genome, occur in two ways (Figure 3). In the add connection mutation, a single new connection gene is added connecting two previously unconnected nodes. In the add node mutation, an existing connection is split and the new node placed where the old connection used to be. The old connection is disabled and two new connections are added to the genome. This method of adding nodes was chosen to integrate new nodes immediately into the network. Through mutation, genomes of varying sizes are created, sometimes with completely different connections specified at the same positions.

Minh et.al (2015) explains that the theory of reinforcement learning provides a normative account, deeply rooted in psychological, and neuroscientific perspectives on animal behavior, of how agents may optimize their control of an environment. To use reinforcement learning successfully institutions approach real-world complexity, however, agents are confronted with a difficult task: they must derive efficient representations of the environment

from high-dimensional sensory inputs, and use these to generalize experience to new situations. Remarkably, humans and other animals seem to solve this problem through a harmonious combination of reinforcement learning and hierarchical sensory processing systems, the former evidenced by a wealth of neural data revealing notable parallels between the phasic signals emitted by dopaminergic neurons and temporal difference reinforcement learning algorithms. While reinforcement learning agents have achieved some successes in a variety of domains. Their applicability has previously been limited to domains in which useful features can be handcrafted or to domains with fully observed, low-dimensional state spaces. They use recent advances in training deep neural networks to develop a novel artificial agent, termed a deep Q-network, that can learn successful policies directly from high-dimensional sensory inputs using end-to-end reinforcement learning. They tested this agent on the challenging domain of classic Atari 2600 games.They demonstrated that the deep Q-network agent, receiving only the pixels and the game score as inputs, was able to surpass the performance of all previous algorithms and achieve a level comparable to that of a professional human games tester across a set of 49 games,

using the same algorithm, network architecture, and hyperparameters. This work bridges the divide between high-dimensional sensory inputs and actions, resulting in the first artificial agent that is capable of learning to excel at a diverse array of challenging tasks.

NEAT was introduced by Stanley and Miikkulainen (2002). Due to its typical characteristics as a genetic algorithm, NEAT shares similarities with reinforcement learning methods. For instance, both methods do not require any labeled training data, because learning signals are generated by a fitness function (genetic algorithm) / a reward function (reinforcement learning), which evaluates the solution quality of any ANN output. NEAT is considered as the first neuro-evolution strategy that can efficiently adapt the structure of an ANN, mainly because of four features.

Stanley and Miikulainen (2002) explained that in contrast to other neuro-evolution approaches, the genetic encoding of NEAT genomes is dynamic and expandable, i.e. not restricted by a fixed dimension. Each genome consists of a set of neuron genes and synapse genes. A neuron gene comprises all information of a neuron, i.e. its bias, response, and activation function. A synapse gene consists of an id, which indicates the interconnected neurons, the current weight of the interconnection, the status of the interconnection (enabled/disabled), and its innovation number, which is necessary to determine whether two genomes can perform a crossover. NEAT provides two mutation features to extend genomes and thus to adapt the structure of ANNs. A synapse mutation generates a new connection between two unconnected neurons, while a neuron mutation removes an existing connection and adds a new neuron instead, which maintains a connection to the previously connected neurons.

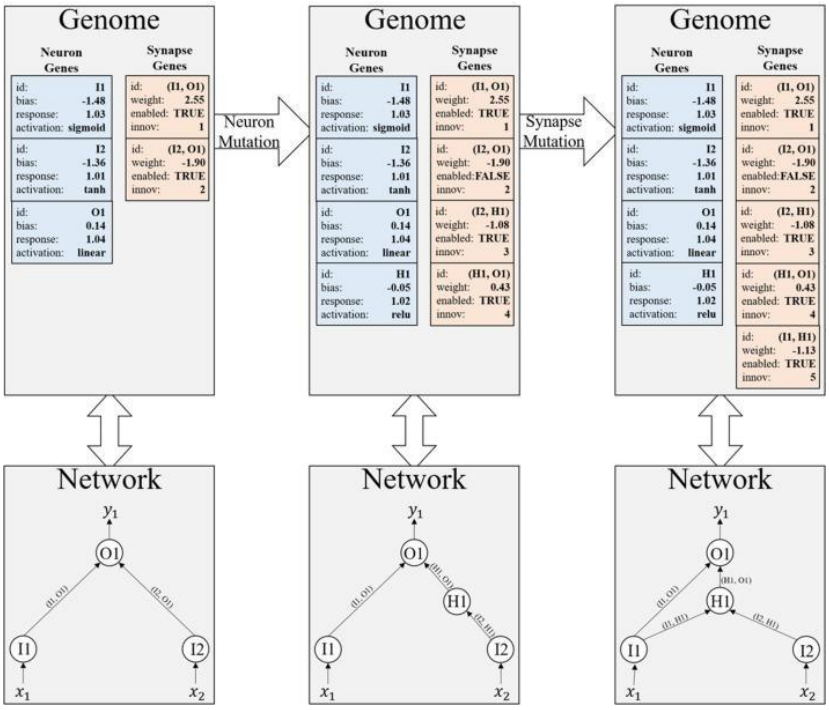


Figure 1. Illustration for the encoding of genomes in NEAT with an example for both a neuron and synapse mutation.

Figure 1 illustrates the encoding of genomes in NEAT as well as an example for both a neuron and synapse mutation. The flexible and expandable encoding is highly efficient, because the size of genomes corresponds to the amount of encoded information. The encoding also provides high scalability, as NEAT can construct ANNs of any complexity by simply adding additional information to existing genomes.

Genetic encoding and mutation features in NEAT: The genome on the left-hand side undergoes first a neuron mutation and thereafter a synapse mutation. In the given example, NEAT randomly removes a synapse ((I2, O1)), creates a new neuron with random attributes (H1), and initializes randomly weighted synapses ((I2, H1); (H1, O1)) to the previously connected neurons. Thereafter, NEAT creates a new synapse ((I1, H1)) with a random weight between two randomly selected unconnected neurons (I1; H1).

A meaningful crossover of information between genomes that represent different topologies is challenging, as network topologies of different sizes also require different genome dimensions to encode all information. To deal with this problem, the algorithm keeps track of the origin of each synapse gene by assigning to each gene a unique innovation number. It is not necessary to also track the origin of neuron genes, as the IDs of the synapse genes already comprise all necessary information about the structure of an ANN, i.e. which neurons maintain a connection to each other. Fig. 2 gives an example of a crossover between two genomes that are partly different in terms of their synapse genes. When pairing two genomes, NEAT randomly selects matching synapse genes, i.e. genes

that are present in both parent genomes. For the remaining genes that are different, NEAT only considers the genes of the parent genome with better fitness. In case that both parents have the same fitness, NEAT adds the unmatching synapse genes of both parents to the offspring genome.

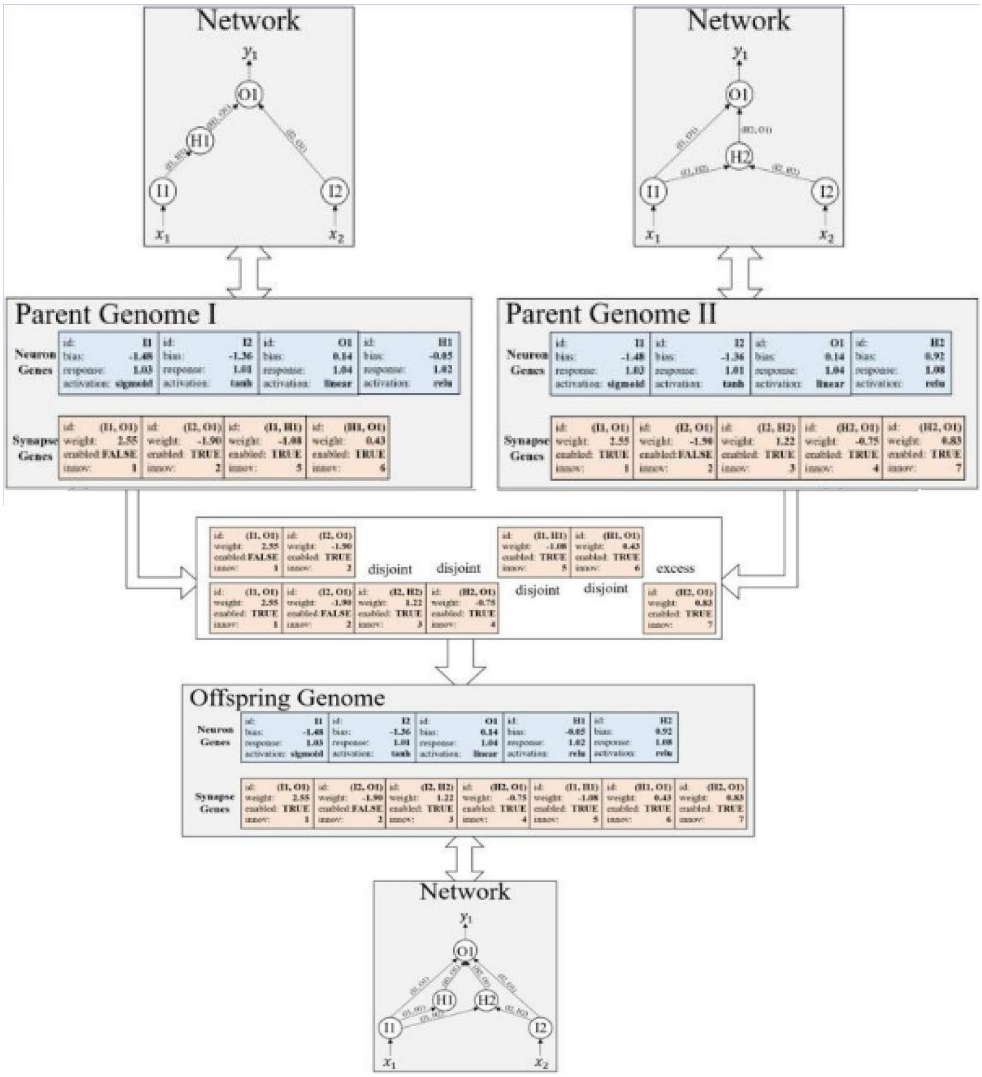


Figure 2. Exemplary crossover of two genomes.

Both parent genomes are matching in terms of the synapse genes with innovation numbers 1 and 2. In the given example, NEAT picks both matching genomes to construct the offspring genome. Furthermore, NEAT considers all unmatching genes in the offspring genome, which indicates that the parent genomes have the same fitness.

Lang et. al (2021) explains that the algorithm divides the total population into different species according to topological similarities so that genomes compete only within their species. They usually cannot immediately compete with longer established genomes having been able to optimize the synapse weights of their current network topology over several generations. NEAT divides the population into species depending on the genetic distance between the genomes.

Stanley and Miikulainen (2002) further discussed that other neuro-evolution implementations suffer from low computational efficiency because they already need to maintain different topologies in the initial population. That leads to high-dimensional genomes even before the first iteration starts. For many neuro-evolution approaches, it is necessary to consider different topologies in the initial population, as a significant change in the topology

will likely lead to an initial fitness loss. Due to the concept of specification, NEAT does not require considering different network structures while initializing a population. Thus, initial genomes do not consider hidden neurons and differ only in terms of the encoded parameters like weights and biases. NEAT extends the topology of ANNs incrementally, and only those topologies survive that can compete with other genomes of the same species. NEAT does not restrict the number of hidden neurons. Thus, the algorithm is theoretically applicable for problems of any complexity.

In their book *Genetic Twists of Fate,* Fields, and Jhonston (2010) claimed that now and then, a mutation causes a shift in the function of a vital protein, making the organism better able to compete for resources with its peers, resulting in the mutation spreading across the population with each generation. Most people eventually bear the mutation, and the characteristic it confers on them becomes dominant. The population has changed over time. It's a slow process, but it's been going on for a long time—the planet evolved over four billion years ago—and it happens. The incredible variety of life on this planet, which so enthralled Wallace, is proof of that.

Hintze et al. (2015) demonstrated that two agents in a *direct competitive environment* take a sample from the same urn and compare it to their reference urn to see if it has a higher or lower mean. The game ends when one agent chooses to keep the reference urn or choose the sampled urn, and the other agent earns the value of its reference as a payoff. If both agents choose the same urn at the same time, it is randomly assigned to one of them. As a result, in a direct competition situation, one agent's decision has a direct effect on the other's option climate and, as a result, fitness.

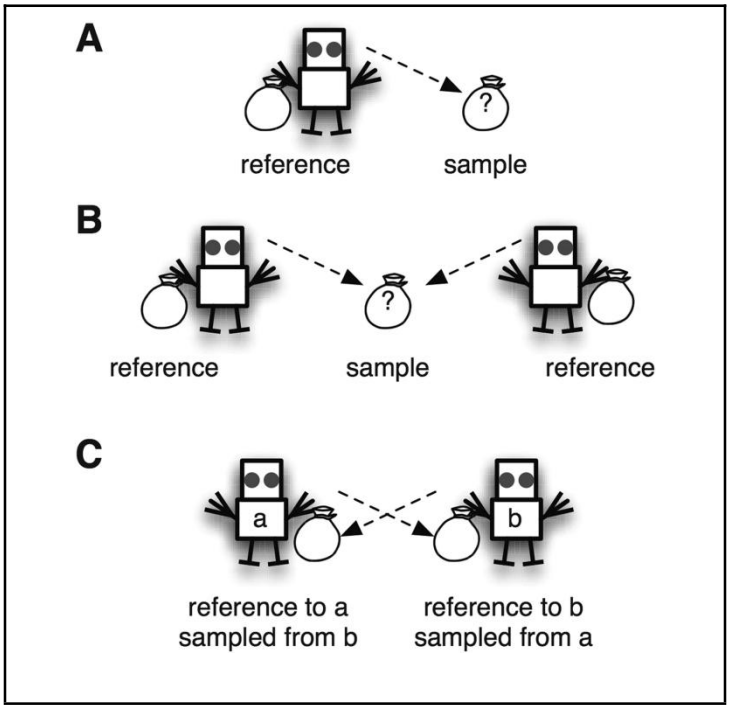


Figure 3. *The Janus face of Darwinian Competition*

Illustration of the three competitive environments

Panel (A) illustrates the indirect competition environment, where the agent, due to sufficient sampling in the past, knows the value of the urn it holds. The agent can sample from the unknown urn and chooses the urn with the higher payoff. Panel (B) illustrates the direct competition environment, where both agents compete over the same urn, while knowing the value of their reference urn. Panel (C) illustrates the extreme competition environment, where both agents sample from each other's urn and can select the opponent's urn, leaving their urn to the opponent.

Ferner, Fischler, Zarubica, and Stuck (2018) explains that ANN tries to emulate the immense success of its biological counterpart by abstracting the complex chemical reactions responsible for our thoughts to much more graspable math. The feedforward version of such an ANN consists of two simple components: neurons and connections. Each neuron has inputs, which are the incoming connections. It applies a simple mathematical operation to this set of inputs and returns the result.Connections connect neurons to each other. Each connection has a weight, which determines how weak or strong the connection is. The neurons are typically organized into layers. The first is

referred to as the input layer and the last one as the output layer. The remaining layers are called hidden layers (Anderson, James. 1995).

Ferner et. al (2018) discussed that for genetic algorithm training starts with a number of genomes, typically referred to as the population. For each of these genomes, a network is built and it is tested against the expected outputs. From these results, we can assign fitness to the genome. A higher fitness indicates that the genome was able to solve a problem better than another. (Anderson, James. 1995)The initial set of genes is the first generation. The weights of all genes are set to a random value. To get to the next generation, all genomes have to be tested. Before that, each genome has a chance that a random gene mutates, E.g. The gene is assigned a new random weight. After that, we select the genomes for the next generation. To select a genome, a so-called roulette wheel selection is performed. This means that every genome has a chance to get to the next generation, based on its fitness. (Bäck, Thomas. 1996) We always select two genomes at a time, so that we can perform a crossover. This means that we swap a part of the genes in the first genome with the second. This process

is repeated until a genome reaches the target fitness, which is set by the trainer.

In a study by Whiteson, Stone, Stanley, and Mikkulainen(2005), a critical feature of NEAT is that it begins with networks of minimal topology (i.e. with no hidden nodes and all inputs connected directly to the outputs). As evolution proceeds, NEAT adds links and hidden nodes through mutation. Since only those additions that improve performance are likely to be retained, it tends to find small networks without superfluous structure. Starting minimally also helps NEAT learn more quickly. When networks in its population are small, it is optimizing over a lower-dimensional search space; it jumps to a larger space only when performance in the smaller one stagnates.

Competitive NEAT

Lang et. al (2021) described that the main idea of Competitive NEAT is that two NEAT algorithms compete in a turn-based manner. In each turn, one of the NEAT algorithms tries to beat the best fitness of the other. InMost the beginning, they initialize a global fitness variable with a high negative value. They executed for the first time NEAT for finding an ANN that allocates jobs to SMDs. At this moment, NEAT cannot utilize any ANN for the sequencing

problem, as the NEAT algorithm for the sequencing task has not been executed yet. Alternatively, they used the FIFO dispatching rule for the sequencing task. Due to the low initial fitness, NEAT will likely find a solution that exceeds the global fitness value within a single generation. If a better fitness value is found, the execution of the NEAT algorithm for allocation terminates and the current generation, as well as the best genome, are saved. Then, NEAT for sequencing will try to find an ANN representation that has a better fitness than the best genome of the previous executed NEAT session. When evaluating the fitness value of a sequencing genome, the best ANN of the previous NEAT session is utilized for the allocation task and vice versa. As soon as the NEAT algorithm for the sequencing problem finds a genome that exceeds the fitness threshold, the algorithm saves the current population and the genome with the best fitness value in a local folder before terminating. The NEAT algorithm for the allocation problem now continues to evolve the last generation of the previous allocation experiment, instead of creating a new generation from scratch. This process repeats until both algorithms cannot improve the global fitness within a user-defined number of generations.

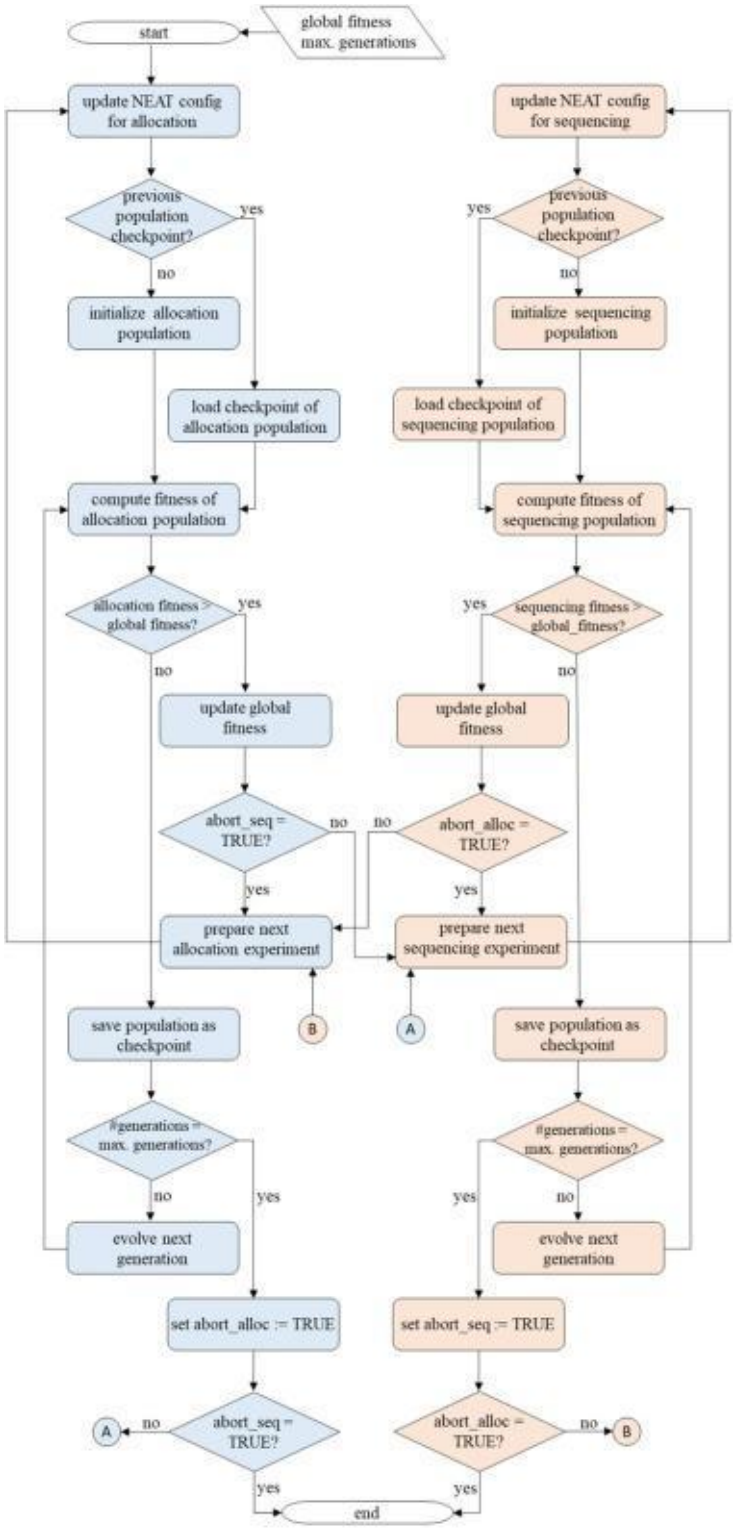


Figure 4. Process flow chart of competitive NEAT

Figure 4 Illustrates the blue flowchart elements represent NEATs search for an ANN representation that solves the allocation problem, while the orange elements describe the search for an ANN that solves the sequencing problem.

Multi-Objective Optimization

Yi Ren Cheng (2017) illustrated that multi-objective optimization problems often deal with multiple conflicting objectives and may be subject to many constraints. In the case of buying a used car, there could be several constraints such as our budget, acceptable mileage, manufacture years, car brand and model, and so on. Due to its computational complexity and the huge solution space presented, it has always been a challenge solving multi-objective optimization problems. In fact, most real-life multi-objective optimization problems are often of exponential size; a straightforward reduction from the knapsack problem shows that they are NP-hard to compute. Thus, it is computationally too costly to find an exact optimal solution if one exists. And most of the time in real applications it is quite hard for the decision-maker to have all the information to correctly and completely formulate them. Therefore, in such situations finding a

near-optimal solution is a practical approach that fits into multi-objective optimization problems. The set of all feasible solutions is called the Pareto curve or surface which represents the solution space of the multi-objective optimization problem. Due to conflicting objectives and constraints, multi-objective optimization problems lead to not a single optimal solution, but a set of non-dominated solutions which is called the Pareto front.

Coello et. al (2019) also clarified that because of conflict among the objectives, solving a MOP produces a set of solutions representing the best possible trade-offs among the objectives (i.e., solutions in which one objective cannot be improved without worsening another one). Such solutions constitute the Pareto optimal set and the image of this set (i.e., the corresponding objective function values) form the so-called Pareto front.

Coello et. al (2007) further expressed that a wide variety of mathematical programming techniques have been developed to tackle MOPs since the 1970s, such techniques present several limitations, from which the most remarkable is that these algorithms are normally quite susceptible to the shape and/or continuity of the Pareto front and that

they usually generate one element of the Pareto optimal set per algorithmic execution. Additionally, some mathematical programming techniques require that the objective functions and the constraints are provided in algebraic form and in many real-world problems that researchers can only obtain such values from a simulator. These limitations have motivated the use of alternative approaches, from which metaheuristics have been a very popular choice, mainly because of their flexibility (i.e., they require little domain-specific information) and their ease of use. Evolutionary algorithms have certainly been the most popular in the last few years in this area, giving rise to a field now known as evolutionary multiobjective optimization.

Reynoso-Mesa et. al. (2014) explain that a MOP consists of three stages: its definition, the optimization problem (search), and multicriteria decision-making (MCDM). In an a priori multiobjective approach, the stages of optimization and decision-making are carried out at the same time, which results in a single solution, since the desired preferences are defined beforehand. On the contrary, in an a posteriori approach, the search does not supply a

single solution but a set of optimal solutions (Pareto optimal solutions). This procedure is more time-consuming, but it gives the designer information about different solutions and provides them with a better understanding of the problem, which will allow them to make a well-informed decision in the decision-making stage. On the other hand, an optimization problem may have multimodal solutions or nearly optimal solutions that are useful for the designer. These alternative solutions, which have a similar or even the same performance as the optimal solutions, are ignored in a traditional multiobjective approach, although they offer interesting and useful information to the designer. In order not to lose this valuable information, there must be taken into account not only the optimal solutions but also the nearly optimal ones.

Evolutionary Algorithms

Wiley(2001) found out that most recent studies focus on evolutionary algorithms (EA) which are a promising method for solving multi-objective optimization problems with conflicting objectives by approximating the Pareto solution set. EA is inspired by biological evolution, it uses biological mechanisms such as reproduction, mutation,

crossover, and selection. The main advantage is that EAs are metaheuristic algorithms, they do not make any assumption about the underlying fitness landscape. Therefore EA often performs well-approximating solutions to all types of problems in many diverse fields such as engineering, biology, economics, marketing, social sciences, and so on. (Wiley, 2001)

According to him, the most well-known EA is a Genetic Algorithm. GAs are commonly used to generate high-quality solution sets by relying on biological inspired operators such as mutation, crossover, and selection. Non-dominated Sorting Genetic Algorithm-II known as NSGA-II and Strength Pareto Evolutionary Algorithm also known as SPEA-2 are well-known variants of GA that have become GA standard approaches. The main drawback of EA is that it relies heavily on stochastic mechanisms, due to its random nature EA is highly unstable in general. Unstable here means under the same problem setting EA could give a very different performance result for each run.

Evolutionary Algorithms (EAs) were initially developed for unconstrained Single-Objective Optimization Problems (SOPs). However, extensive research has been conducted to

adapt them to other types of problems. In recent years, many Multi-Objective Evolutionary Algorithms (MOEAs) have been proposed in the literature to adapt EAs to dealing with Multi-Objective Optimization Problems (MOPs). One of the main components of most modern MOEAs is the ability to maintain genetic diversity within a population of individuals. Maintaining proper diversity is decisive for the behavior of EAs since a loss of diversity could lead to premature convergence, which is a frequent drawback, especially for single-objective optimization. Most MOEAs implicitly manage diversity by considering the objective function space and, in some cases, the decision variable space. Several mechanisms have been proposed in the literature to deal with the above, such as fitness sharing, clustering, and entropy, among others.

Mahrach et. al (2020) highlighted that promoting diversity is a key feature of an efficient and reliable MOEA. It is an intrinsic component in many MOEAs. Because of this, some authors have claimed that the application of MOEAs might be useful when dealing with single-objective problems. Furthermore, several theoretical and empirical studies have shown that multi-objective optimizers can even

provide better solutions than single-objective optimizers. Several mechanisms have been proposed in the literature to deal with the above, such as fitness sharing, clustering, and entropy, among others. Promoting diversity is a key feature of an efficient and reliable MOEA. It is an intrinsic component in many MOEAs. Because of this, some authors have claimed that the application of MOEAs might be useful when dealing with single-objective problems. Furthermore, several theoretical and empirical studies have shown that multi-objective optimizers can even provide better solutions than single-objective optimizers.

MOEAs have been applied to SOPs using various guidelines. Usually, the mechanisms proposed in the literature for solving SOPs employing MOEAs consist of transforming the original SOP into a MOP so that MOEAs can be applied to the transformed problem. This transformation can be done by either replacing the original objective with a set of new objectives or by adding new, additional objectives to the original one. Among these approaches, the best known and most widespread in the literature are: transforming constraints into objectives, considering diversity as an explicit objective function, and multi

objectivization schemes, which transform an SOP into a MOP by modifying its fitness landscape. In any case, these new objectives are included to promote the exploration of different regions, since multi-objective approaches try to simultaneously optimize several objectives. This might make it possible to escape from sub-optimal regions, thus providing a suitable balance between exploration and exploitation. The analysis presented lists the benefits of using additional objectives, named helper-objectives. The main ones are: avoiding stagnation in local optima and maintaining diversity within a population.

Many real-world multi-objective optimization problems are NP-Hard Problems. It is not feasible to use brute force search for solving those problems due to the huge amount of computation involved. It is too computationally costly to find an exact solution but sometimes a near-optimal solution is sufficient. In these cases, MOEAs provide good approximate solutions to problems that cannot be solved easily using other techniques. EA uses biological evolution-inspired mechanisms, such as mutation, crossover, and selection. MOEAs are stochastic search and optimization methods that lead a population of candidate solutions

toward the Pareto front through evolutionary mechanisms and biological operators.

Algorithm 1: Evolutionary Algorithm Pseudocode

1. t←0;
2. InitPopulation[P(t)];(Initializes the population)
3. EvalPopulation[P(t)];(Evaluates the population)
4. while not termination do
5. P(t) ← Variation[P(t)] (Creation of new individuals)
6. EvalPopulation[P(t)]; (Evaluates the new individuals)
7. P(t + 1) ← ApplyGeneticOperators[P(t)]; (Creation of next generation population)
8. t ← t + 1;
9. end while

Candidate solutions also called individuals in EA terms are a set of different decision variable vectors that are randomly generated initially. EAs use mutation and crossover operators. Later these solutions are evaluated by fitness functions which are objective functions defined by the optimization problem. And based on the quality or fitness values of solutions EAs uses a selection operator to choose the best non-dominated solution set to form a new population. This whole process iterates many times or generations until it satisfies the termination condition. MOEAs present many advantages over traditional multi-objective approaches: 1.) MOEAs attempts to search the whole Pareto front instead of one single Pareto optimal solution in each run. 2.) MOEAs do not require any domain

knowledge about the problem to be solved 3.) MOEAs do not make any assumptions about the Pareto curve.

Genetic Algorithm

Carvalho (1999) explained that Genetic Algorithms (GAs) are search and optimization algorithms based on the Theory of Evolution. These algorithms are based on natural selection and survival of the fittest as inexorable for the evolution of a population. Such algorithms codify one possible solution in data structures based on chromosome composition and apply evolutionary techniques such as mutation and crossover to reach an optimal solution to a problem.

```
1: determine objective function (OF)
2: assign number of generation to 0 (t=0)
3: randomly create individuals in initial population P(t)
4: evaluate individuals in population P(t) using OF
5: while termination criterion is not satisfied do
6:     t=t+1
7:     select the individuals to population P(t) from P(t-1)
8:     change individuals of P(t) using crossover and muta-
   tion
9:     evaluate individuals in population P(t) using OF
10: end while
11: return the best individual found during the evolution
```

Figure 5. Pseudocode of Genetic Algorithm

Tanomaru (1995) cites a GA that works by generating populations of individuals that adapt by using genetic operators and, after it, they select the best individuals

and remove the rest. Through this, a GA preserves the best genes to evolve to the next generation of the population until finding an optimal solution to the problem which is being treated. A GA is not like traditional search methods, it differs in the sense it: i) works with parameter coding; ii) uses populations; iii) uses evaluation function; iv) and uses probabilistic concepts instead of deterministic ones.

Linden (2012) in the later years proceeded to expound the main elements of a Genetic Algorithm which are: 1.) Gene represents one characteristic that has to change so that an optimal solution can be found. 2.) Individual or Chromosome, a data structure that codifies a possible solution to the problem. It can be represented as an array of any kind of element (Genes) in which each element represents a characteristic of the problem. The coding must be as simple as possible; it cannot have any representation that is not a possible solution to the problem. If the problem has conditions and limitations, they must be represented in the coding. 3.) Population - a set of chromosomes that will suffer mutation and crossover and will undergo selection to find the optimal solution to the

problem. 4.) Evaluation Function or Fitness Function - a function used to determine the quality of a chromosome, how close or not it is to the solution wished. It is used to discard individuals or choose them to survive in the selection step. 5.) Selection, a step of the algorithm that behaves as natural selection, in which the best chromosomes will reproduce and pass their genetic load on to the next phase or will be discarded. It must privilege the individuals with higher fitness and it must retain some of the individuals with lower fitness because even the least adapted individuals can have important genetic characteristics. 6.) Crossover, a genetic operator that will recombine chromosomes to create new individuals while maintaining and mixing their genetic characteristics. 7.) Mutation - a genetic operator that will change a random element of a chromosome to a random value by "flipping a coin". By doing this, it will explore new characteristics with a random chance. The genetic operators have a chance to happen which cannot be either too high or too low, because if they happen very frequently, the algorithm can become a random algorithm; but also, these operators set the speed at which the algorithm will achieve an optimal solution.

Tacklind and Ong (2016) discussed areas of multiple criteria decision making, where optimal decisions need to be taken in the presence of trade-offs between different objectives. EAs are very attractive for multi-objective analysis concerning classical methods. EAs beginsparameterset of solutions that are randomly generated and called the initial population. The offspring populations are generated by some operators such as the mutation, the crossover, and the selection.

They assessed the performance of the multi-objective evolutionary algorithms. The three algorithms have been coded in the mathematical software package MATLAB and run on a PC with 1GB of ram and Intel Core Duo 2 GHz CPU. For evaluating the performance and strength of these three optimization algorithms, five well-known mathematical test functions have been selected. These test functions have between 10 and 30 bounded design variables and 2 objective functions. The quality of the obtained solutions is assessed using performance measures such as the distance between the generated Pareto Front and the known optimal Pareto Front solutions, and the diversity of the solutions on the Pareto Front.

The results presented in this study indicate that there are clear differences in performance between Multi-Objective Algorithms as the complexity of the problems increases. There are strong indications that including hypervolume as a measurement for problems with 7 or more objectives would've generated more reliable results. The results indicate NSGA-III is better suited for more complex problems than the other algorithms tested.

Non-dominated Sorting Genetic Algorithm III

Deb and Jain (2014) explained that NSGA-III is an effective algorithm to deal with many-objective optimization problems. Its' basic framework remains similar to the original NSGA-II algorithm. The biggest change is using well-distributed reference points to maintain diversity. In addition to emphasizing the dominance relationship, the number of individuals associated with each reference point is also emphasized. When the reference points are evenly distributed over the entire hyper-plane, the resulting solutions may be evenly distributed and close to the PF.

The original NSGA-III study has been demonstrated to work well from three to 15- objective DTLZ and other

problems. A key aspect of NSGA-III is that it does not require any additional parameter. The method was also extended to handle constraints without introducing any new parameter. That study has also introduced a computationally fast approach by which the reference point set is adaptively updated on the fly based on the association status of each reference point over several generations. The algorithm is outlined in Figure 5.

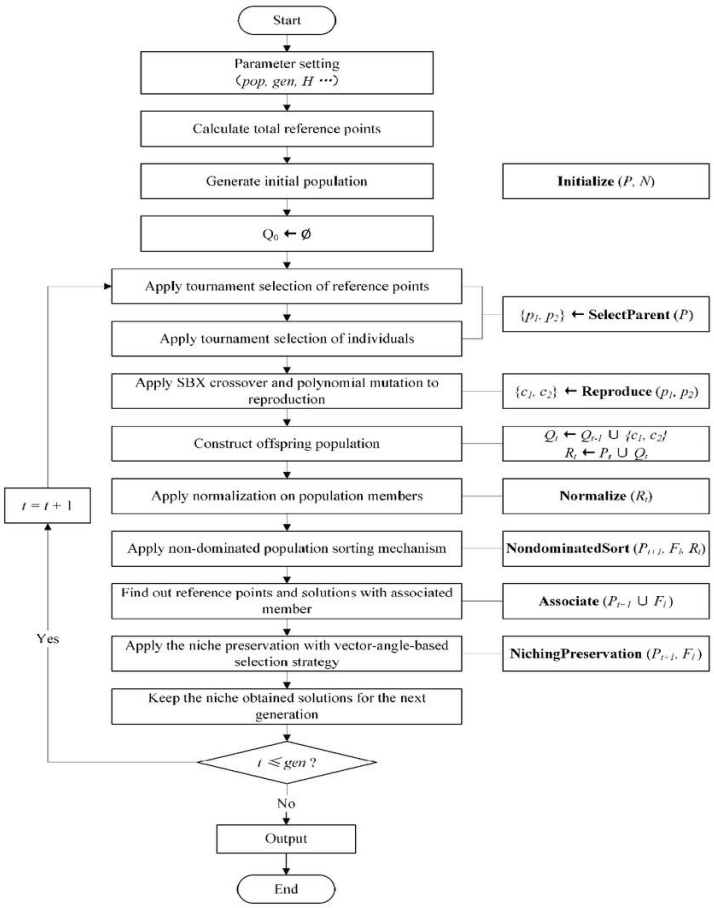


Figure 6. Flow Chart of NSGA-III Algorithm

Seada (2015) explains the process of NSGA-III, where it starts with a random population of size N and a set of widely-distributed prespecified M-dimensional reference points H on a unit hyper-plane having a normal vector of ones covering the entire RM + region. The hyper-plane is placed in a manner so that it intersects each objective axis at one. Das and Dennis's technique is used to placing H = M+p−1 preference points on the hyper-plane having (p + 1) points along each boundary. The population size N is chosen to be the smallest multiple of four greater than H, with the idea that for every reference point, one population member is expected to be found. At a generation t, the following operations are performed. First, the whole population Pt is classified into different non-domination levels, in the same way, it is done in NSGA-II, following the principle of non-dominated sorting. An offspring population Qt is created from Pt using usual recombination and mutation operators. Since only one population member is expected to be found for each reference point, there is no need for any selection operation in NSGA-III, as any selection operator will allow competition to be set among different reference points. A combined population Rt = Pt ∪

Qt is then formed. Thereafter, points starting from the first non-dominated front are selected for Pt+1 one at a time until all solutions from a complete front cannot be included. This procedure is also identical to that in NSGA-II.

SharpNEAT

Green (2012) explains that SharpNEAT is a kit of parts that facilitate research into evolutionary computation, and specifically evolution of neural networks. The code as released provides many example problem domains that demonstrate how the various parts can be wired together to produce a complete working EA. Alternatively, researchers can swap in alternative parts, such as a new/alternative genetic coding or even a new evolutionary algorithm. The provision for such rewiring was a major design goal of SharpNEAT and is facilitated by abstractions made in SharpNEAT's architecture around key concepts such as genome (genetic representation and coding) and evolutionary algorithm (mutations, recombination, selection strategy). A notable point is that NEAT and SharpNEAT search both neural network structure (network nodes and connectivity) and connection weights (inter-node connection strength). This

is distinct from algorithms such as back-propagation that generally attempt to discover good connection weights for a given structure.

An influential study by Aavas Garujel (2019) in Real-Time Strategy Game Micromanagement uses SharpNEAT implementation as an evolution module to adapt to the purpose of their research. This evaluation adapter is the mediator between evolutionary algorithms and the game which they implemented using sockets to be able to run the game simulations in parallel. It gets the configuration from the NEAT module and sets up the game, it also gets a neural network configuration from the evolution module and feeds inputs with the current game state into the network, and uses the output from the network to feed the game and move entities. The adapter returns the final fitness after running the simulation which ends when one of the players has no units left or after a set number of frames.

According to Stanley and Mikkulainen (2005), in competitive coevolution, individual fitness is evaluated through competition with other individuals in the population, rather than through an absolute fitness measure. In other words, fitness signifies only the relative strengths of solutions; an increased fitness in one

solution leads to a decreased fitness for another. Ideally, competing solutions will continually outdo one another, leading to an ”arms race” of increasingly better solutions (Dawkins & Krebs, 1979; Rosin, 1997; Van Valin, 1973). Competitive coevolution has traditionally been used in two kinds of problems. First, it can be used to evolve interactive behaviors that are difficult to evolve in terms of an absolute fitness function. For example, Sims (1994) evolved simulated 3D creatures that attempted to capture a ball before an opponent did, resulting in a variety of effective interactive strategies. Second, coevolution can be used to gain insight into the dynamics of game-theoretic problems.

Lindgren & Johansson (2001) coevolved iterated Prisoner’s Dilemma strategies to demonstrate how they correspond to stages in natural evolution. Complexification encourages continuing innovation by elaborating on existing solutions. To demonstrate the power of complexification in coevolution, NEAT is applied to the competitive robot control domain of Robot Duel. There is no known optimal strategy in the domain but there is substantial room to come up with increasingly sophisticated strategies. The main results were that (1) evolution did complexify when

possible, (2) complexification led to elaboration, and (3) significantly more sophisticated and successful strategies were evolved with complexification than without it. These results imply that complexification allows establishing a coevolutionary arm.

Bryant, Stanley, and Mikullainen stated a real-time version of NEAT (rtNEAT) was developed to allow users to interact with evolving agents. In rtNEAT, an entire population is simultaneously and asynchronously evaluated as it evolves. Using this method, it was possible to build a new kind of video game, NERO, where the characters adapt in real-time in response to the player's actions. In NERO, the player takes the role of a trainer and constructs training scenarios for a team of simulated robots.

According to Ontanon, Synnaeve, Uriarte, Richoux, Churchill, and Preuss (2012), in many RTS games, the game strategy can be roughly divided into two levels of abstraction. Macro management is the level that is concerned with high-level strategic decision-making such as resource planning and opponent modeling. Techniques dealing with micromanagement must choose and adapt sequences of strategic actions to meet goals of varied hierarchy. Examples of techniques applied to this domain include Case-

Based Reasoning and Goal-Driven Autonomy. Micromanagement is the level concerned with direct combat tactics and unit control. Traditionally, micromanagement is accomplished via static AI techniques such as scripts based on simple metrics. More complicated techniques in the literature include Reinforcement Learning (RL) and Evolutionary Algorithm approaches.

The NEAT and rtNEAT algorithms are guided by a fitness metric. In the context of SC: BW unit micromanagement, the fitness should reflect the performance of an individual unit. For both NEAT and RTNEAT, we define the fitness Fi for a unit I as: Fi = T DDI − HP Li IHPi + 1. The function takes in the total damage dealt by the unit (TDD), its hit point loss (HPL) accrued over the match, and its initial hit point (IHP). In theory, the fitness is only upper bound by the total hit points of enemy units. However in practice, the average fitness of each unit falls under [0, 2] where at its lowest the unit has produced no damage and dies, and at its highest value, it has dealt twice as much damage as it has taken.

Watson and Zen (2007) confirmed the viability of NEAT and rtNEAT algorithms in evolving agents for various SC: BW micromanagement scenarios. When the algorithms are allowed

to run non-stop, the win rate of agents against the default SC: BW AI fluctuates highly over generations. However, when evolution is halted upon reaching an acceptable level of performance, both algorithms canplace consistently generate winning agents, with most under 100 generations. Each algorithm differs in the variability of performance over different unit matchups. Factors contributing to the difference in performance include the complexity of the network starting topology and the variation in unit types. There is room to explore network complexity further, and a need to establish standardized evaluation methods for micromanagement agent evaluations. More work is needed to adapt these techniques for commercial RTS game deployment, but results here have shown promising performance in a learning AI capable of defeating scripted AI under short training time.

Ollesen, Yannakakis, and Hallam (2009) introduced the application of NEAT and rtNEAT to a real-time strategy game, Globulation 2 (G2), for dynamic game-challenge balancing. First, the researchers used NEAT to optimize the performance of G2 NPCs and investigate two proposed challenge factors through offline experiments in G2. Results verified that the factors of aggressiveness and

number of warriors contribute to the challenge since obtained neuro-evolved agents were able to outperform all standard AI NPCs available for playing the game. As a step further, a challenge rating formula was designed incorporating six proposed challenge factors, and rtNEAT was used to minimize the difference of challenge rating between the AI agent and its opponent in real-time. Results provide evidence for the effectiveness of real-time learning and suggest that rtNEAT can be successfully applied for dynamic game balancing in RTS games that share features with G2.

The successful application of rtNEAT in NERO was in a game environment with sufficient interaction and fixed initial conditions for the population. There are 50 soldiers (agents) used as a population in the NERO game and all of them are spawned at the same position. Offspring generated through evolution re-spawn at that same position too. On the other hand, due to G2 game design, there is a maximum of 7 agents to interact with one opponent, can place and offspring are placed in the position of the eliminated members of the population and inherit their game

position and current state. This constitutes a significant difficulty in using rtNEAT in a game such as G2.

In a study by Oh, Rho, Et Al. (2019) they made 1v1 battle AI agents for the commercial game "Blade & Soul". The trained agents competed against five professional "Blade & Soul" gamers and achieved a win rate of 62%. Their paper presents a practical reinforcement learning method that includes a novel self-play curriculum and data skipping techniques. Through the curriculum, three different styles of agents were created by reward shaping and were trained against each other. The paper also suggests data skipping techniques that could increase data efficiency and facilitate explorations in vast spaces. Their method can be generally applied to all two-player competitive games with vast action spaces.

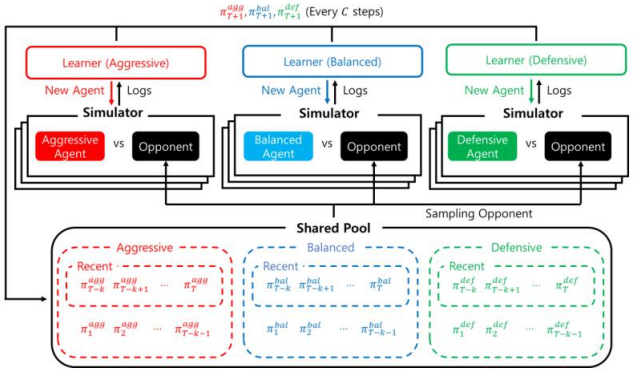


Figure 7. Overview of self-play curriculum with three different styles.

Figure 6 shows an overview of the proposed self-play curriculum with three different types of agents. Agents of each style have their learning process, and all three agent types were trained concurrently. Each learning process consisted of a learner and multiple simulators. The learner and the simulators work asynchronously.

Showalter and Schwartz (2020) conducted a study on neuromodulated multiobjective evolutionary neuro controllers without speciation wherein neuromodulation is a biologically-inspired technique that can adapt the per-connection learning rates of synaptic plasticity, it also has been used to facilitate unsupervised learning by adapting neural network weights, and Multiobjective evolution of neural network topology and weights has been used to design neuro controllers for autonomous robots.

CHAPTER 3 RESEARCH DESIGN AND METHODOLOGY

Description of the Proposed Study

NEAT (NeuroEvolution of Augmenting Topologies) is mostly used in reinforcement learning solely for a bot to learn from its own mistakes. The research revolves around this concept where instead of only learning from past experiences, it also makes the virtual bots learn from the competition.

Graphical assets are created for the simulation of the fights, the main parts are the bots themselves to be made which is how they would look like in the simulation, the playing field which is just one strip where the bots would be, and their weapons and armor. Alongside the fencer's assets is the addition of the whole arena, background design, on-screen data, etc. Graphical assets are created from a mixture of different assets from the Unity Asset Store and Mixamo.

Creating the environment starts with developing the arena for both fencers to fight in. After the environment is established is the addition of the fencers themselves as well as their equipment. Creating one GameObject for both the fencers would suffice since the simulation would just

be creating duplicates of the GameObject once the simulation starts.

Animations are also added and taken from the asset store. Some of which are animations for stabbing, movements, dodging, parrying, lunging, and other fencing movements. Animations are used by the algorithm in learning and developing strategies to optimize their movements. All animations are controlled bythe to, the Animator.

Graphs are added as well in the final build of the simulation to visualize the data generated by the fencers in real-time. Different displays are implemented for different graphs to swap screens in order to view different graphs of data. Graphs and Charts from the Unity Asset Store are used for data visualization and Grapher for the collection of those data for analysis.

The bots movements and actions are based on the combat sports of fencing with their actions depending on what they do would let them gain or subtract points from the fitness score, the bots are coded to do specific actions that are done in games of fencing that include: attacks, defense, leaps, etc. and when the bots are now properly doing the move sets they can now start to fight.

The researchers assume at first that the virtual fencers do not know the game of fencing at all. And that these bots would eventually learn from their mistakes and competition and slowly improve their skillset until it is optimized when reaching a certain fitness score.

Methods and Proposed Enhancements

The researchers would be improving the NEAT (NeuroEvolution of Augmenting Topologies) algorithm by implementing the concept of competition which would be called 'Competitive NEAT'.

Based on the classic NEAT algorithm, to evolve the virtual fencing bot, the algorithm contains a fitness function that computes a single real number indicating the quality of an individual genome: a better ability to solve the problem means a higher score. The algorithm progresses through a user-specified number of generations, with each generation being produced by reproduction (either sexual or asexual) and mutation of the fittest individuals of the previous generation.

The reproduction and mutation operations may add nodes and/or connections to genomes, so as the algorithm proceeds, genomes (and the neural networks they produce) may become more and more complex. When the preset number of

generations is reached, or when at least one individual (for a fitness criterion function of max; others are configurable) exceeds the user-specified fitness threshold, the algorithm terminates.

In this research, the researchers are implementing competition which means the reproduction and mutation operations that these genomes would undergo needs to account for the moves of its opponent compared to his and how he performed in that certain round. This introduces more criteria to the reproduction process which means that the genes filtered should prove to be stronger than the previous genes.

The researchers were able to render and visualize the fencers using the Unity Game Engine. In particular, Unity's Scriptable Render Pipeline was used to give researchers control over the graphics and lighting of the simulation scene. The researchers acquired the animations and environment assets from the Unity Asset Store and placed them in Unity's Scene Editor. The researchers used UnityNEAT to integrate the use of neural networks to control different physical 3D objects within SharpNEAT with pure C#, which allows for a quick reinforcement learning

mechanism for these. To use it, the UnityNEAT folder was downloaded and imported to the Unity project.

The scene is set up by adding an empty game object ("Evaluator") and attaching the Optimizer class to it. In the Optimizer script, they dragged their Unit prefab to the Unit slot. The unit is the thing being evaluated; two fencers. In this script, they can set the trial duration (how many seconds each unit is evaluated) and the number of trials, which is useful for randomness. Attach a script to the Unit prefab which extends UnitController instead of MonoBehaviour. SharpNEAT also has a built-in NeatUI script to display in the user interface the current generation and fitness score. The researchers also added to this script the scoreboard scoring display in the simulation scene and also the dynamic stamina bar.

Intermediate knowledge of the Unity game engine and the C# language is required to be able to alter the effectors, duration of the animation to its playing fields, and the moves being executed. Since the Unity game engine provides a 3D virtual environment, this system does not require the user to have expert knowledge of animation. The researchers only need 1 PC to run this project. Since the researchers are exporting the project file into an

executable file, the project can be run anywhere with a Windows 10 OS.

Our experimental setup has four main components, the NEAT evolution module, the Simulation adapter, the Multi-Objective Optimization Layer, and the Fencing Simulation. NEAT is concerned with running the evolution by assimilating the fitness results and generating networks. The researchers used the SharpNEAT implementation of NEAT by Colin Green for the evolution module. This simulation adapter is the mediator between the evolutionary algorithm, with the multi-objective criterion aspect, and the fencing combat simulation implemented through using sockets to be able to run the game simulations in parallel. It gets the configuration from the NEAT module and sets up the game. It also gets a neural network configuration from the evolution module and feeds inputs with the current game state into the network and uses the output from the network to feed the game and move entities. The researchers control the adapter and have the power to end or continue the simulation at a certain point. Once they decide to end the simulation after a set number of generations, fitness scores are produced, so thus two unique trained neural nets.

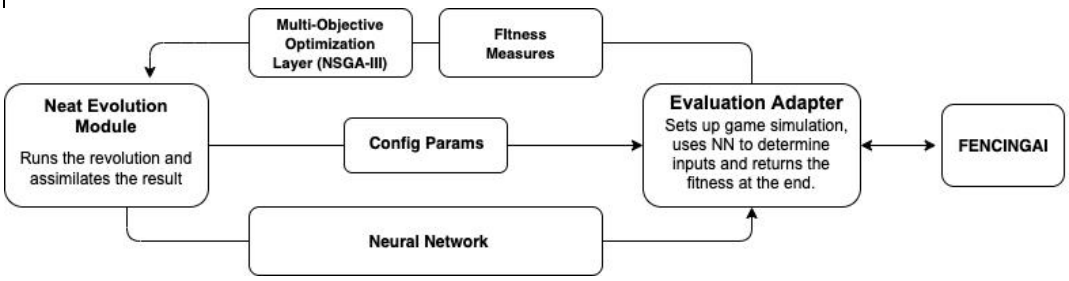


Figure 8. FencingAI Architecture Diagram

Classic NEAT	Competitive NEAT
1. Only one actor 2. Learning from himself 3. Fitness score only depends on internal influences	1. Two actors 2. Learning from its own & enemy 3. Fitness score depends on internal and external influences

Table 1. Comparison of Classic NEAT vs Competitive NEAT

In this table, you could see the comparison between the Classic NEAT and this research's Competitive NEAT. The main feature Competitive NEAT highlights a realistic-looking that it uses Two Actors during the simulation which means it would learn from itself and from its enemy which means that the fitness score of these actors depends on internal and external influences. This would not only make

for an interesting comparison but would also progress towards the improvement of future NEAT-like algorithms.

Assets and 3D Objects

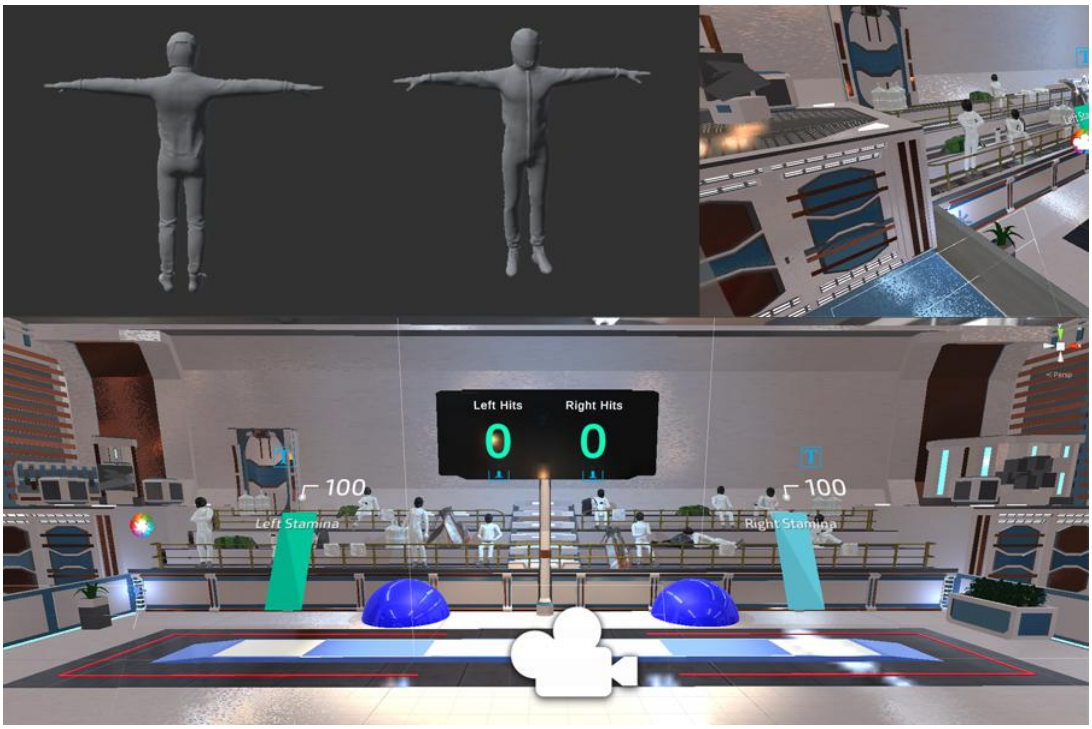


Figure 9. *Assets and 3D Objects*

For the Assets and 3D objects used in this research, the researchers used the already available ones from Mixamo and Unity Asset Store. This is to help for easy implementation and creation of the environment as well as realistic looking models for the simulation. The creation of the arena for the Fencers is under Fencing rules about the arena's looks and dimensions for accurate simulation of the games.

Input Layer

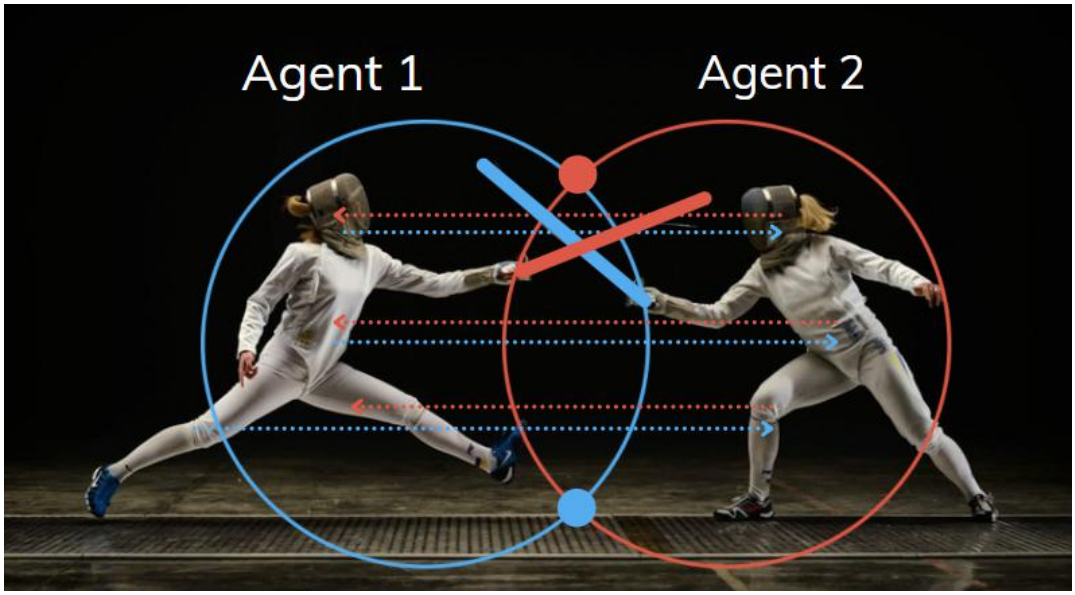


Figure 10. *Input Layer – Fencer Sensors*

Spatial detection methods for movements are used for the input layer. For this, SphereCast is used in detecting the fencer's opponent in a radius and RayCast is for detecting the opponent in a straight line. 5 sensors that are implemented for the fencers:

1. HeadSensor
2. ChestSensor
3. LegSensor
4. SpatialSensor
5. BladeSensor

Output Layer

Output layer is controlled by the Animation State Tree (Animator in Unity). Based on the sensory detection of the input layer, the fencers determine which optimal movements to perform.

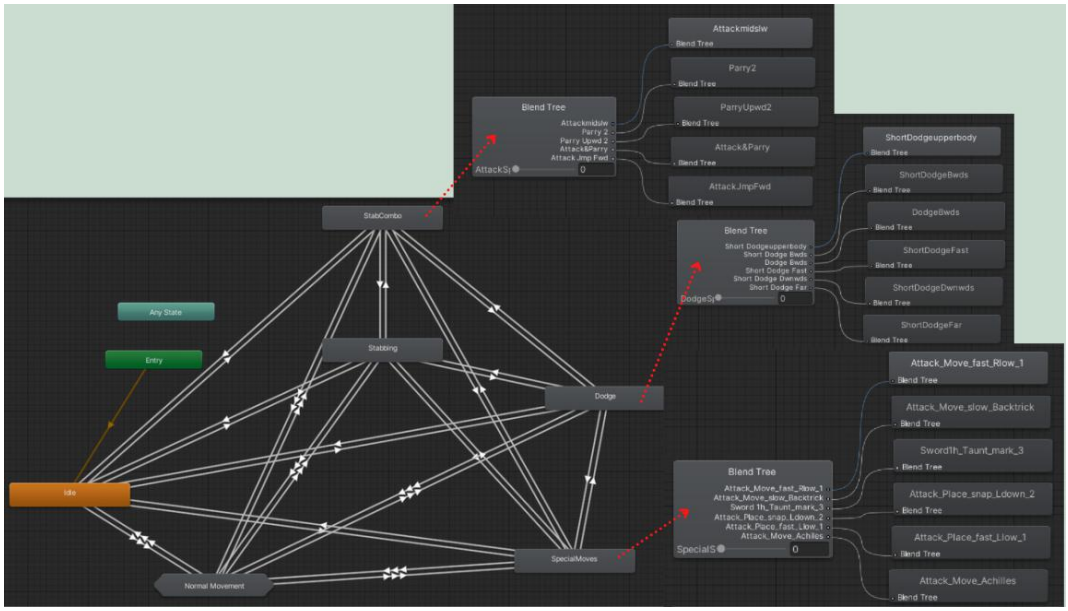


Figure 11. *Animation State Tree*

Figure 11 shows the Animation State Tree and all the movements. This includes state machines, blend trees, and sub-state machines of the fencer's movements. The controller has references to the movements it uses, and it uses a so-called State Machine, which may be thought of as a flowchart or a basic program written in Unity's visual programming language to manage the many animation states and transitions between them.

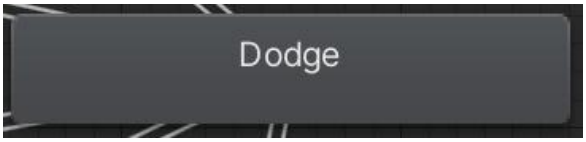


Figure 12. *State Machine*

Figure 12 are State machines that include the main movements like Normal (Pacing), Stabbing, Dodging, Combos, and some Special moves. The fencers have restrictions on the next state they can go to rather than being able to switch immediately from any state to any other. For example, an attack and parry can only be taken when the character is already attacking and not when it is at idle or dodging, so it should never switch straight from the idle state or dodging state to the attack and parry state.

Figure 13. (Unity) *Blend Tree*

Blend trees (Figure 13) specifically in Unity, allow for multiple similar movements to be blended smoothly. These are put under certain state machines and multiple similar Sub-state machines are connected through the Blend Trees rather than directly to the state machine. Examples of similar movements would be the Short Dodge Upper Body, Dodge Backwards, Short Dodge Far, etc. A blending parameter, which is only one of the numeric animation parameters connected with the Animator Controller, is used to regulate how much each of the motions contributes to the final effect. The motions that are blended must be similar and time for the blended motion to make sense. In an Animation State Machine, Blend Trees are a unique form of state.

Figure 14. *Sub-state Machine*

Under the Blend Trees are Sub-state machines (Figure 14) which include more specific and varying moves under the main ones like Attack and Parry and Attack Jump Forward under Combos and the same goes to other state machines. This allows the fencers to strategize movements and react accordingly to the movements done by their opponent. For example, when the opponent lunges, the fencer might choose to dodge, parry or counter-attack depending on what is optimal at that point in time. Rather than handle the entire action with a single state, it makes sense to identify the separate stages and use a separate state for each. For example, categorizing the move Short Dodge Moves under the Dodges state machine is more practical than making it as a state machine and making the whole state machine tree more complicated.

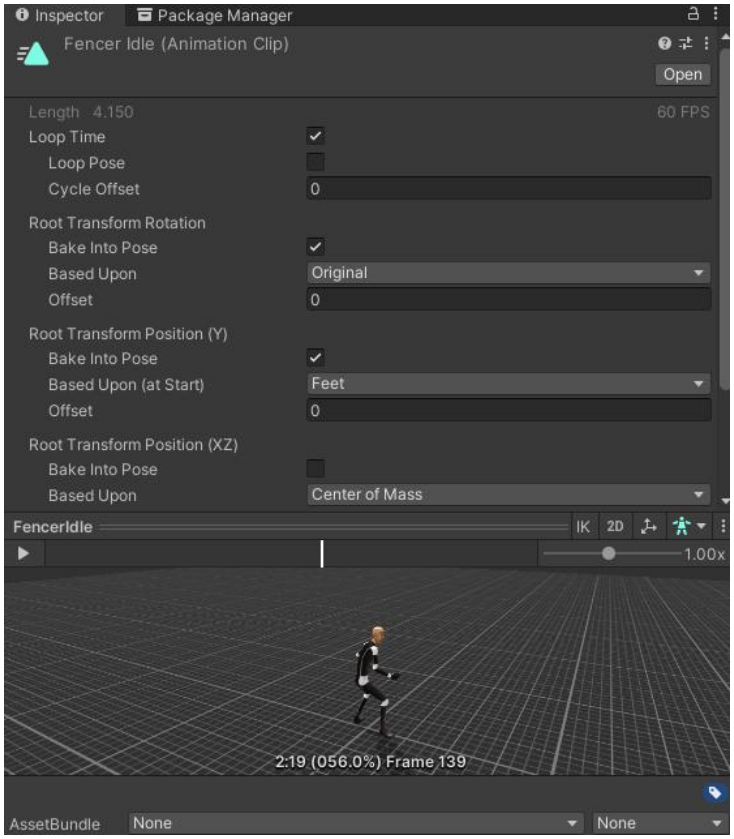


Figure 15. *Animation Properties Example*

Figure 15 displays the properties of animation. This set of settings include the looping of the animation, rotations, speed, preview of the animation, etc. The rigged character model (in this case, the Fencer) contains a unique set of bones that are transferred to Unity's standard Avatar format. As part of the imported character model from Mixamo, this mapping is saved as an Avatar asset.

Algorithm Procedures

1. The program has two virtual fencers with different starting configurations. SharpNEAT will initialize the NEAT training process. Starting moves or inputs from the game objects inside Unity Engine are randomly distributed.
2. The program is set on a 3D plane to make the NEAT's algorithm calculations more stable and to make the program more reliable.
3. These virtual fencers have 2 directional movements, left, right, and their attacking pattern.
4. These Fencers' main objective is to earn points via hitting their opponent in any part of their body. The fitness scores depend on how the fencers performed in each iteration. The fitness functions will depend on different micro objectives such as attacking, stamina, defense, and timing. With this the fencers have to create various solutions which differ per agent, with each iteration of rounds, competing solutions would continually outdo one another, leading to an "arms race" of increasingly better solutions.
5. In regards to the NEAT, the fitness scores would then be the basis whether to reproduce which virtual fencer

deserves to go on. With the application of NSGA-III, the program will map all the fitness scores in a Pareto front, and choose the most optimal result or non-dominant solution for the objectives being assessed. The usual NEAT will just choose the most dominant in each generation but with the algorithm combined with Competitive NEAT and NSGA-III, the past generations are also considered by ranking them on how assertive they are. There will be a time where the ones at the top of the hierarchy will fight with the lower ranks. This broadens the neural net's knowledge and flexibility of the neural net as they will be able to adjust in any kind of situation.

6. The fitness scores produced will be passed to a selection procedure that plays an important role in evolutionary multi-objective optimization through QD measure that promotes finding solutions of high quality as well as maintaining diversity.

7. The SharpNEAT module program will interact with Unity Objects, which will lead to showing the progress of both virtual fencers through generations to see their

gradual progression real-time as they optimize their movements and improve their skill set.

The researchers are using the NEAT algorithm, which is a genetic algorithm designed to efficiently evolve artificial neural network topologies, since it is a genetic algorithm, this stimulates overall the process of natural selection to solve a vast variety of optimization problems, especially those with a non-differentiable or stochastic objective function. NEAT uses terms such as:

Fitness Function

Fitness Score

Population

Chromosome

Gene

Generation

Selection

Crossover

Mutation

(These terms are defined in Chapter 1)

So in the case of the research, we define the *fitness function* as a function that takes the solution, which is the most efficient and best action taken by the AI fencers as input and generates a *fitness score* as output to

evaluate the quality of each solution. At the beginning of the iteration, we randomly generate four solutions (*population*) as our first *generation*. For each solution (*chromosome*), we indicate whether an action by the fencer is good ("1") or not ("0"), representing a *gene*. Each practice's proposed solution has a *fitness score*. The top-scored solutions are more likely to be chosen as parents (*selection*). The two selected *chromosomes* exchange some of their genes based on crossover points (*crossover*). However, there is a small chance that some genes of the new offspring will change (*mutation*). Finally, a new generation is born. The loop will continue until termination requirements are met.

Fitness Equation and Functions

$$\gamma = -\ Euler's\ Constant\ (Growth\ Rate) - 0.57721$$

$$LF = Left\ Fencer$$

$$RF = Right\ Fencer$$

$FF_{LF} = Fitness\ Function\ Left\ Fencer$

$$FF_{RF} = Fitness\ Function\ Right\ Fencer$$

$currHits = current\ hits\ landed\ by\ the\ Fencer$

$atmHits = current\ attempted\ hits\ by\ the\ Fencer$

$allHits = all\ hits\ landed\ by\ the\ Fencer\ through\ generations$

$dodge = dodges\ performed\ by\ the\ Fencer$

$zone = fencer's\ side\ of\ the\ mat$

$(LF\_allHits, RF\_allHits, LF\_allDodges, RF\_allDodges, LF\_allAZone, RF\_allAZone, LF\_allDZone, RF\_allDZone \neq 0)$

Left Fencer's Functions

$$LFRF_{currHits} = (LF\_Hits - RF\_Hits)$$
$$LF_{atmHits} = (LF\_attemptedHits/LF\_allHits)$$
$$LFRF_{allHits} = (LF\_allHits - RF\_allHits)$$
$$RF_{dodge} = (RF\_Dodges/RF\_allDodges) + RF\_Dodges)$$
$$LF_{zone} = (LF\_currAZone/LF\_allAZone) - (RF\_currAZone/RF\_allAZone) + (LF\_currDZone/LF\_allDZone)$$

$$LF\ _{stamina} = LF_{zone} * \gamma$$

Right Fencer's Functions

$$RFLF_{currHits} = (RF\_Hits - LF\_Hits)$$
$$RF_{atmHits} = (RF\_attemptedHits/RF\_allHits)$$
$$RFLF_{allHits} = (RF\_allHits - LF\_allHits)$$
$$LF_{dodge} = (LF\_Dodges/LF\_allDodges) + LF\_Dodges)$$
$$RF_{zone} = (RF\_currAZone/RF\_allAZone) - (LF\_currAZone/LF\_allAZone) + (RF\_currDZone/RF\_allDZone)$$

$$RF\ _{stamina} = RF_{zone} * \gamma$$

Fencer's Fitness Functions

$$FF_{LF} = |(LFRF_{currHits} + LF\_allHits + LF_{atmHits} + LFRF_{allHits} - RF_{dodge} + LF_{zone}) + LF\ _{stamina} * \gamma|$$
$$FF_{RF} = |(RF\,LF_{currHits} + RF\_allHits + RF_{atmHits} + RFLF_{allHits} - LF_{dodge} + RF_{zone}) + RF\ _{stamina} * \gamma|$$

The functions indicated by the researchers are shorthand to represent the code and to measure the fencer's fitness.

The $LFRF_{currHits}$ & $RFLF_{currHits}$ represents the current hits landed by the Fencer on the current generation.

In this equation, $LF\_Hits - RF\_Hits$ Left Hits is subtracted by Right Hits in order to get a value that would represent the Left Fencer's hits compared to the Right Fencer's Hits. The same is for the Right Fencer, but just the opposite $(RF\_Hits - LF\_Hits)$ considering we are comparing the two.

The $LF_{atmHits}$ & $RF_{atmHits}$ represents the attempted hits that the Fencer did on the current generation over all hits landed in its lifetime, this also represents the accuracy of the fencer during that generation.

In this equation, $LF\_attemptedHits/LF\_allHits$ Left Attempted hits are divided by the Left All Hits to get a ratio of the attempted hits at a current generation compared to all the hits in all generations. Similarly, the Right Fencer uses the same equation but uses its own Right Fencer's counterpart values $(RF\_attemptedHits\ /\ RF\_allHits)$.

The $LFRF_{allHits}$ & $RFLF_{allHits}$ represents all hits landed by the Fencer through generations compared to the other fencer.

In this equation, $(LF_allHits - RF_allHits)$Left Fencer's All Hits is being subtracted by Right Fencer's All Hits to get a value that would represent their difference from one another. Similarly, stimulates overall the Right Fencer

uses the same equation but just the opposite $(RF\_allHits - LF\_allHits)$ considering we are comparing the two.

The RF_{dodge} & LF_{dodge} is the dodges performed by the fencer and compares its current generation's dodge to all of its dodges in its lifetime.

In this equation, $(RF\_Dodges/RF\_allDodges) + RF\_Dodges)$ The Right Fencer's Current Dodges is being divided by Right Fencer's All Dodges to get a ratio of the current dodges compared to all dodges in all generations. This value is then added to the current dodges to reinforce this result. Similarly, the Left Fencer uses the same equation but uses its own Left Fencer's counterpart values. $(LF\_Dodges/LF\_allDodges) + LF\_Dodges)$

The LF_{zone} & RF_{zone} represents how much time a fencer spends on the attack zone compared to its other fencer's attack zone and how much stays on defense.

In this equation,

$$(LF\_currAZone/LF\_allAZone) - (RF\_currAZone/RF\_allAZone) + (LF\_currDZone/LF\_allDZone)$$

The Left Fencer's Current Attack Zone is being divided by Left Fencer's All Attack Zone to get a ratio of its current attack zone compared to all of its attack zones in all

generations. This ratio is then subtracted by the ratio of the opposite fencer's attack zones $(RF\_currAZone/RF\_allAZone)$ to compare each other's attack zones, and this value is then added to the ratio of the Left Fencer's Defense $(LF\_currDZone/LF\_allDZone)$ to reinforce this result. Similarly, the Right Fencer uses the same equation but uses its own Right Fencer's counterpart values. $(RF\_currAZone/RF\_allAZone) - (LF\_currAZone/LF\_allAZone) + (RF\_currDZone/RF\_allDZone)$

The $LF_{stamina}$ & $RF_{stamina}$ represents the stamina modifier that is added to the base stamina of the Fencers which is 100 and this value changes through generations. This stamina modifier represents how much stamina does the fencer need to adapt whether it attacks or defends more.

In this equation, $LF_{zone} * \gamma$ The Left Fencer's Zone is multiplied by Euler's constant to reflect the natural growth rate of the fencer and as well as being a multiplier to the Left Fencer's Zone. Similarly, the Right Fencer uses the same equation but uses its own Right Fencer's counterpart values. $RF_{zone} * \gamma$

All of the equations above is then compiled into their respective Fencer's Fitness Equation, for example for the

Left Fencer: $FF_{LF} = |(LFRF_{currHits} + LF\_allHits + LF_{atmHits} + LFRF_{allHits} - RF_{dodge} + LF_{zone}) + LF_{stamina} * \gamma|$

And for the Right Fencer:

$$FF_{RF} = |(RF\,LF_{currHits} + RF\_allHits + RF_{atmHits} + RFLF_{allHits} - LF_{dodge} + RF_{zone}) + RF_{stamina} * \gamma|$$

The $LFRF_{currHits}, LF\_allHits, LF_{atmHits}, LFRF_{allHits}, LF_{zone}$ are being added to each other to get a value that would represent them, and it is being subtracted by RF_{dodge} to reflect the Left Fencer learning from the dodges of the Right Fencer. Once computed, the Left Fencer's Staminda modifier is added to the value and multiplied by the growth rate. The resulting value needs to be an absolute value so that we get the magnitude of the fitness score, irrespective of sign. The Right Fencer also follows a similar approach but uses the opposite of the Left Fencer's equations and values.

Opponent/Behavioral Modelling

Opponent Modelling, specifically, Behavioral Modelling is an integral part to highlight the humanness factor of the study. The researchers proposed 4 behavioral models in order to determine what kind of characteristics the neural nets have, depending on the data found. Using these factors:

Category	Value
Generation	150
Left Fitness	157.8500607
Right Fitness	164.462824375
Left Stamina	107.6013438
Right Stamina	104.9686625
All Left Hits	160.125
All Right Hits	162.375
All Left Dodges	19.4375
All Right Dodges	20.125
All Left Zone Left Fencer (Left Defense)	3802.5625
All Left Zone Right Fencer (Right Attack)	98.5625
All Right Zone Left Fencer (Left Attack)	173.875
All Right Zone Right Fencer (Right Defense)	3587.5

Table 2. 150th Generation Mean Data Table

Aggressive. Based on the table, when a model's value goes over the given value of *All Left Zone Right Fencer (Right Attack)* and *All Right Zone Left Fencer (Left Attack)*, this indicates that the model is aggressive.

Balanced/Stable. Based on the table, when a model's value balances between all the four (4) zone values, this indicates that the model is stable.

Defensive. Based on the table, when a model's value goes over the given value of *All Left Zone Left Fencer (Left Fencer)* and *All Right Zone Right Fencer (Right Attack)* this indicates that the model is defensive.

Presumptuous. Presumptuous models are the ones that go over a specific value and stick with this strategy even though it results in poor performance and low fitness scores.

CHAPTER 4 RESULTS AND DISCUSSION

For this section, refer to the appendixes for the link to the videos of the Fencing Simulation in action.

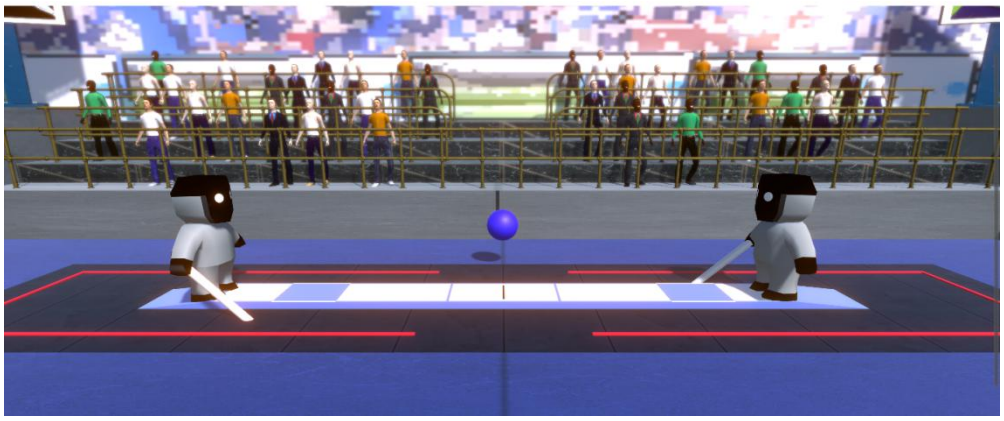


Figure 16. *The first iteration of Fencing Simulation featuring mini cuboid fencers*

During the infancy stages of the development, the researchers used simple cuboid models for the fencers as well as a draft of the background with plain assets. This aided in the understanding and further development of how to approach the animations and controllers for the fencers. The first draft of the background gave the researchers ideas on what the final iteration of the background should look like.

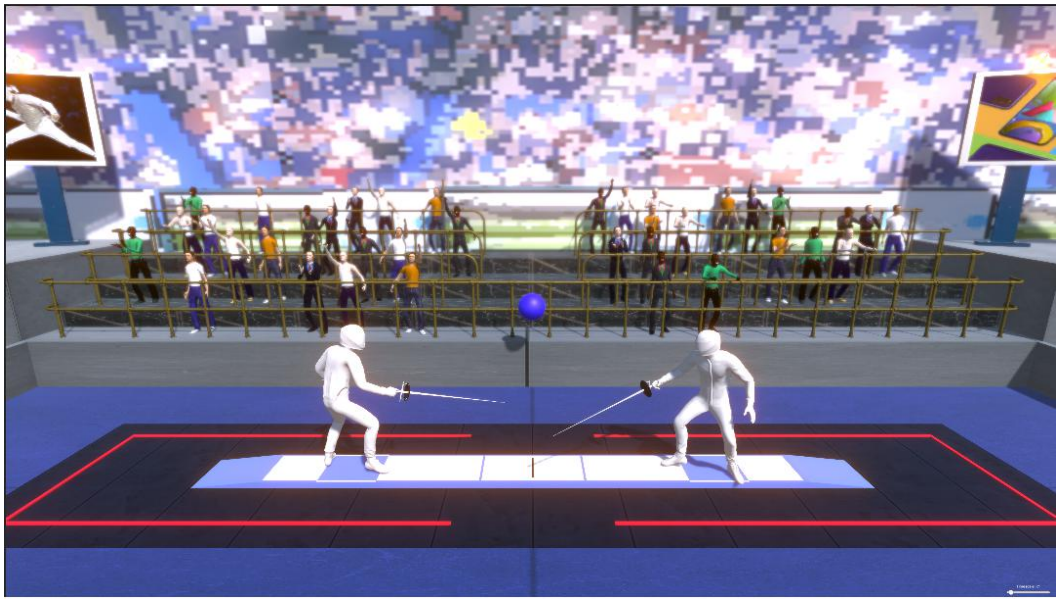


Figure 17. *The second iteration of Fencing Simulation with correct humanoid models and the blade*

The addition of proper models with accurate structure simulates real fencer movements. The correct blades also helped in accurate demonstration of hitting and parrying. The fencing strip (piste) was also widened to accommodate the fencers.

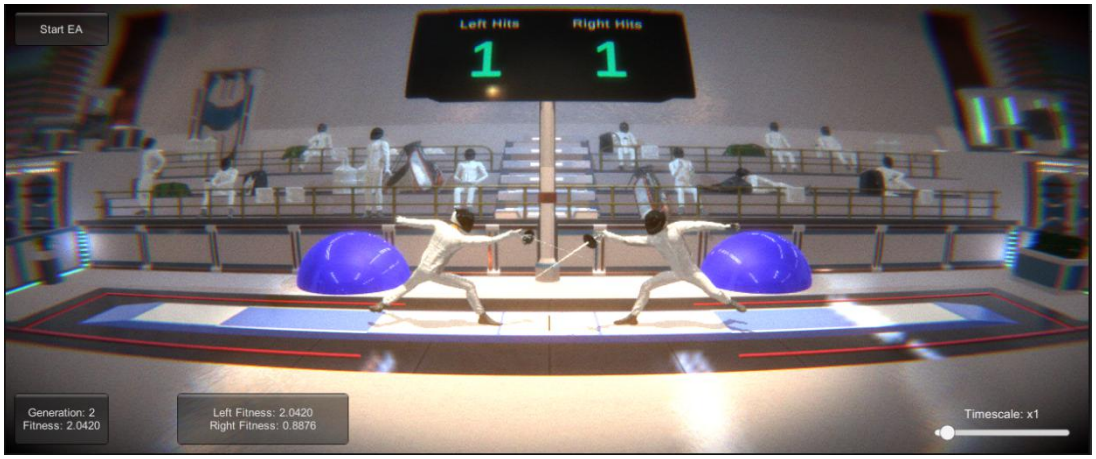


Figure 18. *The third iteration of Fencing Simulation with improved background & lighting and textured models*

In this stage of the development, further improvements in the backgrounds were added as well as proper textures were put to all the assets and models. Plus, proper lighting is added. HDR was also used to implement a more realistic look to the overall scene. Additionally, scoring is also shown in the game scene itself.

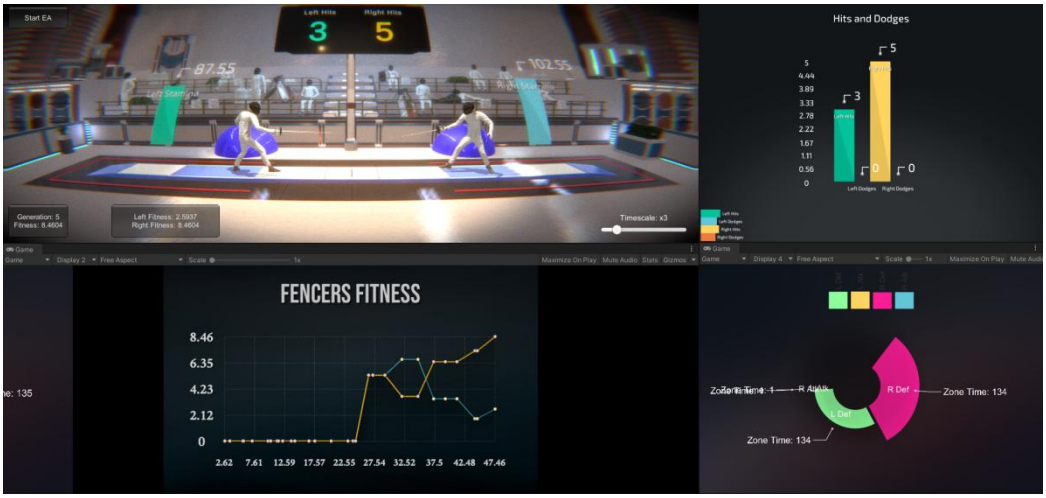


Figure 19. *The fourth iteration of Fencing Simulation with added graphs for data simulation as well as the stamina mechanics*

After the final look of the game, scene has been established, graphs were added for the data simulation. A line graph was implemented to present each fencer's fitness score. The bar graph on the other hand is to show the hits landed and dodges performed by both fencers. Also, the pie chart is to present the fencers total time spent on each zone (left and right zones). These graphs helped the researchers to visualize and analyze the data of both fencers to determine changes and patterns in a fencer's behavior. And lastly, the stamina mechanics were added for the realism of fencers's optimizing the usage of their movements.

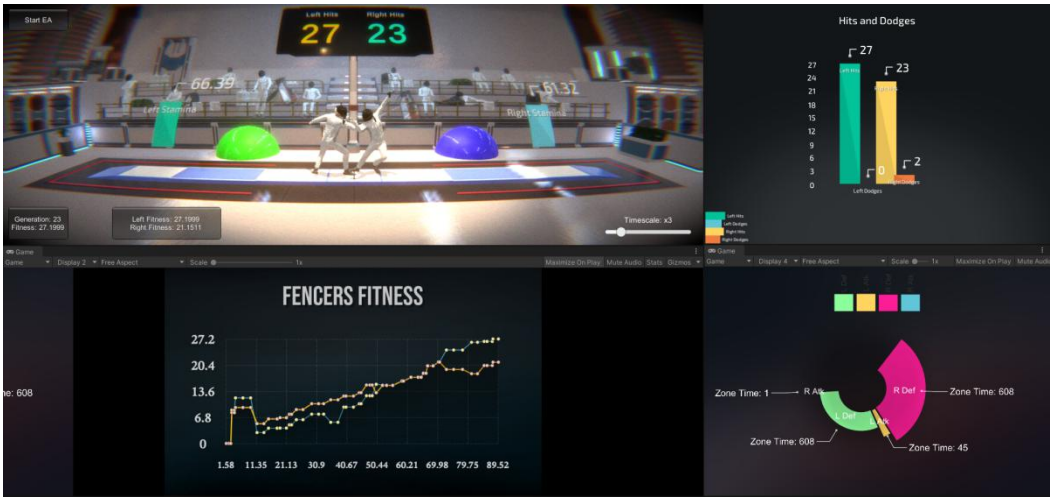


Figure 20. *(early generations)*

In the early stages of the fencers, these generations tend to perform hits more than they do dodges. Based on the fitness function, the early generation's primary objective was to land a hit in order to score (top left corner of figure 19). Since both fencers just want to land a hit, the line graph (top right corner of figure 19) shows that the fencers interchange in fitness scores since the first to land a hit gets a point. The bar graph (lower-left corner of figure 19) shows the hits landed by both fencers as well as the dodges they perform. The pie chart (lower right corner of figure 19) on the other hand shows the amount of time the fencers spend on each zone. L Def/R Defis the time spent by the left fencer in the left zone. This means that the fencer spends most of its time in defense mode and is

hesitant to charge. L Atk/R Atk is the time spent by the fencers in the opposing zone. This states that the fencer performs more attacks and charges to its opponent.

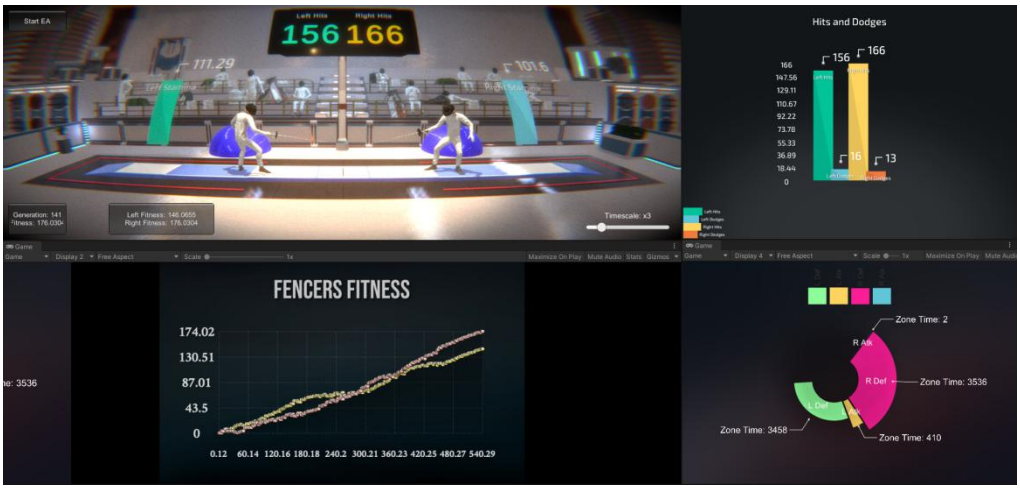


Figure 21. *(later generations)*

On the flip side, the later stages show improvements in strategies involving both attacks and dodges. The fitness scores of both fencers show a steady increase as they evolve and develop optimal strategies. At the same time, we can see a slight improvement in the usage of dodges in further generations. And lastly, the pie chart indicates that as the generations evolve, the fencers tend to be more defensive and stick to their respective zones.

Compared to the early generations, the later ones perform more methodically by using both attacks and dodges. The line graph on later generations shows a linear development in their fitness scores in contrast with the earlier generations. The fencer's hits and dodges show a relatively similar result with both the early and later generations but with a slight increase in the utilization of dodges. And based on both the early and later generations, the pie chart shows similar behavior where fencers spend more time in their respective zones in defense mode rather than being aggressive and charging to the opposite zones. In addition to this, we have observed in our numerous runs that the fencer that develops a good strategy based on its earlier generations, tends to snowball and dominate the other fencer.

This interpretation and analysis of results are based specifically on this single run of the program since every time they start the program, it yields a different result in every run. This means that every generation in every run is different and strategies they may develop can vary from each generation and each run of the program. The evolution simulated here mimics the one present in nature where the same organisms yield different evolutionary traits based on

their environment and current circumstances as well as their own decisions.

For this section refer to Appendix D for the graphs.

In Graphs 1 & 2. First, it shows that the Left Fencer (LF) and Right Fencer (RF) are tied within the first few generations around 1st to 12th Generations for fitness score. Then the LF takes over around the 15th Generation with its strategy to be as aggressive as possible by getting enough hits over the RF. This forces the RF to be more defensive and stay in its zone until it started to adapt and overcome the LF attacks and decided to attack more and overtook the LF around the 50th Generation and stayed dominant until the generational cut.

In Graph 4 ,5, 6, & 7. It shows the number of times the Left Fencer and the Right Fencer were both on the defensive and the attack. Within the first few generations around 1st to the 5th, the RF goes for an attack but was also fought back by the LF which was also on the attack forcing the RF to be on the Defense This changed at around the 50th generation where the RF was able to adapt and optimize its approach on doing its actions and learned how to beat the LF in the long run by utilizing defense and offense.

In Graphs 3 & 8. Since Left Fencer (LF) spends more time in the attack zone, it uses more stamina and needs to have more stamina to get in more attacks. This can also be LF's downfall when it loses too much stamina from doing too many moves. The same goes for the Right Fencer when it comes to dodging said attacks to defend itself from the LF, since dodging also uses stamina when it runs out of stamina it becomes an easy target for the LF to hit it.

In Graphs 9 & 10. Serves as visual markers showing the rankings of each run for the Fitness Score of the Left Fencer, simplified within the table below:

Rank	Run #	Fitness Score
1	12	307.0385
2	15	258.0473
3	5	228.0144
4	11	219.0013
5	4	215.0444
6	1	198.0402
7	10	185.0282
8	13	183.0357
9	2	158.0511
10	3	145.0799

11	8	131.0291
12	16	91.04302
13	7	89.02948
14	9	66.01073
15	14	48.03123
16	6	4.076411

Table 3. Ranking for the different Left Fencers in each run

In Graphs 11 & 12. Serves as visual markers showing the rankings of each run for the Fitness Score of the Right Fencer, simplified within the table below:

Rank	Run #	Fitness Score
1	9	309.0172
2	14	287.0493
3	16	261.0338
4	6	259.9009
5	7	212.014
6	8	194.0386
7	3	184.0746
8	13	174.0361
9	1	133.0135
10	4	123.0209
11	12	118.0178

12	2	111.0561
13	11	81.08385
14	10	79.03452
15	15	63.03846
16	5	41.97556

Table 4. Ranking for the different Right Fencers in each run

Fencers	Run/s fencer/s that Won	Run/s fencer/s that Lost
Defensive Left:	2, 4 & 13	3 & 14
Offensive Left:	1, 4, 12 & 15	7
Defensive Right:	6 & 14	5, 13 & 15
Offensive Right:	6, 8, 9 & 14	11

Table 5. Fencers in Winning & Losing on Defense or Offense

Purpose/ Description of Table 5

Table 5 serves as results taken from the data from each run. The data in the table were then checked if they

were in the threshold of being "Aggressive" and "Defensive" via the 150th Generation Mean Table in Chapter 3.

Top	Defensive Left Fencer	Offensive Left Fencer	Defensive Right Fencer	Offensive Right Fencer
1	**Run 13 (W)** (4232 Ticks, 183.0357 FS)	**Run 7 (L)** (459 Ticks, 89.02948 FS)	**Run 13 (L)** (4245 Ticks, 174.0361 FS)	**Run 14 (W)** (324 Ticks, 287.0493 FS)
2	**Run 2 (W)** (3922 Ticks, 158.0511 FS)	**Run 12 (W)** (406 Ticks, 307.0385 FS)	**Run 5 (L)** (4089 Ticks, 41.97556 FS)	**Run 9 (W)** (299 Ticks, 309.0172 FS)
3	**Run 4 (W)** (3901 Ticks, 215.0444 FS)	**Run 4 (W)** (387 Ticks, 215.0444 FS)	**Run 14 (W)** (3829 Ticks, 287.0493 FS)	**Run 6 (W)** (259 Ticks, 259.9009 FS)
4	**Run 3 (L)** (3894 Ticks, 145.0799 FS)	**Run 15 (W)** (307 Ticks, 258.0473 FS)	**Run 6 (W)** (3799 Ticks, 259.9009 FS)	**Run 8 (W)** (210 Ticks, 194.0386 FS)
5	**Run 14 (L)** (3865 Ticks, 48.03123 FS)	**Run 1 (W)** (277 Ticks, 198.0402 FS)	**Run 15 (L)** (3794 Ticks, 63.03846 FS)	**Run 11 (L)** (148 Ticks, 81.08385 FS)

Table 6. Top 5 Fencers in terms of Defense and Offense

Purpose/ Description of Table 6

Table 6 serves as results taken from the data from each run, ranked by the 5 highest in terms of Ticks and then shown with their corresponding Fitness Scores. The data in the table were then checked if they were in the threshold of being "Aggressive" and "Defensive" via the 150th Generation Mean Table in Chapter 3.

Run	**Results**
1	Left Fencer won as being Offensive (5th place) by 277 ticks on the right area. Hits: 176 - 155 and FScore: 198 - 133.
2	Left Fencer won as being Defensive (2nd Place) by 3922 ticks on the left area. Hits: 142 - 127 and FScore: 158 - 111.
3	Left Fencer lost as being Defensive (4th Place) by 3894 ticks on the left area. Hits: 158 - 171 and FScore: 145 - 184.
4	Left Fencer won both as being Offensive (3rd Place) by 387 ticks on the right area and as being Defensive (3rd Place) by 3901 ticks on the left area. Hits: 184 - 154 and FScore: 215 - 123.
5	Right, Fencer lost as being Defensive (2nd Place) by 4089 ticks on the right area. Hits: 167 - 105

	and FScore: 228 - 41.
6	Right Fencer won as being Offensive (3rd Place) by 259 ticks and as Defensive (4th Place) by 3799 ticks against Left Fencer with it trying to be Defensive but got bullied hard by the Right Fencer. Hits: 90 - 176 and FScore: 4 - 259.
7	Left Fencer was the most Offensive (1st Place) by 459 ticks on the right area but lost to Right Fencer who was on the defensive. Hits: 130-171 and FScore: 89 - 212.
8	Right Fencer won as being Offensive (4th Place) by 210 ticks on the left area. Hits: 152 - 173 and FScore: 131 - 194.
9	Right Fencer won as being Offensive (2nd Place) by 299 ticks on the left area. Hits: 147 - 228 and FScore: 66 - 309.
10	Left Fencer was the winner, being offensive by 64 ticks on the Right zone. Hits: 150 - 114 and FScore: 185 - 79.
11	Right Fencer lost being Offensive (5th place) by 148 ticks on left area. Hits: 173 - 127 and FScore: 219 - 81.
12	Left Fencer won as being Offensive (2nd Place) by 406 ticks on right area. Hits: 244 - 181 and FScore: 307 - 118.

13	Left Fencer won as being the most Defensive (1st place) by 4232 ticks on left area against Right Fencer that was also the most Defensive (1st Place) by 4245 ticks on right area. Hits: 180 - 177 and FScore: 183 - 174.
14	Left Fencer lost being Defensive (5th place) by 3865 ticks on left area to Right Fencer that is the most Offensive (1st Place) by 324 ticks on left area and Defensive (3rd Place) by 3289 ticks on right area. Hits: 128 - 207 and FScore: 48 - 287.
15	Left Fencer won as being Offensive (4th Place) by 307 ticks on right area and Right Fencer lost as being Defensive (5th Place) by 3794 ticks on right area. Hits: 193 - 128 and FScore: 258 - 63.
16	Right Fencer won as being Offensive over Left Fencer by 146 ticks on the left area. Hits: 148 - 204 and FScore: 91 - 261.

Table 7. Runs - Descriptive Results (150th Generation)

Behavioral Models

Behavior:	Win	Loss
Aggressive	Run 9 Right Fencer	Run 7 Left Fencer
Defensive	Run 2 Left Fencer	Run 6 Right Fencer
Balanced	Run 13 Left Fencer	Run 11 Right Fencer

Table 8. The behavior of Models with Wins and Loss (150th Generation)

Aggressive/Offensive Model

Run 9 Right fencer is the model that won using aggressive behavior having 299 ticks on the left zone with a fitness score of 309 against the Left fencer's 66.

Run 7 Left fencer is the model that lost using an offensive behavior having 459 ticks on the right zone with a fitness score of 89 against the Right fencer's 212 fitness score and more hits over the Left Fencer by 171.

Defensive/Cautious Model

Run 2 Left fencer is the model that won using a defensive strategy with 3922 ticks on its respective zone and a fitness score of 158 against right fencer's 111.

Run 6 Left fencer is the model that lost with a defensive strategy with 4089 ticks on its respective zone

with a fitness score of 4 against Left fencer's 259 Fitness score.

Balanced/Stable Model results

Run 13 Left Fencer shows a balanced result with the model winning by being high on the defensive with 4232 ticks on its respective zone also high on the offensive by 272 ticks against the Right Fencer.

Run 11 Right Fencer also shows a balanced result with high defense by being on its area for 3790 ticks and also by going on the offensive by 148 ticks but lost to the Left Fencer of this Run due to not being to put in more hits than the Left by 172 - 127 and their compared fitness scores to 219 - 81.

Presumptuous Models results

Run 7 Left fencer is a model that is presumptuous in being offensive that even though this strategy causes poor performance, the left fencer still sticks to it and eventually results in it losing the whole run this model ended with a fitness score of only 89 against the Right Fencer's 212.

Run 6 Left fencer is a model that is presumptuous in being defensive resulting in poor performance and with the

lowest fitness score out of every fencer in every Run with a fitness score of 4 against the right fencer's 259.

Run 11 Right Fencer is a model that is balanced with high defense by being on its area for 3790 ticks and also by going on the offensive by 148 ticks but lost when it did not change up its strategy to the Left Fencer of this run who is also balanced but due to not being to put in more hits than the Left by 172 - 127 and their compared fitness scores to 219 - 81.

CHAPTER 5 SUMMARY, CONCLUSIONS, AND RECOMMENDATIONS

Summary of the Proposed Study Design and Implementation

The researchers used fencing, specifically the rules and gameplay of epee, as a platform to perform the NEAT Algorithm. Using SharpNEAT, the C# version of NEAT, and uNEATY, the mediator to make NEAT's objects and rendering of images and animation of the Neural Network work in Unity3D Engine. This has made the implementation of NEAT possible in Unity3D, which leads to visually seeing how the fencing AI models/neural nets grow over time. They also integrated data models using the assets from the Unity store to view the results of the training in real-time.

Fencing is a human sport, thus it faces a number of objectives to address. To keep the growth of the neural nets from being stagnant, the researchers used NSGA-III, a form of Evolutionary Multi-objective Optimization technique, to diversify and further improve the models. This helps the neural nets to adapt to certain situations and become stronger in various aspects or skills. More importantly, improve the areas they lack in.

Explicit Modelling, a type of Opponent Modelling, was also employed, to portray the realistic behavior of both fencing AI models and for the two neural nets to recognize

the strategy of their opponent player. It trained the AI models to take information about the opponent as input and produce a representation of the opponent as an output, either in terms of the actions the opponent is to take, the game-playing characteristics the opponent is expected to possess, or the strategies that an opponent is likely to employ.

The 'game strategies' that researchers referenced which were later placed in the input layer, are common attacks and defensive moves from fencing. The 'game-playing characteristics' were completely reliant on the stamina and the ticks factor, which refers to the timing and energy factor used to exert the moves. This resulted in the illustration of different behavioral models, to highlight the 'humanness' of the program. Aggressive, Defensive, Presumptuous, Stable Players are the behavioral models the researchers used to group existing trained AI neural nets.

Summary of Findings

The researchers were able to accomplish the following objectives which were stated in Chapter 1:

Develop a simulation using the NeuroEvolution of Augmenting Topologies (NEAT) within the Unity3D engine that generates a visualization of how different AI fencers evolve and improve over time.

The researchers have accomplished this objective by developing an environment in Unity using the Unity3D Engine as the medium for visualization. Assets and 3D Objects were also added to simulate a working arena for the fencers to fight in. The researchers then implemented SharpNEAT for the Fencer AI's ability to learn and strategize better over time by using competition as the main learning component.

The following specific objectives of this thesis have also been accomplished:

1. Implement multi-objective technique enhancement on the NEAT Algorithm by using simulated fencing competition as its main learning process.

The researchers have accomplished this by updating SharpNEAT's code that would support two competitive agents instead of just one. This would result in making the

fencers learn from themselves from previous generations as well as learning from their opponent.

2. Effectively use multi-objective optimization in Sharp NEAT to find all possible moves/solutions to given constraints.

The researchers have accomplished this objective by implementing different variables that affect the fitness score such as stamina, attack & defense zones, dodges, and hits. This resulted in evolving a fencer into adapting its moves into being more aggressive or defensive depending on its performance from the previous generations.

3. The optimization of the Fencing AI's input and output strategy, find behavioral models better than their counterparts, and aspects such as ‘‘humanness’’ of an AI fighter will also be investigated.

The researchers accomplished this objective by being able to enhance the Fencing AI’s ability to create various strategies of beating their opponents via the use of SharpNEAT within Unity. Through creating various strategies the Fencing AI’s also formed ‘personalities’ during the simulations such as being Cautious/Defensive, Aggressive/Offensive, Stable/Balanced, and Presumptuous models.

Conclusions

This research aimed to develop a simulation of Fencing using the NEAT algorithm in Unity using simulated fencing competition as the main learning process. Based on the results found through the development of the Fencing AI simulation, it can be concluded that it is possible to implement SharpNEAT by using multi-objective optimization on two (2) competitive agents for learning. It is also possible to use multi-objective optimization in Sharp NEAT by adding variables that affect the fitness score for the fencer's evolution. And lastly, it is possible to optimize the Fencing AI's input and output strategy and aspects such as 'humanness' of an AI fencer by opponent modeling and investigating the results of the fencers creating various strategies in adapting to their opponent which resulted in fencers forming their personalities.

Recommendations

To future researchers, further improvements on this study can be explored in terms of improving humanness in terms of additional actions, better animations and hitboxes, more fencer variables, tracking additional statistics, and improved simulation performance. One struggle the researchers encountered is the lack of animations and movements for fencing. Having no motion capture equipment means the researchers can't add realistic movements and animation themselves. To further support the research, adding more multi-objective optimization techniques should be investigated. Adding additional motion-capture to the animations would also help in developing a realistic model of the fencing simulation and give more accurate results.

To future game developers, exploring the application of the fencer's model into other kinds of competitive games and sports would also help in developing sophisticated Artificial Intelligence models in the future of game development. Also, finding ways of monetizing the underlying AI model especially to other fields in the industry would prove to be beneficial for the developers to further financially support their game development.

To future computer scientists, implementation of additional Genetic Algorithms would improve population selection and how fencer's are being evolved and improved overtime. Experimenting on different Evolutionary Algorithms and developing their own algorithms from scratch to find the most efficient one would also be beneficial in improving the simulation.

REFERENCES

Abramovich, Omer, and Amiram Moshaiov. Multi-Objective Topology and Weight Evolution of Neuro-Controllers. 2016, pp. 670–677.

Akut, R. & Kulkarni, S. (2019). NeuroEvolution : Using genetic algorithms for optimal design of deep learning models. 1–6. https://doi.org/10.1109/ICECCT.2019.8869233

Baarslag, T., Hendrikx, M., Hindriks, K., & Jonker, C. (2012). Measuring the Performance of Online Opponent Models in Automated Bilateral Negotiation. Lecture Notes in Computer Science, 1–14. https://doi.org/10.1007/978-3-642-35101-3_1

Baarslag, T., Hendrikx, M., Hindriks, K., & Jonker, C. (2012). Measuring the Performance of Online Opponent Models in Automated Bilateral Negotiation. Lecture Notes in Computer Science, 1–14. https://doi.org/10.1007/978-3-642-35101-3_1

Bakkes, S. C., Spronck, P. H., & Jaap Van Den Herik, H. (2009). Opponent modelling for case-based adaptive game AI.

Entertainment Computing, 1(1), 27–37. https://doi.org/10.1016/j.entcom.2009.09.001

Campos Ciro, G., Dugardin, F., Yalaoui, F., & Kelly, R. (2016). A NSGA-II and NSGA-III comparison for solving an open shop scheduling problem with resource constraints. IFAC-PapersOnLine, 49(12), 1272–1277. https://doi.org/10.1016/j.ifacol.2016.07.690

Carmel, D., & Markovitch, S. (1993). Learning Models of Opponent's Strategy Game Playing. AAAI Technical Report FS-93-02. Published.

Chang Kee Tong, Chin Kim On, Teo, J., & Mountstephens, J. (2012). Game AI generation using evolutionary multi-objective optimization. 2012 IEEE Congress on Evolutionary Computation. Published. https://doi.org/10.1109/cec.2012.6256638

Chen, C. (2017, December 4). GitHub - chen0040/cs-moea: Multi-Objective Evolutionary Algorithms implemented in .NET. GitHub. https://github.com/chen0040/cs-moea

Chiang, T. (2014, June). Tsung-Che Chiang, nsga3cpp: A C++ implementation of NSGA-III, 2014. Department of Computer Science and Information Engineering National Taiwan Normal University. http://web.ntnu.edu.tw/%7Etcchiang/publications/nsga3cpp/nsga3cpp.htm

Coello, C. C., Lamont, G. B., & Veldhuizen, D. V. A. (2007). Evolutionary Algorithms for Solving Multi-Objective Problems (Genetic and Evolutionary Computation) (2nd ed.). Springer.

Deb, K. (2011). Multi-objective Optimisation Using Evolutionary Algorithms: An Introduction. Multi-Objective Evolutionary Optimisation for Product Design and Manufacturing, 3-34. https://doi.org/10.1007/978-0-85729-652-8_1

Dubey, R., Ghantous, J., Louis, S., & Liu, S. (2018). Evolutionary Multi-objective Optimization of Real-Time Strategy Micro. 2018 IEEE Conference on Computational Intelligence and Games (CIG). Published. https://doi.org/10.1109/cig.2018.8490375

Ferner, J. N., Fischler, M., Zarubica, S., & Stucki, J. (2018). Combining Neuro-Evolution of Augmenting Topologies with Convolutional Neural Networks. https://www.researchgate.net/publication/328939814_Combining_Neuro-Evolution_of_Augmenting_Topologies_with_Convolutional_Neural_Networks

Fields, S., & Johnston, M. (2010). Genetic Twists of Fate (The MIT Press) (1st ed.). The MIT Press.

Gajurel, A. (2019, May). Neuroevolution for Realtime Strategy Game Micromanagement. University of Nevada, Reno. https://scholarworks.unr.edu/bitstream/handle/11714/5734/Gajurel_unr_0139M_12862.pdf?sequence=1&isAllowed=y

Gajurel, A., Louis, S. J., Mendez, D. J., & Liu, S. (2018). Neuroevolution for RTS Micro. 2018 IEEE Conference on Computational Intelligence and Games (CIG). Published. https://doi.org/10.1109/cig.2018.8490457

Ganzfried, S., & Sandholm, T. (2011). Game theory-based opponent modeling in large imperfect-information games. AAMAS '11: The 10th International Conference on Autonomous Agents and Multiagent Systems, 2, 533–540.

Hausknecht, M., Lehman, J., Miikkulainen, R., & Stone, P. (2014). A neuroevolution approach to general atari game playing. Computational Intelligence and AI in Games, IEEE Transactions On, 6, 355–366. https://doi.org/10.1109/TCIAIG.2013.2294713

He, H., Boyd-Graber, J., Kwok, K. & Daumé, III, H.. (2016). Opponent Modeling in Deep Reinforcement Learning. Proceedings of The 33rd International Conference on Machine Learning, in Proceedings of Machine Learning Research. 48:1804-1813

Hintze, A., Phillips, N. & Hertwig, R. The Janus face of Darwinian competition. Sci Rep 5, 13662 (2015). https://doi.org/10.1038/srep13662

Jha, S. K., & Josheski, F. (2016). Artificial evolution using neuroevolution of augmenting topologies (NEAT) for

kinetics study in diverse viscous mediums. Neural Computing and Applications, 29(12), 1337–1347. https://doi.org/10.1007/s00521-016-2664-2

Künzel, S., & Meyer-Nieberg, S. (2020). Coping with opponents: multi-objective evolutionary neural networks for fighting games. Neural Computing and Applications, 32(17), 13885–13916. https://doi.org/10.1007/s00521-020-04794-x

Lacerda, E. G. M. de, & Carvalho, A. C. P. de L. F. (1999). Introdução aos algoritmos genéticos. In Sistemas inteligentes: aplicações a recursos hídricos e sistemas ambientais. Porto Alegre: Ed. Universidade/UFRG/ABRH.

Lehman, J. (2013, June 7). Neuroevolution - Scholarpedia. Scholarpedia. http://www.scholarpedia.org/article/Neuroevolution

Lewis, M. (2008). Evolutionary Visual Art and Design. Natural Computing Series, 3–37. https://doi.org/10.1007/978-3-540-72877-1_1

Li, H. (n.d.). Machine Learning: What it is and why it matters. SAS. https://www.sas.com/en_ph/insights/analytics/machine-learning.html

Linden, R. (2021). Algoritmos Geneticos. CIENCIA MODERNA.

Lockett, A. J., Chen, C. L., & Miikkulainen, R. (2007). Evolving explicit opponent models in game playing. Proceedings of the 9th Annual Conference on Genetic and Evolutionary Computation - GECCO '07. Published. https://doi.org/10.1145/1276958.1277367

Lowell, Birger, and Grabkovsky (2011). Comparison of NEAT and HyperNEAT on a Strategic Decision-Making Problem. http://web.mit.edu/jessiehl/Public/aaai11/fullpaper.pdf

Mealing, R. A. (2015). Dynamic Opponent Modelling in Two-player Games. Amsterdam University Press.

Mendonca, M. R. F., Bernardino, H. S., & Neto, R. F. (2015). Simulating Human Behavior in Fighting Games Using Reinforcement Learning and Artificial Neural Networks. 2015

14th Brazilian Symposium on Computer Games and Digital Entertainment (SBGames). Published. https://doi.org/10.1109/sbgames.2015.25

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, Andrei A, Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., Petersen, S., Beattie, C., Sadik, A., Antonoglou, I., King, H., Kumaran, D., Wierstra, D., Legg, S., & Hassabis, D. (2015). Humanlevel control through deep reinforcement learning. Nature, 518(7540), 529-533. https://doi.org/10.1038/nature14236

Mohankumar, N., Bhuvan, B., Devi, N., & Arumugam, S. (2008). A modified genetic algorithm for Evolution of Neural Network in Designing an Evolutionary Neuro-Hardware. 108-111.https://www.researchgate.net/publication/220862351_A_Modified_Genetic_Algorithm_for_Evolution_of_Neural_Network_in_Designing_an_Evolutionary_Neuro-Hardware

Oh, Inseok & Rho, Seungeun, Et Al. (2019). Creating Pro-Level AI for Real-Time Fighting Game with Deep Reinforcement Learning. https://arxiv.org/ftp/arxiv/papers/1904/1904.03821.pdf

Ohazulike, A. E., & Brands, T. (2013). Multi-objective optimization of traffic externalities using tolls. 2013 IEEE Congress on Evolutionary Computation. Published. https://doi.org/10.1109/cec.2013.6557865

Olesen, J. K., Yannakakis, G. N., & Hallam, J. (2008). Real-time challenge balance in an RTS game using rtNEAT. 2008 IEEE Symposium On Computational Intelligence and Games, 87–94. https://doi.org/10.1109/cig.2008.5035625

Osiński, B., & Budek, K. (2021, January 5). Cookie and Privacy Settings. Deepsense.Ai. https://deepsense.ai/what-is-reinforcement-learning-the-complete-guide/

Reynoso-Meza, G., Sanchis, J., Blasco, X., & García-Nieto, S. (2014). Physical programming for preference driven evolutionary multi-objective optimization. Applied Soft Computing, 24, 341–362. https://doi.org/10.1016/j.asoc.2014.07.009

Rienstra, T., Thimm, M., & Oren, N. (2012). Opponent Models with Uncertainty for Strategic Argumentation. Opponent

Models with Uncertainty for Strategic Argumentation. Published.

Risi, S., & Togelius, J. (2015). Neuroevolution in games: State of the art and open challenges. IEEE Transactions on Computational Intelligence and AI in Games, PP(99), [7307180]. https://doi.org/10.1109/TCIAIG.2015.2494596

Rodriguez, J. (2019). The Emergence of Cooperative and Competitive AI Agents. KDnuggets. https://www.kdnuggets.com/2019/06/emergence-cooperative-competitive-ai-agents.html

Roi, Giulio, and Diana Bianchedi. "The Science of Fencing: Implications for Performance and Injury Prevention." Sports Medicine (Auckland, N.Z.), vol. 38, Feb. 2008, pp. 465–81.

Saumya. (2020, August 31). Artificial neural network vs Human brain | Verzeo. Verzeo EduTech Pvt. Ltd. https://verzeo.com/blog-artificial-neural-network-vs-human-brain#:%7E:text=An%20artificial%20neural%20network%20has,types%20of%20working%20and%20structure.&text=For%20the%20training%20session%20of,the%20same%20set%20of%20examples.

Schadd, F., Bakkes, E., & Spronck, P. (2007). Opponent modeling in real-time strategy games. Proceedings of the GAME-ON 2007, 61–68.

Schrum, J., & Miikkulainen, R. (2016). Discovering multimodal behavior in ms. Pac-man through evolution of modular neural networks. IEEE Transactions on Computational Intelligence and AI in Games, 8, 67–81. https://doi.org/10.1109/TCIAIG.2015.2390615

Seada, H., & Deb, K. (2015). U-NSGA-III: A Unified Evolutionary Optimization Procedure for Single, Multiple, and Many Objectives: Proof-of-Principle Results. Lecture Notes in Computer Science, 34–49. https://doi.org/10.1007/978-3-319-15892-1_3

Showalter and Schwartz (2020).Neuromodulated multiobjective evolutionary neural controllers without speciation. https://link.springer.com/article/10.1007/s12065-020-00394-9

Smyrnakis, M., & Leslie, D. S. (2010). Dynamic Opponent Modelling in Fictitious Play. The Computer Journal, 53(9), 1344–1359. https://doi.org/10.1093/comjnl/bxq006

Stanley, K. (2006). Exploiting Regularity Without Development. AAAI (National Conference on Artificial Intelligence). Published.

Stanley, K. O. (2014, December 12). NeuroEvolution of Augmenting Topologies. Ucf.Edu. http://www.cs.ucf.edu/%7Ekstanley/neat.html

Stanley, K. O., & Miikkulainen, R. (2002). Efficient reinforcement learning through evolving neural network topologies. In Proceedings of the 4th Annual Conference on Genetic and Evolutionary Computation (GECCO'02). Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 569–577.

Stanley, K. O., & Miikkulainen, R. (2002). Evolving Neural Networks through Augmenting Topologies. Evolutionary Computation, 10(2), 99–127. https://doi.org/10.1162/106365602320169811

Stanley, K. O., & Miikkulainen, R. (2004). Competitive Coevolution through Evolutionary Complexification. Journal of Artificial Intelligence Research, 21, 63-100. https://doi.org/10.1613/jair.1338

Stanley, K. O., & R. Miikkulainen. (2002). Evolving neural networks through augmenting topologies. Evolutionary Computation, 10, 99-127. https://doi.org/10.1162/106365602320169811

Stanley, K., Bryant, B., & Miikkulainen, R. (2005). Real-Time Neuroevolution in the NERO Video Game. IEEE Transactions on Evolutionary Computation, 9(6), 653-668. https://doi.org/10.1109/tevc.2005.856210

Stanley, K.O., (2007). Compositional pattern producing networks: A novel abstraction of development. Genetic Programming and Evolvable Machines: Special Issue on Developmental Systems. Vol. 8:2. Pp 131-162.

Stanley, K.O., D'Ambrosio, D., Gauci, J., (2009). A Hypercube-based indirect encoding for evolving large-scale neural networks. Artificial Life. Vol. 15:2. pp 185-212.

Such, F., Madhavan, V., Conti, E., Lehman, J., Stanley, K., & Clune, J. (2017). Deep neuroevolution: Genetic algorithms are a competitive alternative for training deep neural networks for reinforcement learning. https://arxiv.org/pdf/1712.06567.pdf

Tan, T. G., Anthony, P., Teo, J., & Ong, J. H. (2011). Neural network ensembles for video game AI using evolutionary multi-objective optimization. 2011 11th International Conference on Hybrid Intelligent Systems (HIS). Published. https://doi.org/10.1109/his.2011.6122174

Tanomaru, J. (1995). Motivacao, Fundamentos e Aplicacoes de Algoritmos Geneticos. Congresso Brasileiro de Redes Neurais, III Escola de Redes Neurais.

Than, K. (2018, February 27). What is Darwin's Theory of Evolution? Livescience.Com. https://www.livescience.com/474-controversy-evolution-works.html#:%7E:text=The%20theory%20of%20evolution%20by,heritable%20physical%20or%20behavioral%20traits

Tsung, C. (2014). A C++ implementation of NSGA-III. Tsung-Che Chiang >> Publications >> Nsga3cpp - Main. Retrieved 6 C.E., from http://web.ntnu.edu.tw/%7Etcchiang/publications/nsga3cpp/nsga3cpp.htm

Tutorailspoint. (2016). Genetic Algorithms - Introduction - Tutorialspoint. Tutorialspoint. https://www.tutorialspoint.com/genetic_algorithms/genetic_algorithms_introduction.htm#:%7E:text=Genetic%20Algorithm%20(GA)%20is%20a,take%20a%20lifetime%20to%20solve

Unity. (n.d.). Unity - Manual: Scriptable Render Pipeline introduction. Unity Documentation. https://docs.unity3d.com/Manual/scriptable-render-pipeline-introduction.html

van Willigen, W. H., Haasdijk, E. W., & Kester, L. (2013). Fast, Comfortable or Economical: Evolving Platooning Strategies with Many Objectives. In Proceedings of the 16th International IEEE Conference on Intelligent Transport Systems (ITSC 2013)

Vesga, L. (2019, March 17). GitHub - lfarizav/NSGA-III: NSGA-III, A-NSGA-III, and A^2-NSGA-III algorithms based on Kanpur Genetic Algorithms Laboratory's code. They solve Multi-objective Optimization Problems (MOPs) and Many-objective Optimization Problems (MaOPs) with constraints (Real and binary decision variables). GitHub. https://github.com/lfarizav/NSGA-III

Whiteson, S., Stone, P., Stanley, K. O., Miikkulainen, R., & Kohl, N. (2005). Automatic feature selection in neuroevolution. Proceedings of the 2005 Conference on Genetic and Evolutionary Computation - GECCO '05, 1225–1232. https://doi.org/10.1145/1068009.1068210

Whitley, D., T Starkweather, & C Bogart. (1990). Genetic algorithms and neural networks: optimizing connections and connectivity. Parallel Computing, 14, 347–361. https://doi.org/https://doi.org/10.1016/0167-8191(90)90086-O

Zhen, J. S., & Watson, I. (2013). Neuroevolution for Micromanagement in the Real-Time Strategy Game Starcraft:

Brood War. Lecture Notes in Computer Science, 8272, 259–270. https://doi.org/10.1007/978-3-319-03680-9_28

Zhu, D. (2020, May 26). How I Built an Intelligent Agent to Play Flappy Bird. Medium. https://medium.com/analytics-vidhya/how-i-built-an-ai-to-play-flappy-bird-81b672b66521

APPENDICES

Appendix A – Letter to the Adviser

March 5 , 2021

Bobby D. Gerardo

Professor 6

CICT - West Visayas State University (Main Campus)

Luna St., La Paz, Iloilo City 5000 , Iloilo, Philippines

Dear Dr. Bobby D. Gerardo,

The undersigned are BS in Computer Science Research 1/Thesis 1 students of CICT, this university. Our thesis/capstone project title is *"Fencing AI; NeuroEvolution of Augmenting Topologies (NEAT) towards Competition and Survival of the Fittest"*.

Knowing of your expertise in research and on the subject matter, we would like to request you to be our **ADVISER**.

We are positively hoping for your acceptance. Kindly check the corresponding box and affix your signature in the space provided. Thank you very much.

Respectfully yours,

1. Bryan Abesamis, Signature
2. Timothy Diosaban, Signature
3. Alyssa Danielle Magallanes, Signature
4. Neil Ryan Sustiguer, Signature

PS:

Advisers, are task to work with the students in providing direction and assistance as needed in their thesis/capstone project. They shall meet with the students weekly or as needed to provide direction, check on progress and assist in resolving problems until such a time that the students passed their defenses and submit their final requirements, as well as, preparing their evaluations and grades.

Action Taken:	
ok ☒ I Accept.	DR. BOBBY D. GERARDO
☐ Sorry. I don't accept.	Signature over printed name of the Adviser

CC:

CICT Dean
Research Coordinator
Group
**To be accomplished in 4 copies*

Appendix B - Letter of Request to the Technical Editor

	ADVISER'S ENDORSEMENT FORM (For Thesis Manuscript)	Document No.	WVSU-ICT-SOI-03-F10
		Issue No.	1
		Revision No.	0
	WEST VISAYAS STATE UNIVERSITY	Date of Effectivity:	April 27, 2018
		Issued by:	CICT
		Page No.	Page 1 of 1

Respectfully endorsed to the **Technical Editor**, the attached manuscript of the thesis entitled:

Fencing AI: NeuroEvolution of Augmenting Topologies (NEAT) towards Competition and Survival of the Fittest in a Fencing Simulation

The said manuscript has been presented to me for preliminary evaluation and guidance, and after a series of corrections/directions given which were implemented by the proponents whose names are listed hereunder and their thorough research, we have come to its completion.

Now, therefore, I hereby **ENDORSE** the said thesis manuscript to the Technical Editor for **TECHNICAL EDITING**.

DR. BOBBY D. GERARDO
Adviser's Name & Signature

Date: **January 5, 2022**

Group Members:
1. Bryan Kent S. Abesamis
2. Timothy A. Diosaban
3. Alyssa Danielle R. Magallanes
4. Neil Ryan S. Sustiguer

Appendix C - Letter of Request to the English Editor

	TECHNICAL EDITOR'S ENDORSEMENT FORM (For Thesis Manuscript)	Document No.	WVSU-ICT-SOI-03-F11
		Issue No.	1
		Revision No.	0
	WEST VISAYAS STATE UNIVERSITY	Date of Effectivity:	April 27, 2018
		Issued by:	CICT
		Page No.	Page 1 of 1

Respectfully endorsed to the **English Editor**, the attached manuscript of the thesis entitled:

Fencing AI: NeuroEvolution of Augmenting Topologies (NEAT) towards Competition and Survival of the Fittest in a Fencing Simulation

Said manuscript was presented to me and was reviewed and edited in terms of technical specifications, correctness of diagrams and other technical matters. The corrections and suggestions were carried and implemented by the proponents whose names are listed hereunder.

Now therefore, I hereby **ENDORSE** the said thesis manuscript to the English Editor/Grammarian for **English Grammar Editing.**

Dr. Frank I. Elijorde
Technical Editor's Name & Signature

Date: January 10, 2022

Group Members:
1. Bryan Kent S. Abesamis
2. Timothy A. Diosaban
3. Alyssa Danielle R. Magallanes
4. Neil Ryan S. Sustiguer

Appendix D - Letter to English Format Editor

January 3 , 2022
Bonna Sobrepeña Palma
Professor 6
CAS - West Visayas State University (Main Campus)
Luna St., La Paz, Iloilo City 5000 , Iloilo, Philippines

Dear Dr. Bonna Sobrepeña Palma,

The undersigned are BS in Computer Science Research 1/Thesis 1 students of CICT, this university. Our thesis/capstone project title is **"Fencing AI; NeuroEvolution of Augmenting Topologies (NEAT) towards Competition and Survival of the Fittest in a Fencing Simulation ".** Knowing of your expertise on the English language and grammar, we would like to request you to be our English Editor.

We are positively hoping for your acceptance. Kindly check the corresponding box and affix your signature in the space provided. Thank you very much.

Respectfully yours,

1. Bryan Abesamis, Signature
2. Timothy Diosaban, Signature
3. Alyssa Danielle Magallanes, Signature
4. Neil Ryan Sustiguer, Signature

DR. BONNA SOBRAPEÑA PALMA
Signature over printed name

Appendix E - Letter of Request to the Thesis Format Editor

	ENGLISH EDITOR/GRAMMARIAN'S ENDORSEMENT FORM (For Thesis Manuscript)	Document No.	WVSU-ICT-SOI-03-F12
		Issue No.	1
		Revision No.	0
	WEST VISAYAS STATE UNIVERSITY	Date of Effectivity:	April 27, 2018
		Issued by:	CICT
		Page No.	Page 1 of 1

Respectfully endorsed to the **Thesis Format Editor**, the attached manuscript of the thesis entitled:

Fencing AI: NeuroEvolution of Augmenting Topologies (NEAT) towards Competition and Survival of the Fittest in a Fencing Simulation

Said manuscript was presented to me for English grammar editing, corrections has been made and the proponents whose names are listed hereunder implemented said corrections and changes in the revised manuscript.

Now therefore, I hereby **ENDORSE** the said thesis manuscript for **Thesis Format Editing**.

DR. BONNA SOBREPEÑA PALMA
English Editor/Grammarian's Name and Signature

Date: January 17, 2022

Group Members:
1. Bryan Kent S. Abesamis
2. Timothy A. Diosaban
3. Alyssa Danielle R. Magallanes
4. Neil Ryan S. Sustiguer

Appendix F - Letter of Request to the Thesis Coordinator

	THESIS FORMAT EDITOR'S ENDORSEMENT FORM (For Thesis Manuscript)	Document No.	WVSU-ICT-SOI-03-F13
		Issue No.	1
		Revision No.	0
	WEST VISAYAS STATE UNIVERSITY	Date of Effectivity:	April 27, 2018
		Issued by:	CICT
		Page No.	Page 1 of 1

Respectfully endorsed to the **Thesis Coordinator**, the attached manuscript of the thesis entitled:

Fencing AI: NeuroEvolution of Augmenting Topologies (NEAT) towards Competition and Survival of the Fittest in a Fencing Simulation

Said manuscript was presented to me and has checked the preliminaries, thesis document convention and end matters, made some corrections which were implemented by the proponents whose names are listed hereunder.

Now therefore, I hereby **ENDORSE** said manuscript to the Thesis Coordinator for appropriate action.

DR. MA. LUCHE SABAYLE
Thesis Format Editor's Name and Signature

Date: JANUARY 21,2022

Group Members:
1. Bryan Kent S. Abesamis
2. Timothy A. Diosaban
3. Alyssa Danielle R. Magallanes
4. Neil Ryan S. Sustiguer

Note: *This form should be accomplished and signed if the corrections and changes made by the Thesis Format Editor have been implemented and the four (4) new copies have been printed ready for bookbinding.*

Appendix G - Signed letter for Output & Final Build Evaluation

Appendix H - Certification for Bookbinding

Appendix I – Gantt Chart

GANTT CHART

PROJECT TITLE: Fencing AI; NeuroEvolution of Augmenting Topologies (NEAT) towards Competition and Survival of the Fittest in a Fencing Simulation

PROJECT MANAGER: BSCS-4AAI Group 1 (BRYAN ARESAMIS, ALYSSA DANIELLE MAGALLANES, TIMOTHY DIOSABAN, & NEIL RYAN SUSTIGUER)

	TASK TITLE	START DATE	DUE DATE	DURATION	PCT OF TASK COMPLETE	Start of Thesis writing															
						March					April					May					
						11	16	17	18	19	1	9	21	22	23	1	9	10	15	25
1	March 11 - April 23																			
1.1	Sent letter of invitation for Thesis Adviser and Approved letter.	3/11/21	3/16/21	5	100%															
1.2	Creation & Completion of Chapter 1	3/17/21	3/19/21	2	100%															
1.3	Creation & Completion of Chapter 2	3/18/21	3/19/21	1	100%															
1.4	Added more content to Chapters 1 & 2	4/1/21	4/6/21	5	100%															
1.5	Creation of Chapter 3 start of creation simulation within Unity3D	4/8/21	4/9/21	1	100%															
1.6	Consultation with Thesis Adviser on Chapters 1-3 and showing prototype output	4/8/21	4/8/21	0	100%															
1.7	Consultation with Thesis Adviser, added more to chapter 3	4/21/21	4/21/21		100%															
1.8	Added more to Chapter 3, and revised Objectives Chapter 1	4/21/21	4/23/21	0	100%															
1.9	Added more to Chapter 2 related literature	4/21/21	4/23/21		100%															

	TASK TITLE	START DATE	DUE DATE	DURATION	PCT OF TASK COMPLETE	Start of Thesis writing														20% & 50% Defense Phase										
						March					April					May				June					July						
						11	16	17	18	19	1	9	21	22	23	1	9	10	15	25	1	9	10	15	25	1	5	15	25	31	8
1	March 11 - April 23																													
2	June - July 15																													
2.1	Reviewed Manuscript and revised chapters 1-3 of any mistakes.	6/1/21	6/9/21	8	100%																									
2.1.1	20% Defense	6/10/21	6/10/21	0	100%																									
2.2	Consulted with Steven Kunzel (researcher in Universität der Bundeswehr München) front runner for Multi-objective algorithms to Improve Simulation by adding more Multi-Objective algorithms.	6/11/21	6/15/21	4	100%																									
2.3	Developed the other Fencer and the graphical assets seen in the build.	6/16/21	6/23/21	7	100%																									
2.4	Added data simulation (graphs and charts). Contacted BitSplash support for the addition of Graphs and Charts assets to be used for data simulation.	6/24/21	6/25/21	1	100%																									
2.5	Started Chapter 4, revised chapter 3 and added more formulas to simulation.	6/25/21	7/5/21	10	100%																									
2.6	Revised Chapter 4 and consulted with Thesis adviser.	6/25/21	7/8/21	13	100%																									
2.7	Added more to Chapter 2 Related Literature	6/25/21	7/12/21	17	100%																									
2.8	Added more terms to definition of terms.	6/25/21	7/14/21	19	100%																									
2.9	50% Defense	7/15/21	7/15/21	0	100%																									

	TASK TITLE	START DATE	DUE DATE	DURATION	PCT OF TASK COMPLETE
1	March 11 - April 23				
2	June - July 15				
3	July 16 - October 30				
3.1	Revised Manuscript to the panels suggestions and recommendations	7/16/21	7/23/21	7	100%
3.2	Added in a way to get data from each run in simulation and added in new features for fencer A.I.	7/26/21	7/29/21	3	100%
3.3	Consulted with Mr. Mark Joseph Solidarios for any idea to integrate Python within Unity3D	9/8/21	9/8/21	0	100%
3.4	Ran 16 runs for data collection and ran experimental runs to check if simulation is okay.	9/9/21	9/30/21	21	100%
3.5	Cleaned and compiled data into a sheet file and started to check for results to add it in Chapter 4	10/1/21	10/15/21	14	100%
3.6	Started Chapter 5, reviewed data and checked for data validation techniques	10/18/21	10/28/21	10	100%

Final Defens

September					October									
8	9	15	25	30	1	5	15	25	31					

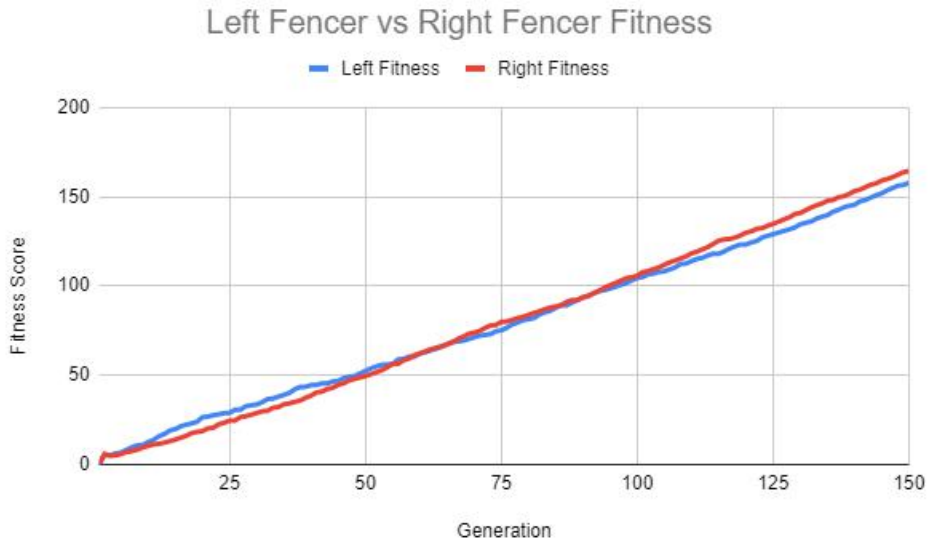
	TASK TITLE	START DATE	DUE DATE	DURATION	PCT OF TASK COMPLETE
1	March 11 - April 23				
2	June - July 15				
3	July 16 - October 30				
4	November - December 2				
4.1	Consulted with Dr. Regin A. Cabacas, Ph. D. Given suggestions that instead of data validation, we shall do performance evaluation and compare to previous studies that are related to the Thesis	11/10/21	11/10/21	0	100%
4.2	Researched on new literature's that can be compared to thesis	11/11/21	11/17/21	6	100%
4.3	Consulted with Dr. Frank Elijorde Provided guidance on the manuscript and the procedures about presenting the results and Chapter 4 of the thesis overall. Also suggested on presenting the results by comparing it with similar literature.	11/18/21	11/18/21	0	100%
4.4	Added in a new figures for Chapters 2 & 3 and the pseudocode of the algorithm for chapter 3 and researched for more literature	11/19/21	11/23/21	4	100%
4.5	Final Consultation with Thesis Adviser and Provided guidance for the manuscript and Adviser also gave the go signal for the Final Defense Presentation.	11/30/21	11/30/21	0	100%
4.6	Finalized Thesis Manuscript and got ready for Thesis final defense	12/1/21	12/1/21	0	100%
4.7	FINAL DEFENSE (PASSED)	12/2/21	12/2/21	0	100%

Final Defense Phase

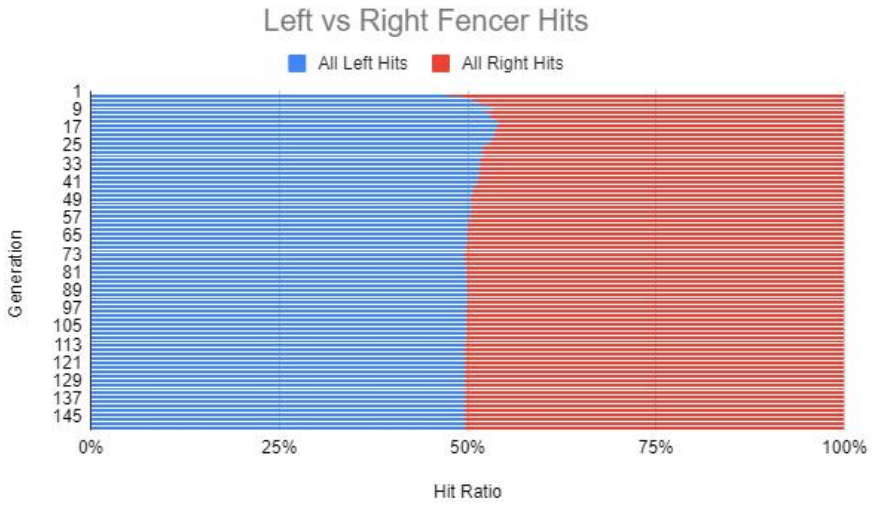
September					October					November					December				
8	9	15	25	30	1	5	15	25	31	10	11	17	25	30	1	2	3	4	5

Appendix J – Graphs and Charts

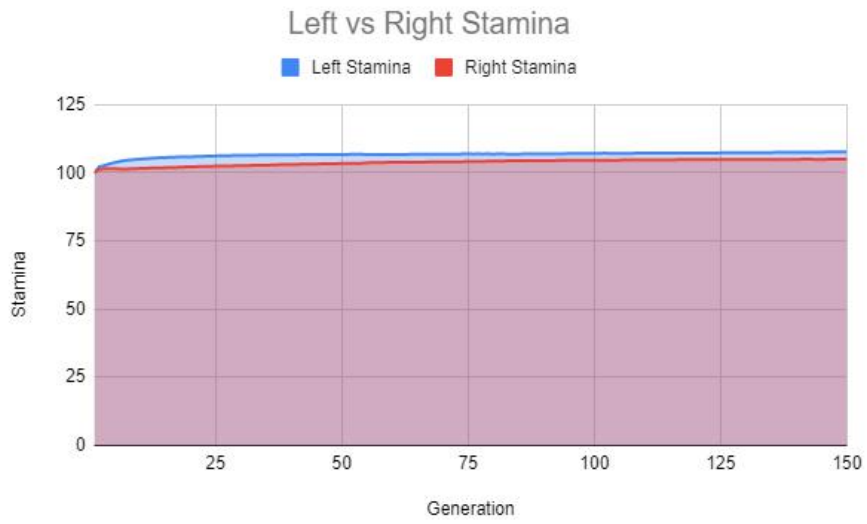
Graph 1. Left Fencer vs Right Fencer Fitness



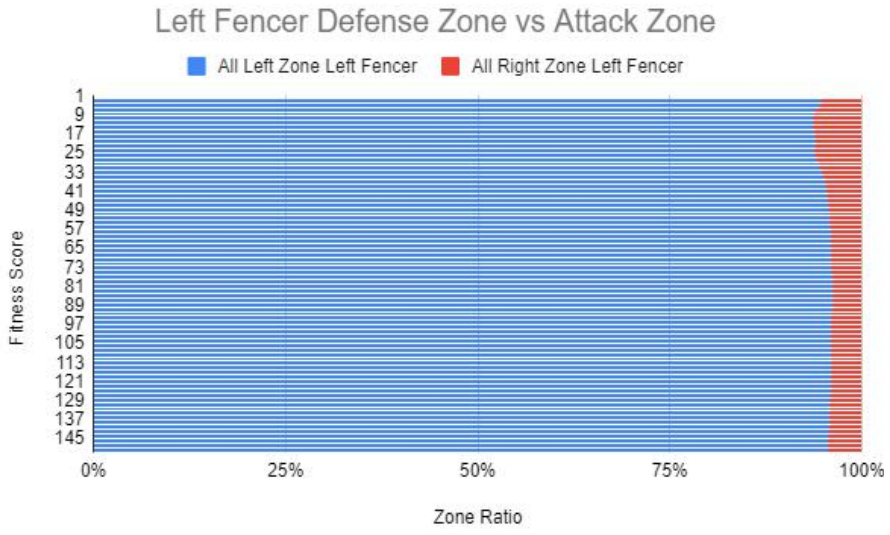
Graph 2. Left Fencer vs Right Fencer Hits



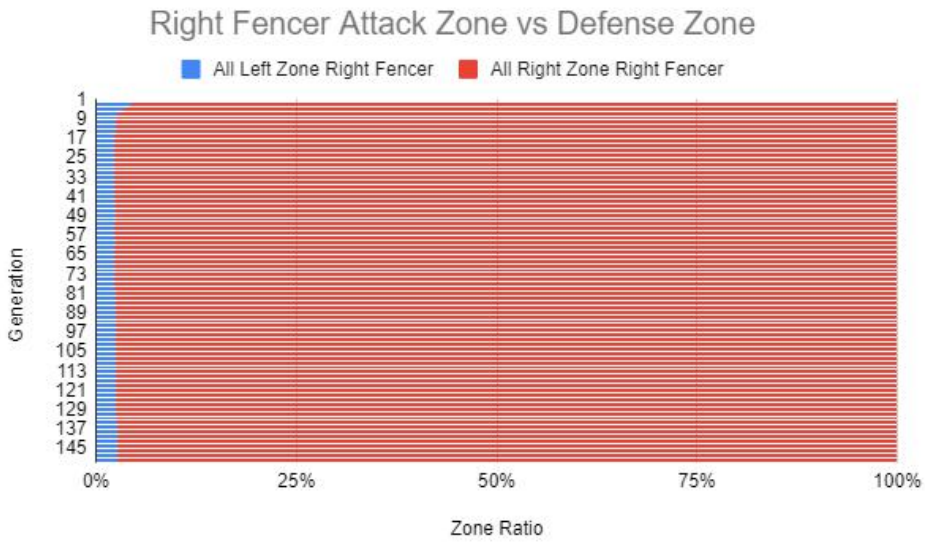
Graph 3. Left Fencer vs Right Fencer Stamina



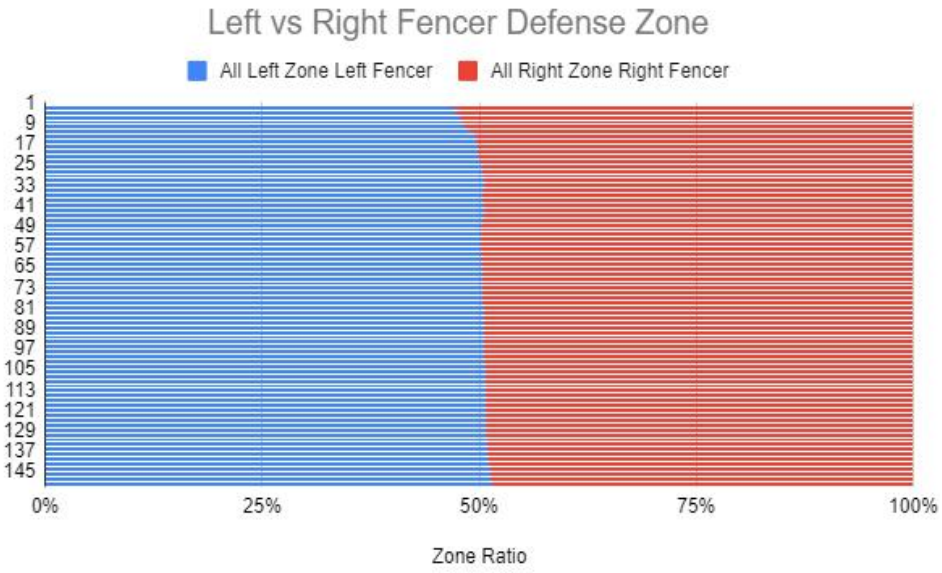
Graph 4. Left Fencers Defense Zone and Attack Zone



Graph 5. Right Fencers Defense and Attack Zone

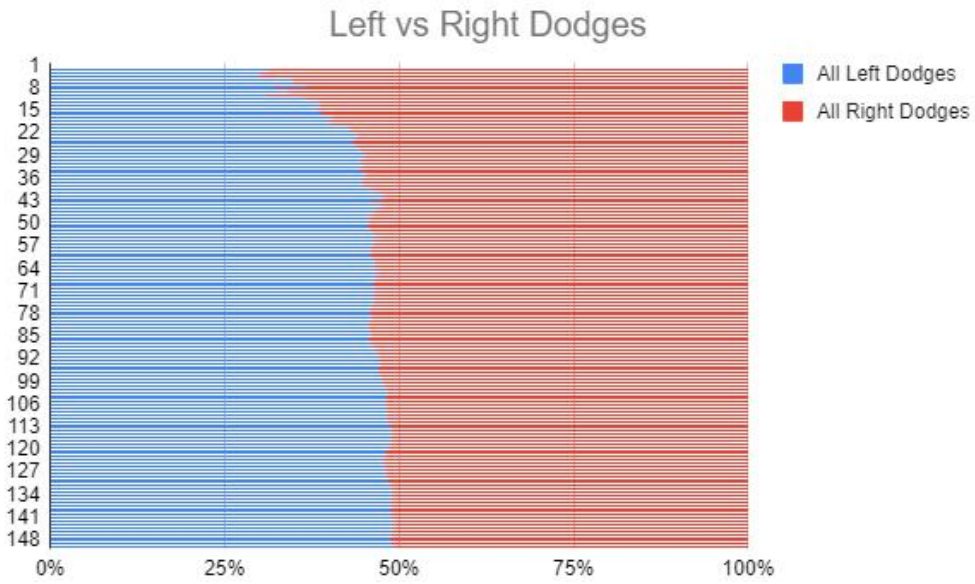


Graph 6. Left Fencer vs Right Fencer Defense Zone

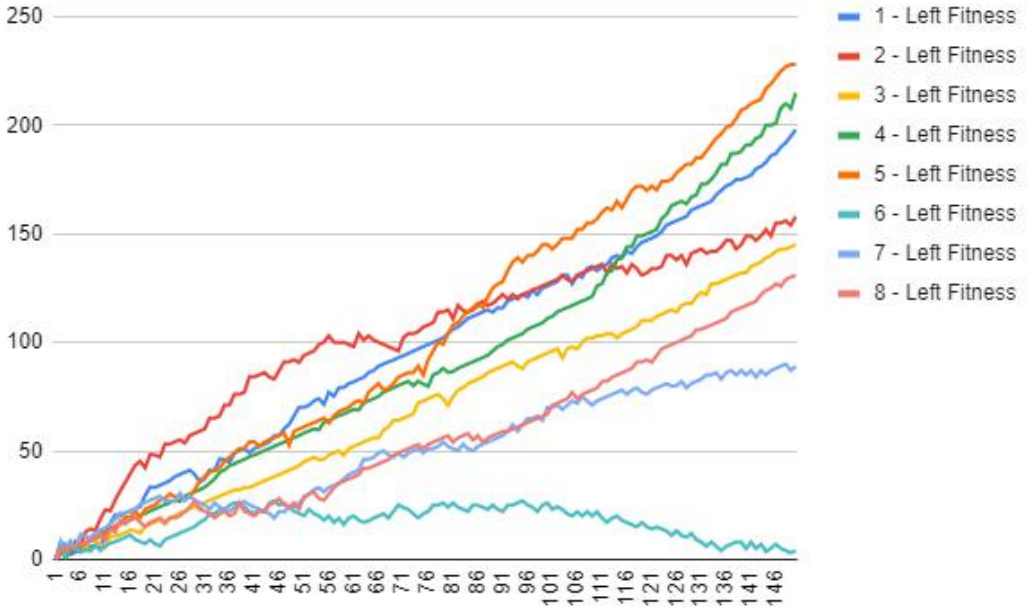


Graph 7. Left Fencer vs Right Fencer Attack Zone

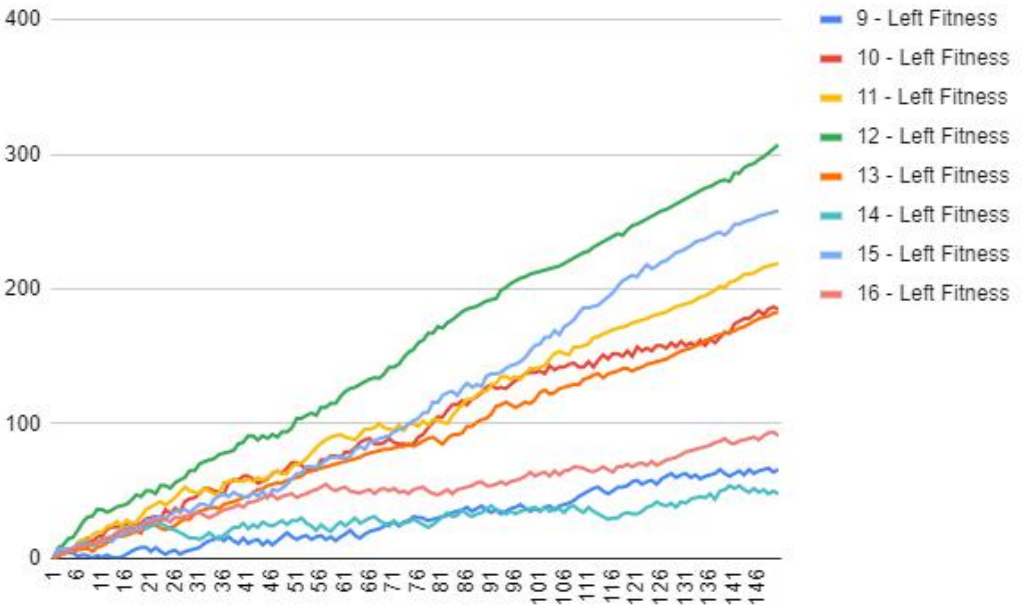
Graph 8. Left Fencer amount of Dodges vs Right Fencer Dodges



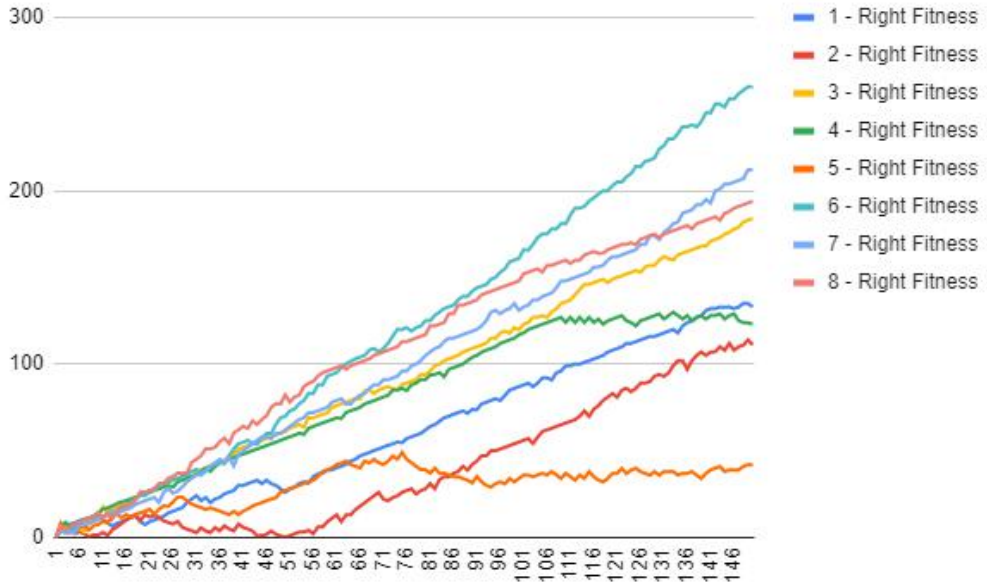
Graph 9. Runs 1-8 Left Fencer's Fitness Score



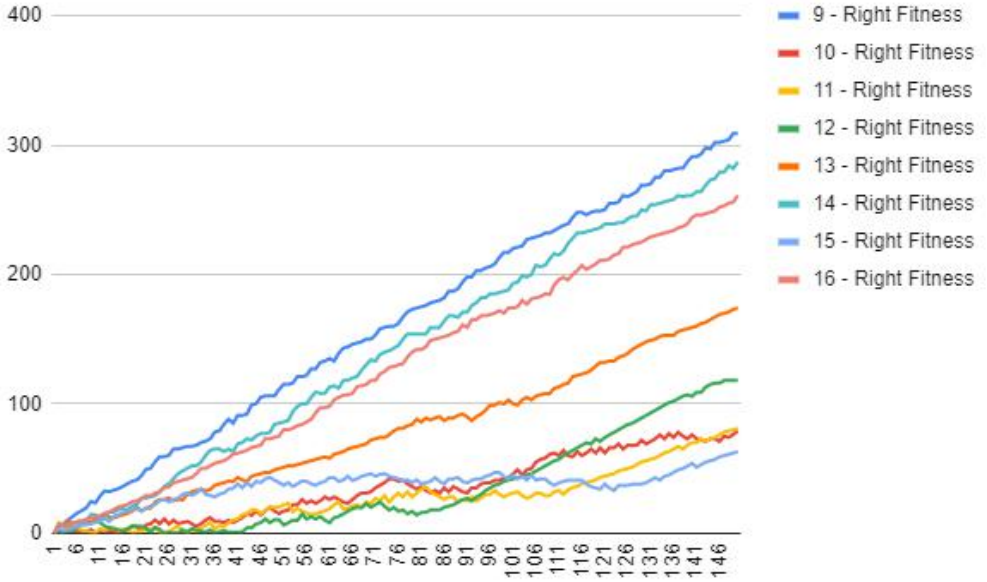
Graph 10. Runs 9-16 Left Fencer's Fitness Score



Graph 11. Runs 1-8 Right Fencer's Fitness Score



Graph 12. Runs 9-16 Right Fencer's Fitness Score



Appendix K - Data Tables

Aggregated Mean Table

Generation	Left Fitness	Right Fitness	Left Stamina	Right Stamina	All Left Hits	All Right Hits
1	0	0	100	100	0	0
2	5.442597444	5.916515438	102.2523438	101.2355063	1.625	1.8125
3	4.940350063	4.985683494	102.6444688	101.5388088	2.9375	3
4	6.216347063	5.032820313	103.2613125	101.4768775	4	3.875
5	6.612548688	5.79089375	103.7029375	101.45275	5.3125	5.125
6	8.146235875	6.9872055	104.1361813	101.4018081	6.4375	6.0625
7	9.641473563	7.492926563	104.4850625	101.3785063	7.75	6.9375
8	10.743435506	8.265670106	104.6929813	101.42562	8.875	7.6875
9	10.938076699	9.460385006	104.836825	101.5155213	9.75	8.875

10	12.4971 2581	10.44838 201	104.98418 13	101.57660 19	10.9375	9.75
11	13.7605 6757	11.21063 538	105.13118 13	101.62766 25	12.125	10.812 5
12	15.7929 6494	11.59092 794	105.2525	101.71703 75	13.3125	11.75
13	17.0670 821	12.20187 006	105.40831 88	101.72597 75	14.8125	12.75
14	19.0323 2911	13.08595 643	105.52848 75	101.77226 38	16	13.562 5
15	19.8252 1181	13.92719 005	105.58226 25	101.85840 25	17.0625	14.562 5
16	21.3760 9369	14.97404 463	105.68066 25	101.89408 13	18.25	15.562 5
17	22.3271 1475	16.05413 719	105.75221 88	101.95067 13	19.25	16.562 5
18	23.0013 5163	17.57678 581	105.79889 38	102.02371	20.4375	17.687 5
19	23.9074 1288	18.25458 375	105.80602 5	102.13996 81	21.4375	18.812 5
20	26.4969 2375	18.65706 006	105.86676 88	102.18708 88	22.6875	19.5

21	26.79017213	20.21285375	105.9155375	102.2419694	23.625	20.625
22	27.61487081	20.59275433	105.9647688	102.3174263	24.625	21.625
23	28.11018275	22.53293619	106.0601938	102.3269419	25.4375	22.625
24	28.70245713	23.38375091	106.1142813	102.3608425	26.3125	23.8125
25	28.92622681	24.52940103	106.153625	102.4317425	27.0625	25
26	30.64175088	24.63015006	106.2706938	102.4070569	28.0625	25.8125
27	30.74148644	26.57983439	106.25495	102.5013231	29	26.9375
28	32.68883244	27.08156464	106.3120125	102.51893	30.1875	27.9375
29	33.1821575	27.89906288	106.3580813	102.5553806	31.0625	28.8125
30	33.60658081	28.85697052	106.3880438	102.5959406	32	30
31	34.92283425	29.82299031	106.4200313	102.6334144	32.875	30.8125

32	36.640995	30.07720419	106.4410438	102.68128	34	31.5625
33	36.97088938	31.81966744	106.449575	102.7370738	35	32.875
34	38.06623	32.14927043	106.4828688	102.7659556	35.9375	33.8125
35	38.97892	33.75605956	106.5152688	102.7943256	36.9375	34.8125
36	40.08970938	34.20426501	106.5373063	102.8332275	38.125	35.9375
37	42.1510325	35.1187465	106.4880125	102.9399856	39.5	36.9375
38	43.42483313	35.83216675	106.5048625	102.9810481	40.5	37.875
39	43.54046	37.37581554	106.50865	103.0416731	41.3125	39
40	44.44623625	38.59233084	106.5351813	103.0769881	42.1875	39.9375
41	44.65496813	40.37165469	106.5771875	103.0884288	42.9375	41.0625
42	45.43012813	40.85453756	106.60735	103.1188219	43.8125	42.0625

43	45.45052125	42.29295381	106.643725	103.1382538	44.5	43
44	46.51429563	43.00997303	106.6255125	103.2105469	45.375	44.125
45	46.8986	44.60374769	106.654925	103.2281375	46.25	45.25
46	48.4129575	45.487823	106.6754688	103.2551869	47.375	46.25
47	49.10586	47.05718538	106.6647563	103.3114994	48.375	47.4375
48	49.57826563	47.94071106	106.6902625	103.3317031	49.1875	48.4375
49	51.00739563	48.72409885	106.7005813	103.3653513	50.25	49.4375
50	52.38768	49.49892511	106.7092375	103.4009831	51.4375	50.4375
51	53.83728813	50.55319806	106.7263125	103.4274844	52.6875	51.5625
52	54.92189688	51.705826	106.7582813	103.436695	53.8125	52.75
53	55.82819938	52.84069763	106.7719813	103.4655094	54.9375	53.9375

54	56.04694438	54.34774806	106.7780563	103.5006769	55.9375	55.25
55	56.15802625	56.37661931	106.6679688	103.6505406	56.625	56.3125
56	58.86802375	56.35055744	106.673575	103.6837106	58	57.3125
57	59.10041688	58.384769	106.6555563	103.7407013	58.9375	58.5625
58	60.3961325	59.27826781	106.6817313	103.7530763	60	59.625
59	60.46779563	61.188948	106.6566688	103.8155469	60.875	60.9375
60	62.135625	62.20969688	106.6732563	103.8354631	62.125	62.125
61	62.8718775	63.33404963	106.6768063	103.8673813	63.0625	63.1875
62	63.89152625	64.53382219	106.7116313	103.8679919	64.1875	64.375
63	64.804715	65.33742813	106.720025	103.8946619	65	65.125
64	65.97561688	66.21091188	106.79595	103.8580831	66.0625	66.25

65	67.27013125	67.3157537 5	106.8241875	103.862805	67.1875	67.25
66	68.66714563	68.42173688	106.8134938	103.910655	68.625	68.5625
67	68.90345313	70.34825813	106.8033375	103.9528431	69.5	69.75
68	69.35789438	71.61331375	106.8220938	103.96677	70.375	70.9375
69	70.24448875	73.20192313	106.8328688	103.9877669	71.3125	72.125
70	71.264895	73.86794813	106.8542625	103.9979525	72.1875	73.125
71	72.17730438	74.76410625	106.8666938	104.017045	73.125	73.9375
72	72.5961525	76.49493875	106.847125	104.0677606	73.9375	75.0625
73	73.05715938	77.85672625	106.8668625	104.0778506	74.875	76.375
74	74.64468563	78.17666375	106.8920563	104.0831513	75.8125	77.125
75	75.11718188	79.86927063	106.909125	104.0988619	76.9375	78.3125

76	76.64532813	80.09850438	106.8799563	104.1572719	78.0625	79.1875
77	78.45247313	81.01674125	106.8954438	104.1712769	79.1875	80.125
78	79.47778875	82.03234438	106.8788625	104.2166769	80.3125	81.1875
79	80.8161925	82.68754313	106.9004	104.2287931	81.5	82.1875
80	81.38016188	83.760035	106.8670063	104.2998963	82.375	83
81	82.00076188	84.84333875	106.90585	104.2916263	83.25	84
82	83.99844313	85.81059313	106.9209625	104.3027681	84.5625	85.125
83	85.21831375	86.93414	106.8425938	104.4109938	85.75	86.3125
84	86.05652	87.90463	106.865875	104.4134938	86.8125	87.375
85	87.73286313	88.49853	106.8728375	104.4322	88	88.3125
86	89.17376313	89.23964688	106.890275	104.4398313	89.1875	89.25

87	89.05039563	91.06337188	106.893925	104.4618	89.9375	90.4375
88	90.68596063	92.0531775	106.9100938	104.4702125	91.125	91.5625
89	92.157555	92.13120563	106.9218125	104.4828875	92.4375	92.625
90	93.37618938	93.73156688	106.939575	104.4927375	93.5625	93.75
91	94.62743125	94.34559625	106.9646563	104.4919938	94.75	94.75
92	95.80305313	96.04470125	106.9826188	104.4981813	96	96
93	97.36221875	97.10015375	106.9945625	104.5096313	97.3125	97.25
94	97.56308125	99.10981438	107.0144688	104.5128	98.125	98.4375
95	98.55509625	100.40829	107.0376313	104.5122375	99.375	99.875
96	99.54004063	101.670755	107.0456688	104.52685	100.375	101
97	100.6207881	103.0283238	107.0693438	104.5258125	101.5	102.25

98	101.6233113	104.35412	107.0871563	104.5308063	102.5625	103.4375
99	103.114335	104.915885	107.1045875	104.5358875	103.75	104.4375
100	104.4329481	105.7143806	107.1241875	104.5382375	104.875	105.4375
101	104.99735	107.7262338	107.1419688	104.542425	105.9375	106.75
102	106.5148013	108.3964925	107.1796438	104.526325	107.125	107.8125
103	106.9154769	109.4314356	107.1290375	104.6025938	107.875	108.625
104	107.8515869	110.4133	107.1430063	104.6098438	108.875	109.6875
105	108.2119525	111.88194	107.1343813	104.6407125	109.6875	110.75
106	109.2730231	113.3002581	107.1387813	104.6567875	110.8125	112.0625
107	110.4589981	114.318775	107.1511875	104.6650063	112	113.25
108	112.3593094	115.4160713	107.165075	104.67155	113.3125	114.375

109	112.5664131	116.58325	107.1680938	104.6890063	114.375	115.625
110	113.9887006	118.2822813	107.185775	104.691475	115.4375	116.8125
111	115.1739413	119.2265688	107.1987	104.6987125	116.5	117.9375
112	115.7385744	120.5290981	107.220225	104.6973375	117.5625	119.0625
113	117.1480019	121.7841313	107.213875	104.7234	118.8125	120.4375
114	118.0310663	123.5665306	107.2092625	104.7475625	119.875	121.625
115	118.026795	125.3524681	107.2112938	104.7648063	120.75	122.9375
116	119.3349944	126.1091781	107.2134813	104.78155	121.75	123.9375
117	120.9641144	126.3461888	107.2223063	104.791975	122.875	124.75
118	122.1546825	127.0963944	107.2313688	104.8019688	123.875	125.5625
119	123.2128188	128.29969	107.2334938	104.8182438	124.9375	126.625

120	123.2186481	129.8621069	107.239575	104.8327375	125.625	127.5625
121	124.3981838	130.5970656	107.247475	104.8428125	126.6875	128.6875
122	125.22349	132.0325538	107.2612438	104.8469688	127.625	129.75
123	127.1103394	132.4012694	107.2793688	104.8467625	128.9375	130.75
124	128.1763769	133.7182569	107.2976438	104.8456813	130	131.8125
125	128.9094181	134.7841013	107.3052875	104.8557313	131	132.9375
126	129.9318769	136.0132338	107.3335875	104.8448	132.125	134.0625
127	130.8547619	137.281005	107.3429375	104.8526438	133.0625	135.1875
128	131.7332338	138.446105	107.3663563	104.846350	134.1875	136.375
129	132.7812338	140.2868313	107.37245	104.857025	135.4375	137.8125
130	134.5492039	140.8418038	107.385275	104.8612813	136.625	138.875

131	135.4106957	142.4049188	107.392725	104.8704	137.8125	140.1875
132	136.1006546	143.9614713	107.4038	104.8760375	138.9375	141.5
133	137.7231791	145.15881	107.4147688	104.8813875	140.25	142.75
134	138.8495936	146.2227788	107.4242313	104.8883	141.25	143.75
135	139.6040481	147.7226669	107.4384813	104.8921875	142.4375	145
136	141.5406269	148.1517713	107.44955	104.89745	143.75	146.125
137	142.5923735	149.6586494	107.459225	104.9036438	145	147.3125
138	144.0347131	150.2633975	107.4731813	104.9050875	146.1875	148.3125
139	144.8992428	151.5178038	107.4820438	104.912125	147.3125	149.5
140	145.4535869	153.2108106	107.4899688	104.9203938	148.5	150.9375
141	147.4134903	153.9572944	107.5030625	104.9227625	149.625	151.875

142	148.2260871	155.4019319	107.5149188	104.92865	150.8125	153.1875
143	149.5418441	156.7130831	107.5289063	104.93005	151.9375	154.375
144	150.8837231	157.4860625	107.5740813	104.90365	153.25	155.5
145	151.9108959	159.1004125	107.5804875	104.9142688	154.3125	156.625
146	153.5951117	159.9162588	107.5858625	104.9242063	155.6875	157.8125
147	154.9043972	160.966565	107.5970313	104.9280188	157	158.9375
148	156.1572876	162.2799688	107.6059563	104.9339375	158.1875	160.1875
149	156.5682859	163.7290281	107.5928375	104.9621	159.1875	161.4375
150	157.8500607	164.4628244	107.6013438	104.9686625	160.125	162.375

Generation	All Left Dodges	All Right Dodges	All Left Zone Left Fencer	All Left Zone Right Fencer	All Right Zone Left Fencer	All Right Zone Right Fencer
1	0	0	0	0	0	0
2	0.3125	0.6875	65.1875	3.5	3.8125	75.1875
3	0.375	0.875	90.5	4.25	5.0625	98.625
4	0.375	0.875	118.4375	4.875	6.875	131.75
5	0.5625	1.0625	143.375	5.4375	8.5	159.875
6	0.5625	1.0625	171.25	5.625	10.6875	186.1875
7	0.6875	1.1875	199	5.875	13.0625	217
8	0.6875	1.4375	223.125	6.25	15.1875	242.9375
9	0.8125	1.5625	247.6875	6.875	16.875	265.25
10	0.8125	1.8125	271.75	7.25	18.5	290.125
11	1.25	2.1875	296.875	7.9375	20.4375	311.875
12	1.4375	2.3125	327.125	8.5625	21.8125	337.1875
13	1.8125	2.875	351.9375	8.75	24.0625	360.937

						5
14	1.8125	2.9375	379.1875	9.0625	26.125	385.6875
15	1.875	2.9375	403.3125	9.5625	27.125	410.875
16	1.875	2.9375	427.9375	9.75	28.5625	435
17	2	2.9375	452.8125	10.1875	30.0625	456.75
18	2.125	3.25	477.3125	10.8125	30.75	482.3125
19	2.3125	3.4375	504.375	11.5	32.875	508.0625
20	2.5625	3.4375	528.875	12.0625	34.125	531
21	2.5625	3.4375	553.625	12.75	34.9375	557.75
22	2.875	3.625	586.5	13.8125	37.125	586.125
23	3	3.875	613.125	14.25	41.25	608.25
24	3	3.9375	637.1875	14.5625	42.3125	635.0625
25	3.0625	4	669.1875	15.3125	43.9375	657.875
26	3.1875	4.0625	697.1875	15.6875	45	686.1875
27	3.4375	4.3125	721.5625	16.5	45.375	712.5
28	3.5625	4.3125	745.5	17	46.125	735.5

29	3.6875	4.5	772.9375	17.4375	46.25	759.5
30	3.8125	4.75	797.125	18.375	46.75	782.125
31	3.8125	4.8125	821.6875	18.5625	46.875	806.5
32	3.9375	4.875	846.8125	18.9375	47.3125	830.3125
33	3.9375	4.9375	870.9375	19.5625	47.4375	852.125
34	4.125	5	894.9375	20	48	876.6875
35	4.25	5.1875	919	20.4375	48.5625	902.125
36	4.4375	5.5	943.8125	21.0625	48.8125	928.9375
37	4.5625	5.625	967.875	21.875	49.25	951.4375
38	4.5625	5.625	992.625	22.625	49.375	977
39	4.8125	5.625	1021.75	23.75	50.1875	1004.5
40	5.0625	5.6875	1049.6875	24.1875	50.625	1029.5
41	5.125	5.6875	1074.5	25.25	51.5625	1053.3125
42	5.375	5.875	1103.875	25.5625	53.125	1081.625
43	5.375	6.3125	1130.8125	26.5625	54.125	1105.81

						25
44	5.6875	6.3125	1158.4375	27.8125	54.6875	1133.9375
45	5.6875	6.5	1182.375	28	55.375	1163.3125
46	5.6875	6.6875	1207.125	28.875	55.875	1191.75
47	5.8125	6.875	1231.3125	29.375	56.5625	1212.5625
48	6.0625	7.125	1256.0625	30.1875	57.375	1239.0625
49	6.125	7.3125	1280.3125	30.625	57.9375	1268.4375
50	6.1875	7.4375	1305.25	31.3125	58.5	1297.1875
51	6.1875	7.5	1330.25	31.625	59.125	1322.75
52	6.3125	7.5	1354.375	31.625	60.0625	1346.625
53	6.5625	7.625	1379.6875	32.4375	60.5	1371.9375
54	7.0625	8	1404.6875	33.0625	60.75	1396.625
55	7.1875	8.375	1429.25	33.8125	61.6875	1419.37

						5
56	7.3125	8.5	1453.625	34.125	62.5	1443.75
57	7.375	8.625	1478.5	35.25	63.1875	1467.625
58	7.375	8.75	1503.5625	35.75	64	1491.9375
59	7.5625	8.875	1528.25	36.625	64.625	1515.1875
60	7.5625	8.9375	1552.75	37.125	65.625	1539.375
61	7.75	8.9375	1576.9375	37.6875	66.125	1565.3125
62	7.875	9.0625	1601.4375	38.0625	67.3125	1584.8125
63	7.875	9.1875	1626.0625	38.6875	68	1606.8125
64	8.125	9.25	1653.9375	39.125	69.3125	1635
65	8.125	9.25	1677.875	39.375	70.25	1655.75
66	8.25	9.375	1705.8125	39.9375	71.5	1682.9375
67	8.25	9.5	1729.75	40.5625	72	1706.625

68	8.4375	9.75	1754.5625	41.25	73	1727.9375
69	8.4375	9.875	1779	41.9375	73.75	1753.25
70	8.6875	9.875	1803.625	42.5625	74.8125	1780.625
71	8.75	10.0625	1828.5625	42.9375	75.75	1805.1875
72	8.75	10.125	1853.5	44.125	76.375	1830.3125
73	8.9375	10.375	1877.6875	44.6875	77.0625	1855.25
74	9.125	10.4375	1902.8125	45.125	78.125	1880.9375
75	9.1875	10.75	1930.3125	45.4375	78.9375	1904.5
76	9.375	11.125	1955.0625	46.125	79.5	1929.9375
77	9.375	11.125	1980.3125	46.5	80.0625	1949.4375
78	9.5	11.125	2005.3125	47.3125	80.6875	1973.125
79	9.625	11.3125	2035.0625	47.625	81.25	1998.125
80	9.75	11.5625	2070.3125	49.0625	82.625	2024.18

						75
81	9.9375	11.875	2097	49.5	85.875	2048
82	9.9375	11.875	2120.9375	49.75	86.4375	2075.375
83	10	11.875	2149.3125	51	87.3125	2101.75
84	10.1875	12	2173.375	51.25	88.0625	2125.1875
85	10.25	12.125	2197.6875	51.875	88.625	2147.625
86	10.25	12.25	2221.625	52.25	89.8125	2169.6875
87	10.5	12.375	2246.375	53.4375	90.5625	2192.5625
88	10.5625	12.375	2270.375	53.875	91.6875	2216.4375
89	11.0625	12.6875	2294.5	54.375	92.5625	2241.375
90	11.3125	12.75	2322.125	55.5625	94.375	2263.5625
91	11.5625	13	2346.6875	55.9375	96.0625	2288.8125
92	11.687	13.125	2371.4375	56.5	97.375	2316.06

	5					25
93	11.8125	13.1875	2395.5	57.25	98.4375	2340.3125
94	11.8125	13.1875	2419.5	57.5	99.75	2362.875
95	11.9375	13.4375	2443.25	57.9375	101.0625	2385.3125
96	12.0625	13.5625	2467.4375	58.625	102	2406.0625
97	12.1875	13.625	2491.6875	59	103.3125	2429.6875
98	12.3125	13.625	2516.5	59.5	104.125	2452
99	12.4375	13.75	2541.25	60.0625	105.375	2479
100	12.6875	13.75	2565.625	60.25	106.25	2504.0625
101	12.75	13.75	2590.1875	61.0625	107.5	2525.8125
102	12.8125	13.8125	2614.625	61.25	109.0625	2550.0625
103	12.937	13.9375	2644.375	63.0625	110	2574.18

	5					75
104	13.0625	14.1875	2668.75	63.875	110.75	2596.875
105	13.3125	14.375	2694.6875	64.875	111.6875	2617.875
106	13.4375	14.625	2718.6875	65.1875	112.5625	2647.8125
107	13.75	14.8125	2742.9375	65.9375	113.9375	2671.9375
108	13.75	14.8125	2767.4375	66.375	114.9375	2695.0625
109	14.1875	15.25	2792.125	67.125	115.625	2718.0625
110	14.25	15.25	2816.75	67.5	116.875	2742.625
111	14.375	15.25	2841.5625	67.875	117.875	2765.375
112	14.5625	15.5	2866.375	68.375	119.1875	2787.4375
113	14.875	15.5625	2891.125	69	120.5625	2811.4375
114	14.875	15.5625	2915.8125	69.6875	121.625	2834.5

115	15.0625	15.75	2940.3125	70.6875	122.6875	2857.75
116	15.1875	15.9375	2964.75	71	123.875	2880.6875
117	15.3125	16.1875	2989.625	71.8125	125.25	2903.375
118	15.4375	16.3125	3014.5	72.625	126.4375	2927.6875
119	15.4375	16.4375	3038.6875	73.375	127.5625	2952.1875
120	15.4375	16.625	3066	74.375	128.75	2976.6875
121	15.625	16.875	3090.0625	75.1875	130	2999.5
122	15.625	17.0625	3114.25	76.125	131.5	3024.75
123	15.75	17.1875	3138.4375	77.125	133.3125	3046.75
124	15.75	17.1875	3161.875	77.5625	134.75	3071.75
125	15.9375	17.3125	3186.25	78.8125	135.8125	3098.9375
126	16.0625	17.5	3210.25	79.0625	137.625	3121
127	16.25	17.5	3234.25	80.1875	138.8125	3140.6875

128	16.4375	17.75	3258.125	80.8125	140.6875	3160.8125
129	16.5625	17.875	3281.875	81.9375	142.0625	3181.125
130	16.75	17.875	3306.125	82.8125	143.4375	3201
131	17	17.875	3329.9375	83.375	144.9375	3218
132	17.3125	18	3353.9375	84.375	146.4375	3238.5625
133	17.375	18.125	3377.8125	85.125	147.75	3257.9375
134	17.375	18.125	3401.625	86.125	149.125	3273.062
135	17.375	18.3125	3428.3125	86.8125	150.9375	3298.375
136	17.625	18.3125	3452.5	87.5	152.5	3319.25
137	17.625	18.4375	3476.3125	88.0625	154.0625	3341.0625
138	17.75	18.5625	3499.5	88.5625	155.5625	3363.3125
139	17.9375	18.8125	3523.6875	89.25	156.6875	3384.3125
140	18.25	19.3125	3548.5625	90.0625	158.0625	3405.5

141	18.4375	19.3125	3572.5	90.4375	159.8125	3421.6875
142	18.75	19.4375	3600.25	91.8125	161.75	3440.6875
143	18.8125	19.5	3624.25	92.3125	163.5	3457.875
144	19	19.75	3653.25	93.625	166.8125	3475.6875
145	19	19.75	3680.5	95.0625	168.1875	3496.125
146	19	19.8125	3705.125	96	169.375	3515.375
147	19	19.9375	3729.0625	96.5	170.8125	3532.125
148	19	19.9375	3753.1875	97.0625	171.9375	3551.4375
149	19.25	20.0625	3777.75	97.9375	172.8125	3568.6875
150	19.4375	20.125	3802.5625	98.5625	173.875	3587.5

Appendix L – Disclaimer

Disclaimer

This simulation software project and its corresponding documentation entitled "Fencing AI; NeuroEvolution of Augmenting Topologies (NEAT) towards Competition and Survival of the Fittest in a Fencing Simulation" is submitted to the College of Information and Communications Technology, West Visayas State University, in partial fulfillment of the requirements for the degree, Bachelor of Science in Computer Science. It is the product of our own work, except for the utilization of the already existing UnitySharpNEAT.

We hereby grant the College of Information and Communications Technology permission to freely use, publish in local or international journal/conferences, reproduce, or distribute publicly the paper and electronic copies of this software project and its corresponding documentation in whole or in part, provided that we are acknowledged.

Bryan Kent S. Abesamis
Timothy A. Diosaban
Alyssa Danielle R. Magallanes
Neil Ryan S. Sustiguer